Strategies to use Prior Knowledge to Improve the Performance of Deep Learning

An Approach Towards Trustable Machine Learning Systems

JAY MORGAN
Swansea University
Department of Computer Science

Submitted to Swansea University in fulfilment
of the requirements for the Degree of
Doctor of Philosophy

With supervision from:
Monika Seisenberger
Jane Williams
Adeline Paiement

Submitted 2021
53,570 Words.

Abstract

Machine Learning (ML) has been a transformative technology in society by automating otherwise difficult tasks such as image recognition and natural language understanding. The performance of Deep Learning (DL), in particular, has improved to the point where it can be applied to automotive vehicles – a situation in which trust is placed on the ML systems to operate correctly and safely. Yet, while fundamental ML algorithms can be formally verified for safety without much trouble, the same may not be said for DL. A key problem preventing the trustworthiness of DL is the existence of *adversarial examples*, where small changes in input result in catastrophic misclassifications, thereby undermining their use in safety-critical systems.

Using pre-existing knowledge from domain experts has been shown to successfully increase not only the performance but critically the resilience of DL models to adversarial examples. The current thesis developed four different strategies of integrating prior expert knowledge into DL models: feature specialisation, specialised information processing, stimulation of attention mechanisms, and augmentation of training data. Prior knowledge from three scientific domains was used (Quantum Chemistry, Corpus Linguistics and Astrophysics) as case studies to provide a comprehensive framework for evaluation of the strategies performance given different types of data (i.e., text-based, image-based, and graph-based) and model architectures (e.g. recurrent, graph, and convolutional). For the Quantum Chemistry and Corpus Linguistics case studies, two novel datasets are introduced to facilitate the training of prior knowledge informed DL models. Each of the four proposed strategies were tested independently on the case studies to understand their isolated contribution, as well as combined with other strategies to evaluate their interaction.

The results show that, combined, the four prior knowledge integration strategies (a) are an effective method of increasing model performance; (b) result in fewer misclassifications as a result of misleading features; (c) lead to increased model robustness to adversarial examples; (d) create informative representations by visualising learnt representations of prior knowledge; (e) lessen the number of training samples needed to achieve adequate model performance; and (f) lead to better generalisation to different problem tasks other than those the model was trained for. The findings show

the prior knowledge integration strategies used here improve the performance of ML while being more resilient to adversarial examples. This can lead to more trustworthy ML systems in practice.

Contents

List of Figures **i**

List of Tables **v**

Research outputs **xiii**

Acronyms **xvii**

1 Introduction **1**
1.1 Methods . . . 4
1.2 Case Studies . . . 4
1.2.1 Quantum Chemistry . . . 5
1.2.2 Corpus Linguistics . . . 7
1.2.3 Solar/Astrophysics . . . 8
1.3 Summary of Thesis Objective . . . 9
1.4 Contributions . . . 10
1.5 Organisation of Thesis . . . 11

2 Preliminaries **13**

3 Literature Review **19**
3.1 Feature Selection & Extraction . . . 19
3.2 Feature Specialisation . . . 21
3.2.1 Indirect Specialisation . . . 21
3.2.2 Direct Specialisation . . . 23
3.3 Specialised Information Processing . . . 24
3.4 Attention on Data . . . 25
3.5 Generating Adversarial Examples . . . 27
3.5.1 Creating Adversarial Examples for Image Classifiers . . . 28
3.5.2 Creating Adversarial Examples for Object Detectors . . . 30
3.6 Knowledge Representation & Logic-based Methods . . . 34
3.6.1 Knowledge Graphs . . . 34
3.6.2 First Order & Fuzzy Logic . . . 35
3.6.3 Verification of Deep Learning . . . 37

3.6.4 Combining Machine Learning with Interactive Theorem Proving . 38

4 Datasets & Base Models **41**
4.1 Energy Estimation of Quantum Chemical Systems 42
4.1.1 Augmented-QM9 & Crystal Datasets 42
4.1.2 Base Models 44
4.1.3 Experimental Setup 47
4.1.4 Evaluation Metrics 47
4.2 Online Grooming Detection 47
4.2.1 Online Grooming Chat Logs 48
4.2.2 Base Models 50
4.2.3 Experiment Environment 52
4.2.4 Evaluation Metrics 53
4.3 Detection of Type-II Solar Bursts 53
4.3.1 WAVES Dataset 53
4.3.2 Base Model 54
4.3.3 Experimental Setup 55
4.3.4 Evaluation Metrics 56

5 Feature Specialisation **59**
5.1 Background Study on Indirect Specialisation 60
5.2 Application of Indirect Specialisation 61
5.2.1 Integrating Knowledge on OG Processes 61
5.2.2 Estimating Auxiliary Physical Properties 63
5.2.3 Predicting Solar Burst Properties 63
5.3 Direct Specialisation 64
5.3.1 Latent Representations for Word Semantics 65
5.3.2 Direct Specialisation of Usage of Grooming Language 65
5.3.3 Supervised WSR Modification 67
5.3.4 Manifold Learning 67
5.3.5 Elastic Pulling 68
5.4 Experiments 69
5.4.1 Indirect Specialisation 69
5.4.2 Direct Specialisation 72
5.5 Chapter Summary 75

6 Specialised Information Processing **77**
6.1 Background on Encoding Specialised Concepts with Specialised Information Processing 78
6.2 Specialised Information Processing in Recurrent Networks 79
6.3 Specialised information processing in Graphs 81
6.3.1 Specialised Message Production 82
6.3.2 Specialised Node Update 83
6.4 Experiments 84
6.5 Chapter Summary 87

7 Attention on Data **89**
7.1 Background on Attention 90
7.2 Stimulating Attention 92

7.2.1 Direct Attention Stimulation . . . 93
7.2.2 Supervised Attention Stimulation . . . 94
7.3 Experiments . . . 95
7.4 Conclusion . . . 97

8 Augmenting Training Data **101**
8.1 Data-meaningful Transformations . . . 102
8.1.1 Enhancing WAVES Spectrograms with Data Augmentations . . . 103
8.2 Adversarial Examples . . . 105
8.2.1 A Background on Adversarial Examples . . . 106
8.2.2 Manifold Properties . . . 107
8.2.3 Estimating Sparsity/Density . . . 108
8.2.4 Constructing Neighbourhoods . . . 109
8.3 Experiments . . . 111
8.3.1 Adversarial Training . . . 111
8.3.2 Contribution of Data-specific Augmentations . . . 117
8.4 Chapter Summary . . . 123

9 Combining Prior Integration Strategies **127**
9.1 Online Grooming Detection . . . 127
9.1.1 Conclusion of Combining Corpus Linguistics and Deep Learning . 131
9.2 Chemical Energy Estimation with Deep Learning . . . 133
9.2.1 Ablation Study . . . 135
9.2.2 Effectiveness for Estimating Stable Configurations . . . 136
9.2.3 Generalisation to Larger Systems . . . 136
9.2.4 Interpretation of the Atoms' Hidden States . . . 138
9.2.5 Conclusion of Combining Chemical Physics Knowledge and Deep Learning . . . 139
9.3 Solar Burst Detection with Deep Learning . . . 140
9.3.1 Comparison with Baselines . . . 140
9.3.2 Conclusion on Combining Solar Physics Knowledge with Deep Learning . . . 141
9.4 Effect of Prior Knowledge on the Amount of Required Data . . . 142
9.4.1 Data-Sampling Process . . . 142
9.4.2 Results of Energy Estimation with Limited Data . . . 144
9.4.3 Results of Limited Online Grooming Detection Chat Logs . . . 145
9.5 Chapter Summary . . . 147

10 Conclusion **149**

Bibliography **153**

Appendices **167**

List of Figures

1.1 Overview of applying Deep Learning to a graph-based quantum chemical scenario 6
1.2 The online grooming detection process using Deep Learning 7
1.3 Different types of SRBs that are used as image-based inputs to the object detector. (Left) example of background noise and interference with no burst present. (Middle) includes a Type II burst outlined with a white square. (Right) examples of Type III bursts. Due to the presence of Type III overlapping with Type II bursts, in addition to various bands of noise, detection of Type II bursts can sometimes be a difficult task. 9

4.1 Example energy curve where the stable configuration occurs where the energy value is at its lowest. 42
4.2 Infinite Crystal (left) and Growing Crystal (centre: seed, right: growing) structures. 43
4.3 Frequency of system sizes available in the Augmented QM9 dataset out of 10K molecules. 45
4.4 Overview process of constructing OG chatlog corpus. CL analysis is required to generate a series of word variant pairs and 3-word collocates. The output of this analysis is combined with the original conversations from PJ and the non-grooming subset of PAN2012. 50
4.5 OGD-R (left) & OGD-T (right), for integration of CL priors. Orange and green indicate where word variants and OG processes priors may be integrated, respectively. 52
4.6 Example of WAVES spectrogram. 54

5.1 An additional OG process detection branch (green) is added to the classifier module (in OGD-E for illustration). This additional OG process branch predicts the a pseudo-likelihood of OG process for each word in the conversation. 62
5.2 Depiction of the Elastic Pulling process between two word-pair variants. 68
5.3 Graphical representation of the DNN's predicted likelihood of the *Compliance testing* OG process. Higher likelihood values are indicated in red, while lower values are represented by black colours. 70

5.4 Effect on DSG and AE performance when the auxiliary outputs are incrementally added. 71

6.1 Visualisation of the OG process information processing for LSTM (left) and GRU (right). Activation gates are shown in squares, with addition, subtraction, and multiplication of vectors in circles. 82
6.2 One TSNE visualisation of different BT messages produced by Augm-MPNN for first 1000 molecules in the testing data. For clarity, the different BT messages (single, double, triple bonds) are split into three separate plots, with a fourth plot showing all BT messages together in a single plot. Messages were extracted from the output of the final BT specialised update function (Equation 6.10 implmentation 2) before the final scalar energy prediction is made. 87

7.1 Visualisation of attention weights from [1] for local attention (left) and global attention (right). Entries in these matrices demonstrate the importance of each English word (column) for its German translation (row). 91
7.2 AUPR results of OGD-T with direct augmentation and stimulation using different layers of self-attention. Performance is shown by solid lines and dotted lines are the trend lines. 95
7.3 Comparison of confusion plots of OGD-R/OGD-T without prior knowledge augmentations and with supervision+direct stimulation LSTM. Colour intensity of cells denotes the log scale of number of conversations for readability. 98

8.1 Original image (left) and augmented version with the removal of a random segment (right). The red rectangle highlights the difference between the original and augmented image. 104
8.2 Example of **D4** augmentation. Left figure shows the erosion of Type II burst, where the higher frequencies are more eroded than those at lower frequencies. Right figure dilates the same burst, increasing the dilation for lower frequencies. 104
8.3 Augmented image with horizontal noise (left) and vertical noise (right). 105
8.4 Original (left) and **D4** augmented (right) event. 105
8.5 Example where a data point x_i lies close to the class decision boundary. In these situations, too large ε values may push the synthetically generated point over true class boundaries. 108
8.6 Example scenario where true class boundaries are revealed when more data is collected. 108
8.7 Iterative ε-expansion process in a binary class scenario. The two classes are distinguished by the dotted and solid circles. 111
8.8 Proposed adaptive neighbourhoods for the Iris dataset. The three classes of flower are represented by different shaped markers. The size of the neighbourhood for each sample is indicated with a circle centred on the data point. Intersections between neighbourhoods of different classes are not real but are visualisation artefacts coming from the 2D projection of 4 dimensions. 112

8.9 Adversarial example generation methods applied to an solar event containing both Type II and Type III bursts. Top row shows the original predictions made without applying any adversarial attacks. The second row shows PGD, third FGSM, and bottom row DAG method. Left column images are the original event with a red rectangle to highlight the Type II burst, with the centre column being the amount of pixel-wise perturbation. The result of the perturbation is shown in the right column with predicted detections made by Faster R-CNN in blue rectangles. 115
8.10 Boxplot of F_1 score for Faster R-CNN trained with varying data augmentations. 119
8.11 Effect of λ parameter on the amount erosion. In the left figure $\lambda = 0.5$ and right $\lambda = 2.0$. Figure (right) demonstrates an increased erosion of burst segments in higher frequencies than those of left figure. 119
8.12 Boxplot of Precision, Recall, and F_1 score between two different forms of erosion. 120
8.13 PR and ROC curves for Faster R-CNN trained on augmented and non-augmented data. Figures (a) and (b) are for the augmented data, while (c) and (d) are for non-augmented data. For each plot, the area under curve (AUC) score is shown where higher numbers are better. 121
8.14 PR Gain curves for Faster R-CNN trained with non-augmented data ('Base model', left) and trained with augmented data ('Using augmented data', right). 122
8.15 Three example inputs to the Faster R-CNN. Images in the first column are the input to the trained models. Middle column images are the predictions by Faster R-CNN trained on non-augmented data. Right column are the predictions made by Faster R-CNN trained with augmented data. 123

9.1 Performance of base and augmented models on the UCG dataset. For the augmented models, we test with all augmentations and without the auxiliary atom type prediction. 135
9.2 Energy estimations at different scalings of picked molecules for base (left) and augmented (right) GNNs. Top: $C_5H_9NO_2$, bottom: $C_6H_6O_3$. . 137
9.3 Energy estimations (red for Augm.-MPNN, green for MPNN) at different scalings for a small (15 atoms, left) and large (75 atoms, right) crystalline system. Blue is ground truth. 138
9.4 Impact of system size on accuracy when trained on systems of up to 75 (top) and 25 (bottom) atoms from UCG. 138
9.5 Contribution of a moving atom (left, red curve) and mean contribution of static atoms (right, red curve) toward the energy estimate of a crystal (black curves) by Augmented-SchNet. 139
9.6 Robustness of base and augmented models to smaller training sets for Augmented QM9, when sampling on molecule types. Top: effect on energy estimation accuracy (AE), bottom: effect on finding stable configuration (DSG). Left: MPNN, right: SchNet. Blue: base model, red: augmented model. 144
9.7 Robustness of base and augmented models to smaller training sets for the SCG subset. Left: MPNN, right: SchNet. Blue: base model, red: augmented model. 145

9.8 Robustness of base and augmented models to smaller training sets for Augmented QM9, when sampling on molecule's scaling. Top: effect on energy estimation accuracy (AE), bottom: effect on finding stable configuration (DSC). Left: MPNN, right: SchNet. Blue: base model, red: augmented model. 146
9.9 Performance of augmented and non-augmented OGD-R (top) and OGD-T (bottom) on varying sizes of training datasets. Dotted grey lines indicate the highest performance. 147

List of Tables

1.1 The use of different prior integration strategies in each of the case studies. 4

3.1 Summary of Adversarial example generation methods for object detectors. 33

4.1 Number of stable and OoE systems in our datasets 44
4.2 Statistics on the Augmented QM9's molecules 44
4.3 Summary statistics of continious variables in Augmented-QM9 dataset. . 45
4.4 Available properties in both the Augmented-QM9 and all Crystalline datasets. 46
4.5 Processes used by groomers in order to establish a connection with a child. 49
4.6 Breakdown of the conversations, sessions, and messages between groomer and non-groomer users. 49
4.7 Message statistics for the groomer subset of the corpus. 50
4.8 Available properties in OG Chatlog dataset 51
4.9 Data properties of OG collocates 51
4.10 Number of samples in the train and test set per burst and activity type. Data is shown in the format: frequency (percentage). 55

5.1 Comparison of occurrence frequencies for selected variants and all words in the corpus. 66
5.2 Impact of each CL augmentation on OG classification. Bold are improved results. 69
5.3 Impact of each physics integration strategy on energy estimation on the Augm. QM9 dataset for MPNN (top) and SchNet (bottom) base models. Results are presented in the format: mean (std). The specialised interaction methods that optimise at best energy and geometry are highlighted in bold for each model. 70
5.4 Comparative performance results between base Faster R-CNN and Faster R-CNN with auxiliary target prediction of solar activity. Results are shown over 10 trails in the format: mean (standard deviation). 72
5.5 Impact of each CL augmentation on OG classification. Bold are improved results. 73

5.6 average distance between pairs of selected variants $\overline{\mathcal{D}}_{var}$ and all other pairs of words $\overline{\mathcal{D}}_{non\,var}$ in the wsr spaces 73
5.7 WSR algorithms applied to varying GloVe representations. Bold values denote the improvement over no augmentation. 74

6.1 Performance of both base models OGD-R and OGD-T when using Specialised information processing with LSTM and GRU cells. Metrics are compared against their baseline cells. 85
6.2 Impact of each domain knowledge integration strategy on MPNN (top) and SchNet (bottom) for Augm. QM9. Results are in the format: mean (std). The specialised interaction methods that optimise at best energy and geometry are highlighted in bold for each GNN. 86
6.3 Impact of each specialised interaction strategy for MPNN with $\alpha = 0$ (no generic interaction used) on Augmented QM9. Results are in the format: mean (std) over molecules 86

7.1 Impact of each CL knowledge integration on OG classification. Bold are improved results with respect to no augmentation, i.e. base models. . . . 96

8.1 F_1 score of DNN for the Iris dataset using various adversarial defence methods. Scores are in the format: mean (standard deviation) over 10 k-folds. Bold font face indicates the best form of attack for each type of defence method. 113
8.2 F_1 score performance on the WAVES dataset using Faster R-CNN. Numbers highlighted in a bold font face indicate the best achieving adversarial attack for each form of defence. 116
8.3 IoU performance on the WAVES dataset using Faster R-CNN. Numbers highlighted in bold indicate the best achieving type of adversarial attack for each form of defence. 116
8.4 Performance results for Faster R-CNN trained with different augmentations applied to the training data. Results are in the format: Mean (Standard Deviation) of 10 trails. Best performance over 10 trails is highlighted using a bold font face. 118
8.5 Comparitive performance for the effect of λ in Equation 8.1. 120
8.6 Comparative results between training Faster R-CNN with and without Augmentations. 121
8.7 IoU scores for best performing Faster R-CNN trained with (non)-augmented WAVES data. 122

9.1 Summary of knowledge integration strategies applied singularly to the base models. Bold are improved results with respect to no augmentation, i.e. the base model. 129
9.2 Comparative evaluation of OG classification methods. Best performing metrics are presented in a bold font face. 130
9.3 Area under PR Gain curves for both base and augmented models. 130
9.4 Progressive additions of CL-augmentations to a simple LSTM model similar to OGD-R with no pre-training of WSR 131
9.5 Performance of base models and their augmented counterpart when swapping word variants in the testing data. 131

9.6 Impact of each domain knowledge integration strategy on MPNN (top) and SchNet (bottom) for Augm. QM9. Results are in the format: mean (std). The specialised interaction methods that optimise at best energy and geometry are highlighted in bold for each GNN. 133
9.7 Evaluation of the base (left) and augmented (right) models. For each DNN, best results between base and augmented models are highlighted in bold. 134
9.8 Ablation study and effect on handling different atom types in crystals of UCG 136
9.9 Performance on WAVES dataset for different models. With exception of [2], whose model type is determinisic, metrics are reported over 10 trials in the format: mean (standard deviation). 141

10.1 High level implementation style for the different case studies presented in this thesis. 179

Declaration

This work has not previously been accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

Signed [redacted].....

Date 06/08/2021.....

STATEMENT 1

This thesis is the result of my own investigations, except where otherwise stated. Where correction services have been used, the extent and nature of the correction is clearly marked in a footnote(s). Other sources are acknowledged by footnotes giving explicit references. A bibliography is appended.

Signed [redacted].....

Date 06/08/2021.....

STATEMENT 2

I hereby give consent for my thesis, if accepted, to be available for photocopying and for inter-library loans after expiry of a bar on access approved by the Swansea University.

Signed [redacted].....

Date 06/08/2021.....

Acknowledgements

Nothing is made in isolation, and this thesis is no exception. Throughout the production of this work, I have relied on the help and support of my family, friends, and colleagues. It is to all of them that I give my thanks. A special thank you to my supervisors without whom, creating the work in this thesis would not be possible. During the course of my candidature, they have provided not only their expertise, but also their encouragement, and guidance for all my ventures.

This thesis was supported by a joint fund between the College of Science and the Hillary Rodham Clinton School of Law, Swansea University.

Research outputs

Grants

- **Developing Resilience against Online Grooming (DRaOG)** – This thesis includes content relating to integrating Corpus Linguistic knowledge into a Deep Learning application to detect online grooming in chat logs. The grant application was based on the research outputs of 'Integrating linguistic knowledge into DNNs: Application to online grooming detection'. This work and its outputs were included in a grant application to further collaborate with police forces and social workers in the UK. This grant is a collaborative effort between the Department of Computer Science, and College of Law, Swansea, and Université de Toulon, France. More information about the grant and the funding body can be found on the End Violence Against Children website at `https://www.end-violence.org/grants/swansea-university` and at the Project's website: `https://www.swansea.ac.uk/project-dragon-s/`
  - Project Name: DRAGON-S
  - Awarded: January 2021
  - Awarding Body: End Violence Against Children (EVAC)
  - Funding Amount: $519,420

Articles

- Crook, T., Morgan, J., Pauly, A., & Roggenbach, M. (2021). A Computable Analysis perspective on (Verified) Machine Learning. arXiv preprint preprint. arXiv:2102.06585. Publication under review
  - This work outlines the systematic properties involved in verifying the robustness of classifier models within *stable regions*. From this, we apply these principles in the formation of our work for Adaptive Neighbourhoods.

  - Theoretical background of machine learning and collaboration in the design of formulation was done by Morgan, J..

- Morgan, J., Paiement, A., Pauly, A., & Seisenberger, M. (2021). Adaptive Neighbourhoods for the Discovery of Adversarial Examples. arXiv preprint arXiv:2101.09108.
  - This paper presents the theoretical work of adaptive neighbourhoods, to which experimental evidence of their effectiveness are demonstrated in this thesis.
  - Design of method and implementation was done by Morgan, J..

- Morgan, J., Paiement, A., Lorenzo-Dus, L., Kinzel, A., Di Cristofaro, M. (2020) Integrating linguistic knowledge into DNNs: Application to online grooming detection. Paper under review.
  - Our work on the detection of online grooming from text-based conversations, including the dissemination of the construction of the corpus, and the key methodologies to incorporate linguistic knowledge into the design of neural networks to produce a state-of-the-art classifier.
  - Acquisition of the data was done by Morgan, J.. Linguistic analysis was accomplished by Lorenzo-Du N., Kinzel A., and Di Cristofaro M.. The design of methods was done by Paiement A. and Morgan J., and implementation and testing was done by Morgan J..

- Morgan, J., Paiement, A., & Klinke, C. (2020). VIMPNN: A physics informed neural network for estimating potential energies of out-of-equilibrium systems. Paper under review.
  - This paper introduces both the various crystallographic datasets and physics-informed methods to improve the estimation performance of graph-based neural networks.
  - Simulation of the dataset was done by Morgan, J. with guidance on the correct physics properties by Klinke, C.. Design of the methods by Paiement, A. and Morgan, J. with implementation and testing by Morgan, J..

- Morgan, J., Paiement, A., Seisenberger, M., Williams, J., & Wyner, A. (2018). A chatbot framework for the children's legal centre. Frontiers in Artificial Intelligence and Applications, 313, 205–209.
  - This work incorporates the use of auxiliary losses in a multi-task learning framework to better handle conversational steps towards the classification of children's right to legal representation.
  - Design of the methods, implementation, and testing done by Morgan, J..

Talks

- 2021-07-06 - Computability in Europe (CiE) 2021, Ghent University, Belgium - A Computability Perspective on (Verified) Machine Learning, Tonicha Crook, Jay Morgan, Arno Pauly, and Markus Roggenbach - Talk presented by Tonicha Crook.

- 2021-03-30 - British Colloquium for Theoretical Computer Science (BCTCS) 2021, Liverpool, United Kingdom - Trustable Machine Learning Systems. `https://blog.morganwastaken.com/2021-03-29/BCTCS-2021-Presentation`
- 2020-12-07 - Bio-Ontology Research Group (BORG) Seminar, King Abdulla University of Science and Technology (KAUST), Jeddah, Saudi Arabia - Using Logic to Predict Protein Sequence Functions
- 2019-04-09 - Edinburgh Science Festival 2019 - Developing Resilience against Online Grooming
- 2019-01-08 - Japan Advanced Institute of Science and Technology (JAIST) Research Seminar, Nomi, Japan - Verification of Machine Learning Models
- 2018-12-13 - International Conference on Legal Knowledge and Information Systems (JURIX) Conference 2018, Groningen, Netherlands - A Chatbot Framework for the Children's Legal Centre

Datasets

- **Crystallography & Molecular Dataset** – A novel dataset for training Deep Neural Networks on out-of-equilibrium crystal and molecular formations to predict system energy values. For full details on its design, and simulation of this dataset, please refer to Section 4.1.1.
- **Online Grooming Chat Logs** – A new dataset has been developed to facilitate the training of Machine Learning algorithms to detect of online grooming. This data, through a collaboration with Corpus Linguistic experts, includes full transcripts of conversations from convicted groomers, and adds various forms of linguistic annotations. Details on this dataset can be found in Section 4.2.1.

Acronyms

Al	Aluminium.
AUPR	Area Under Precision-Recall.
BCE	Binary cross-entropy.
BT	Bond Type.
CL	Corpus Linguistics.
CNN	Convolutional Neural Network.
Cu	Copper.
DFT	Density Functional Theory.
DL	Deep Learning.
DNN	Deep Neural Network.
FCC	Face-Centred Cubic.
GMM	Gaussian Mixture Model.
GRU	Gated Recurrent Unit.
IoU	Intersection over Union.
LSTM	Long-Short Term Memory.
ML	Machine Learning.
MTL	Multi-task Learning.
OG	Online Grooming.
OGD	Online Grooming Detection.
OoE	Out-of-equilibrium.
ROC	Receiver Operating Characteristic.
RPN	Region Proposal Network.

SCG Stable Crystal Growth.
SRB Solar Radio Burst.
STL Single-task Learning.

UCG Unstable Crystal Growth.

WSR Word Semantics Representation.

Chapter 1

Introduction

While Deep Neural Network (DNN)s are good at learning to perform tasks such as image recognition with high degrees of accuracy, they can be susceptible to slight changes (often called perturbations) to the input data [3, 4, 5]. For instance, given a DNN that learns to recognise if an image contains a car, slight changes to a small number of pixels within this image (even those that are not located on the car itself) may alter the DNNs output to now predict the image contains a plane [6]. If the DNN's decision changes due to this perturbation, the resulting image that creates this change of decision is called an adversarial or counterexample.

This issue of DNN sensitivity to perturbation poses a troubling question for the community: If systems use DNNs - especially those systems where safety is paramount [7] - it could be difficult to predict if they will operate correctly in the face of slight erroneous input. Consequentially, predicting and detecting failures in systems composed of Deep Learning (DL) models could be challenging to determine. For instance, as there is evidence to show that image-recognition systems can be rendered inaccurate through small changes to specific pixels, it becomes more challenging to trust an autonomous vehicle, where slight defects can occur due to sub-optimal camera conditions such as damage or object occlusion [8, 9]. Predicting and troubleshooting these types of issues in safety-critical systems is necessary, but it is not simple to interpret the decisions made by Machine Learning (ML) classifiers as with the traditional logic-based methods.

If indistinguishable changes to specific pixels in an image may lead to a misclassification, one may question the modelling mechanisms (such as feature learning) the DNN is attending to for its classification. Are these features an artifact present in the training and testing data but would not realistically occur in real-world scenarios? Furthermore, are these features consistent with what an expert in this domain might say are essential when classifying the same data?

One way to help address this concern of trustworthiness is with prior knowledge. Here the pre-existing wealth of knowledge from domain experts can inform the DL model and improve the confidence in trusting the output of the DL model. This inclusion of prior knowledge may help the model in four aspects. Firstly, DNNs, as opposed to other more traditional ML methods, require far less data pre-processing and feature extraction to be accurate models. The consequence is that it is possible to encode the data in a format applicable for the DNN and have the model learn from the supervision of examples. However, given that real-world data can be noisy and sometimes erroneous [10, 11, 12], one may be less confident that what the model is learning is beneficial for its task. If, for example, the errors occur only in one of two classes the model needs to discriminate between, it may use the occurrence of these errors to justify its class prediction. The DNN, in this case, is performing the task as instructed – finding and using discriminatory features available in the data – but these features it is finding during the learning process are superfluous and are not helpful for real-world tasks, thus rendering the DNN ineffective for its task. Prior knowledge may improve the learning process by constraining the features that it may attend to for classification or regression. By helping the model attend to particular, and indeed, essential features for classification, it may help the DNN improve its performance on a task and be more resilient to minimal changes in the input space. Moreover, using prior knowledge, one may adapt and specialise computation mechanisms of the DNN to match the knowledge of the problem better. For instance, this thesis introduces methods to specialise recurrent and graph-based DNNs to known representations in the domain. The latent representations of the examples are more useful for the classification or regression and possibly provide more informative visualisations of these latent representations to understand the DNN's decision processes.

Secondly, features could become more informed by the domain constraints, making features more informative for the decision process [13]. These features, when visualised, could give analysts more indication as to why inputs result in specific classification labels. As a simple example, visualising the DNN's attentiveness to important features may lead to insights as to what the model is learning for its decision process. With the expert's insight of feature importance these visualisations can be tailored to check for inconsistencies with the expert's opinion.

Thirdly, DNNs can learn many different functions, with some being more accurate and less sensitive to noisy data than others. The learning process is then searching through the possible functions to find one that minimises the errors on the dataset. Through the introduction of priors, the functions it is possible to learn may be restricted and reduced [14], thereby making it easier to find the optimal function, where this optimal function is both more accurate and more robust to noise, and closer to how the expert would deal with the same task.

Finally, DNNs typically require large amounts of annotation to infer general statistical patterns in supervised tasks, and increasing the amount of annotations can also improve the effectiveness of existing models [15]. However, when the number of annotations is limited, so too may be the performance of these models. In scientific domains outside of ML research, such as physics, medical, and social sciences, it is common to have a substantial amount of data without any annotations, thereby limiting the usability of supervised learning with DNNs on these datasets. Methods such

as weak supervision, few shot learning, and transfer learning, can help to address the small amounts of annotation. Indeed, these methods are also used in this thesis, but our results show that prior knowledge may be another complementary way to address this problem, helping DNNs learn from the small amount of annotations by providing an inductive bias to the learning process.

The aim of this research is to make DNNs more *trustable* or trustworthy, where trust in this sense that the model is not using superfluous or erroneous indicators in the data, that it is less susceptible to noise and limited amounts of annotations; that it is generally more accurate; and that it provides a means of communicating feature importance via visualisation. To address this aim, methods to integrate prior expert knowledge were developed. There may be other methods to make an ML system more trustworthy. Principle candidates include formal verification or the introduction of logic into DNNs. However, we focus on prior knowledge for two reasons: firstly, the work of formal verification introduces issues of scalability – it can be impractical to formally verify transformers due to their size and non-linear complexity. Secondly, while logic can be used to represent the knowledge of experts, creating representations of this knowledge in logical forms for these scientific domains can be challenging. Therefore, in order to produce a reasonable solution to the questions posed in the scientific domains and case studies, we propose the use of integrating knowledge into the design of DNNs better take advantage of prior knowledge and the success of newer ML techniques.

This research has explored the theoretical and general perspectives and concrete examples by working with experts from Quantum Chemistry, Corpus Linguistics, and Astrophysics. In these ventures, the thesis has outlined the fundamental principles on integrating prior expert knowledge into DNNs to improve its performance and robustness in these domain-specific tasks, and visualise the outputs to ensure the classification and the attended features match what is understood by the experts. These methods were thoroughly evaluated using many different forms of architectures, including recurrent, graph, and convolution-based models, various problem tasks that demonstrate how well these principles help. The principles behind integrating prior expert knowledge are shown to be effective at improving task performance, providing means of visualisation, and addressing situations with small amounts of training data. This work has additionally shown that these strategies can be generalised to many networks and tasks, thus leading to potential improvements in other domains other than those used here.

While we work with experts from different scientific domains, the methods presented in this thesis take advantage of varying *levels* of knowledge – from the knowledge located within the data itself (such as adversarial training); or some basic understanding that can be gained from a principled investigation of the data or domain; to the knowledge of domain experts (of which examples of this knowledge appear in this thesis: understanding of online grooming tactics, feature properties of Type II bursts, etc.). Our work is advantaged by this approach as, more basic forms of knowledge may be easier to integrate, but more complex forms provide larger benefits. These various forms of methods ultimately provide flexibility to the implementer of the DNN in the domain and may be applicable to a wider audience other than the expert in said domain.

Table 1.1: *The use of different prior integration strategies in each of the case studies.*

Prior strategy	Quantum Chemistry	OG Detection	Solar Burst Detection
Feature Specialisation - Indirect	x	x	x
Feature Specialisation - Direct		x	
Specialised Information Processing	x	x	
Attention on Data		x	
Augm. Training Data - Adversarial Training			x
Augm. Training Data - Data-specific Augm.			x

1.1 Methods

The strategy to incorporate prior knowledge into DNNs may take many forms. In this thesis, various approaches are explored and evaluated to create viable strategies that practitioners may use when adapting DNNs to their problem tasks. Each of these approaches receives a dedicated chapter within this body of work. The strategies that make up this work are:

1. **Feature specialisation** – the process of making latent features more specialised towards the task via two complementary approaches: a) indirect specialisation with multi-task learning; b) direct specialisation by modifying existing features.
2. **Specialised information processing** – additional internal model computations that provide specialised feature extractor networks each with unique parameters.
3. **Attention on data** – improve the model's attentiveness to important input features through stimulation of the model's activation towards these features.
4. **Augmenting training data** – enhancing the training data through adversarial training and data-specific augmentations. These augmentations aim to improve the DNN's robustness towards noisy and misleading features using the prior knowledge to determine meaningful augmentations.

Not all methods are applicable for each of the various case studies (discussed in the next section). Table 1.1 outlines how each prior strategy was tested, where checked columns indicate that this case study was used to evaluate this strategy.

1.2 Case Studies

The objective of this thesis was the design of strategies for the integration of prior domain knowledge into DNNs. Following this objective, it was necessary to work with experts in different domains to find common themes of using prior knowledge to leverage DL in their domain. Three case studies were used to cover different aspects of scientific research. These case studies enabled the testing of the prior integration

strategies on varying architectures, in addition to various forms of data such as graphs, text, and images.

As one would expect, each of these areas had challenges it wished to address using DL. Therefore, the relevant strategies were selected and used to design implementations suited to the data available and the DNN architecture in question. The data available for the problem task and the DNNs used are described in Chapter 4.

These three areas are:

1. **Quantum Chemistry** – using graph representations and graph-based DNN regressors.
2. **Corpus Linguistics** – using textual representations along with recurrent and transformer classifiers.
3. **Solar Physics** – using image representations with object detectors.

1.2.1 Quantum Chemistry

The first case study considers the domain of chemistry. In this study, chemicals can be represented as graphs and processed by graph-based DNNs to estimate their properties traditionally computed using chemical simulations. Chemical simulations have practical industrial applications, e.g. drug or material discovery [16]. Simulating crystal systems, in particular, provides useful properties such as surface absorption, chemical reactions, and surface magnetism [17]. Simulations can also be used for the calculation of *potential energies* under different physical conditions (known as Equation of State), such as the positions of interacting atoms. By varying the atomic positions and calculating the energy values at these different positions, it is possible to find the positions at which the atomic interactions are at their most stable. Despite chemical simulations being capable of this task, they typically require large amounts of computing resources and do not scale well to larger system sizes [18, 19], even when simplified using Kohn-Sham Density Functional Theory (DFT) [20] based on electronic density in place of individual electrons [21]. As many interesting and realistic systems are formed from a large number of atoms, a computationally efficient, scalable, and accurate method for chemical property estimation would be desirable [22].

DNNs may help in the discovery of stable chemical systems by quickly estimating the potential energy at different spatial configurations, thereby reducing the search of possible configurations to be later verified with classical chemical simulations. Therefore, in this case study, we were concerned with estimating the potential energy of *Out-of-equilibrium (OoE)* molecular and crystalline systems, where the atoms are at positions that are not at the minimum of potential energy. These energies are determined at the electronic ground-state at given positions of atoms for static systems.

This study was supported with new datasets of diverse and OoE molecules and crystals. The MD17 [24] and ISO17 [16] datasets provide the potential energy and interatomic forces for eight small organic molecules (MD17) and 129 isomers of C7O2H10

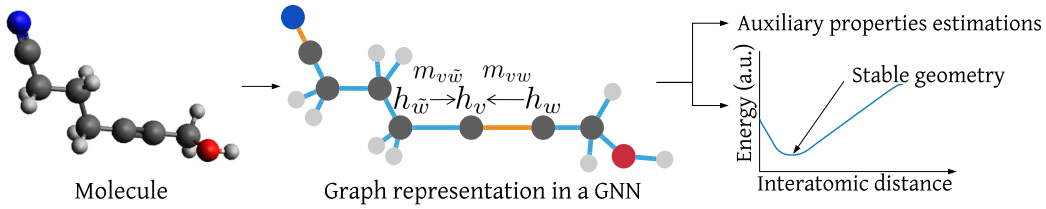


Figure 1.1: *Overview of applying Deep Learning to a graph-based quantum chemical scenario. A Molecular/Crystalline system defined by the properties of positional coordinates of atoms, type of atoms, and types of bonds are fed to a Deep Learning network to predict the energy of a system. Graph representation of a molecule (left from [23]) for the estimation of potential energy through passing of messages between interacting atoms. Nodes are atoms and colour denotes different atom types. Edges link chemically bonded atoms and colour denotes different bond types. Non-bonded atoms may also share edges in fully connected graphs, but these edges are not represented for readability.*

(ISO17), with (independent) perturbations of their atoms' positions. QM9 [25, 26] contains a more diverse set of 134k molecules using carbon, hydrogen, oxygen, nitrogen, and fluorine atoms, but at their stable configuration only. The QM9 dataset was augmented [25, 26] with OoE configurations for 10k of its molecules at a [90%, 150%] range of interatomic distances. In addition, two datasets composed of infinite crystals and finite growing crystals of Aluminium (Al) and Copper (Cu) atoms were created. The details of these datasets can be found in Chapter 4.

Through this case study, this work has demonstrated that integrating prior knowledge on the problem into DNNs improves both the accuracy of potential energy estimation and the applicability of the DNN to estimate well for more diverse chemical systems. At the same time, it has highlighted the applicability of graph-DNN for the energy prediction of OoE molecular and crystalline systems, where the atoms are at positions that are not at the DFT calculated minimum of potential energy. Such systems are represented as chemical graphs, with nodes denoting individual atoms and edges the type of bond between them, as illustrated in Figure 1.1.

Integrating prior physics knowledge into the design of the DNN was shown to be an effective strategy, leading to an improvement in accuracy and generalisation power to more diverse chemical systems. Specifically, two physics integration strategies were used: (1) the prior knowledge of the covalent bonding of atoms to produce a specialised architecture that takes advantage of the input bonding information; (2) introduction of auxiliary estimations to further relate internal representations to relevant physical properties. Knowledge of covalent bonds between pairs of atoms was used to improve the DNNs' accounting for atomic interaction. While auxiliary estimations was used to further relate internal representations to relevant atom-wise and system-wise physical properties. These enhancements were applied to both graph-based MPNN and Convolutional Neural Network (CNN) SchNet, to significantly improve their performance and produce the new state-of-the-art models to the estimation of energy of OoE chemical systems.

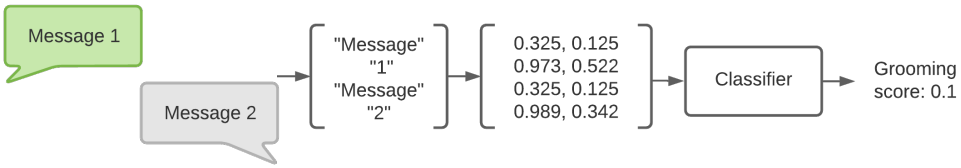


Figure 1.2: *The online grooming detection process. Text-representation of chat logs are converted into a tokenised representation, where tokens represent words, or sub-words. Tokens are further represented by vector-based forms (word embeddings) where clustered vectors mean similar things. These vectors are used by DNN to produce a 'grooming' score. Higher scores indicates an increased level of grooming occuring within the conversation. This process is the basic example of classification of conversations with DL, to which we apply CL prior knowledge.*

1.2.2 Corpus Linguistics

This second study has provided a means of testing the integration strategies with text data. For the classification of this type of data, recurrent and transformers were used and adapted from the domain knowledge of Corpus Linguistics. Corpus Linguistics (CL) uses a set of texts (corpora) to study the usage of language within text and provides quantitative analysis of the patterns within the usage of language through statistical methods [27, 28]. Common examples of the output of CL analysis include the co-occurrence of words that form common collocations, where such insights might be invaluable for discriminatory analysis in other classification tasks.

The objective of this case study was the identification of Online Grooming (OG) using a combination of CL and DNNs. OG is a communicative process of entrapment in which an adult lures a minor into taking part in sexual activities online and, at times, offline [29, 30]. The work aimed to detect instances of OG through the classification of whole conversations. In the context of a law enforcement investigation, the main aim of such automatic processing of large databases is to allow human investigators to review conversations that are flagged as at risk in more detail. Classification in this context then requires the ability to capture subtleties in the language used by groomers and provide this knowledge to the law enforcement investigators.

The groomer messages' theme and immediate purpose may vary throughout the conversation to achieve the overarching goal of entrapping the victims [29]. Groomers use a series of inter-connected "sub-goals", or communicative processes, referred to as *OG processes* here, namely gaining the child's trust, through sharing personal information, planning activities, building a relationship, isolating them emotionally and physically from their support network, checking their level of compliance, with groomer-proposed activities, introducing sexual content and in some cases, trying to secure a meeting offline. The language used within these processes is not always sexually explicit, which makes their detection more challenging. However, CL analysis also flags some contexts associated with the OG processes, in the form of word collocations (i.e. words that occur within the same window of 7 words) that tend to occur more frequently and therefore can be associated with OG processes. We propose to exploit the relations between the OG processes and their overarching goal of OG to improve the final OG classification. We use the CL identified context windows to guide the learning of our DNN.

In this case study, combining the two disciplines of CL and DL has helped to create a classifier to detect grooming chat logs. This study has enabled the testing of the prior integration strategies in text-based classifiers, specifically recurrent networks and transformers. This base case of classification of text can be seen in Figure 1.2. This basic process was augmented to include CL prior knowledge to enhance the classifier's performance while also providing mechanisms for an explanation of the decision process. The work demonstrates that, when integrated into DNNs, the products of CL analysis may allow the better capture of language subtleties while simplifying and guiding the learning task. Furthermore, it was shown that CL knowledge may help law enforcement interpret the DNN decision process towards producing evidence for potential prosecution.

1.2.3 Solar/Astrophysics

Our third and final case study tested our proposed prior knowledge integration strategies in image-based object detectors. For this scenario, we used the study of Solar physics. Solar physics is the study of mechanisms behind Solar activity and the relationship between our Sun and its environment. One solar phenomenon of interest are Solar Radio Burst (SRB)s. These bursts are caused by plasma ejecting from the Sun creating friction with solar windows, creating a low-frequency electromagnetic signal.

While historically SRBs were defined by their morphology, SRBs can be categorised based on their physical properties, such as the frequency range in which they occur, duration, intensity, and the speed of the decay of the frequency [31]. These categories are labelled *Type I, Type II, Type III* and so on. In Figure 1.3, we show the regular depiction (a spectrogram) of solar bursts with time along the x-axis and frequency along the y-axis. From this, we see background radiation, a group of Type III bursts, and a labelled Type II burst. Type II drifts decay slower than Type III bursts, creating a distinctive visual difference between these two types of bursts.

While SRBs may be caused by various factors, Type II SRBs can be indicative of coronal mass ejections (CME), where energy stored in the magnetic fields of a star build up and result in an explosive release of electromagnetic and particle radiation. Usually a CME has no noticeable effect on Earth. However, in rare cases, the electromagnetic radiation interacts with the Earth's magnetosphere causing disruptions to radio transmissions, satellites, and electrical transmission lines [32, 31]. In some cases, electrical devices can either be impaired or operate incorrectly, such as with *cosmic bit flips* where binary data stored in a computer's memory can change from a 0 to a 1 and vice versa. This event is known to have caused an error in the 2003 Belgian election, where a candidate received more votes than is possible [33]. Thus it may be necessary to expect the arrival of a CME in order to shut down devices to prevent any damage. CMEs not only have an effect on our computer hardware, but they may also pose more direct damage to the health of astronauts and pilots, so detecting them early may be essential to protecting the health of these individuals.

As SRBs travel at a speed of light, it is possible to use the presence of Type II SRBs to predict an upcoming CME event. However, detecting these solar bursts could require individuals to constantly monitor the data captured by solar burst detectors. Neverthe-

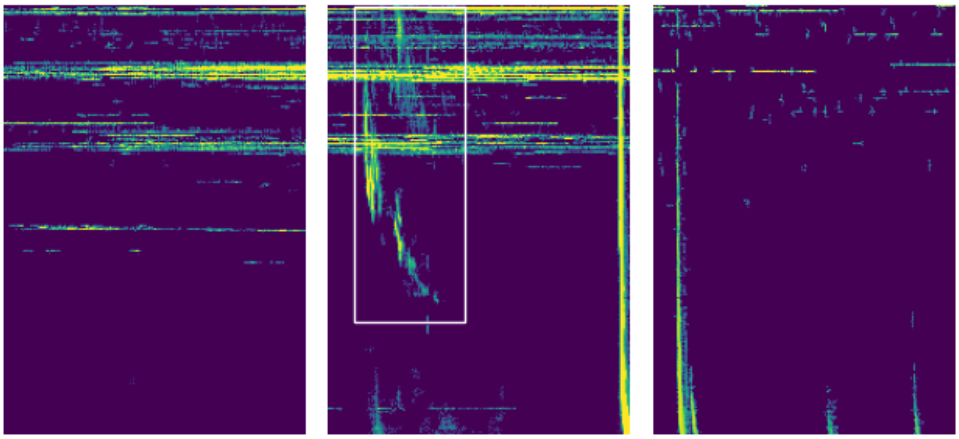


Figure 1.3: *Different types of SRBs that are used as image-based inputs to the object detector. (Left) example of background noise and interference with no burst present. (Middle) includes a Type II burst outlined with a white square. (Right) examples of Type III bursts. Due to the presence of Type III overlapping with Type II bursts, in addition to various bands of noise, detection of Type II bursts can sometimes be a difficult task.*

less, recent ML solutions attempt to automate this detection process [32, 2, 34, 35]. However, detecting Type II bursts is more challenging due to their shape and their frequent overlap with Type III bursts, and therefore efforts for automated classification of Type II bursts are met with limited success.

This case study has used an object-detection model to detect Type II bursts. Given that there is a small amount of annotated data, this study has relied on pre-trained weights, and prior expert knowledge to improve the robustness of the detection model. To integrate prior knowledge in this domain, feature specialisation of the Type II and background properties of the data was used. Further experiments were conducted using adversarial training and data-specific augmentation, making use of the prior knowledge present in data to define searchable regions with which to construct adversarial examples.

1.3 Summary of Thesis Objective

The work of this thesis aims to develop strategies to integrate prior knowledge into the design DNNs. For this, we must introduce the general principles behind how prior knowledge can form inductive biases and help the DNN to learn by constraining the learning process. These principles are organised into four chapters by how each of the methods are incorporating the knowledge into the design DNNs.

In addition to developing the strategies, we aim to give a comprehensive evaluation of how well these strategies perform in different scenarios such as different DNN architectures, and different types of input/output data. To achieve this aim, we have selected three case studies in very different scientific domains test the integration strategies. Using these case studies has enabled the fully realisation of the proposed principles and has demonstrated their usage in realistic scenarios for various types of data. These

case studies have provided insights into the performance and generalisability of each method and allowed the exploration of how each of these methods complements other methods..

For two of our case studies, namely the detection of online grooming, and estimation of quantum chemical energy, our objective involves the construction and use of two novel datasets to facilitate the training of DNN models. These datasets are designed and annotated with specific properties to allow the use of prior knowledge, an essential criterion for the success of the methods presented in this work. For example, our new OG dataset includes annotations of grooming strategies or processes by expert linguists, our various chemical systems provide examples of OoE systems at different scales of systems for DNNs to learn atomic properties at various sizes of chemical systems.

Our final objective is to demonstrate not just how each of these methods work in its isolation, but also their additive effects and interaction when incorporated into a single DNN. We aim to show that it is possible to make best use of prior knowledge, even if the source of knowledge is the same, by using a combination of techniques that adapt the design of the DNN in different ways.

1.4 Contributions

The contributions of this thesis are:

1. A set of methods to integrate prior knowledge strategies covering a wide range of different DNN architectural types.
    - Methods for specialising existing features within DNNs to enhance the usefulness of the representation for classification performance and visualisation.
    - Specialised information processing to provide distinct representations of specialised concepts in the domain being modelled.
    - Excitation and stimulation-based mechanisms for recurrent and transformer architectures to help DNNs focus on essential inputs.
    - A method for providing domain-meaningful data augmentations. This consists of two methods: 1) adversarial training; and 2) data-specific augmentations. Our method of adversarial training is made to be conscious of the data being perturbed with the addition of adaptive neighbourhoods to determine the maximum amount of perturbation that can be applied to each data point. This method firstly allows the adversarial generation algorithm to modulate its strength to the individual sample of its data that its perturbing. Secondly, it may enable existing adversarial generation techniques (such as FGSM and C&W) in situations where the potential adversarial cannot be inspected visually, such as with non-image data.
2. Concrete implementations of these prior integration strategies in three case studies. The experimental results of each of the strategies show how they may work

in practice while also demonstrating the potential benefits and limitations of the approach.

- Three methods for modifying semantic relationships with word embeddings. We leverage the identification of preferred variants from CL analysis to propose a selective text normalisation by modifying word embeddings in support of the classification. We propose and compare three implementations.
- Three competing methods for providing specialised representations of chemical bond-types into GNN architectures. We leverage the existence of different types of node relations to modulate the information flow within the GNN. To this end, we formulate and compare the three strategies, namely specialised message production, specialised node update, and specialised update function. We provide different implementations of the specialised node update for GRU and dense layer-based update functions.
- A method for providing specialised representations of grooming communicative strategies into recurrent networks. We leverage the identification of key themes from CL analysis to exploit their relationship to text class for improved classification.
- A method for stimulating recurrent networks to provide attention to essential grooming strategies into the input conversations. Identifying critical themes from CL analysis is used to exploit their relationship to text class for improved classification.

3. A thorough evaluation of prior knowledge integration strategies both individually and jointly. They have been applied to the base models (DNNs with no prior integration strategies) of each case study to demonstrate their flexibility.

4. Novel datasets for training DNNs:

   - Chat logs of online grooming.
   - Three new datasets for training quantum chemical systems to learn from OoE molecular/crystalline systems.

5. State-of-the-art models for:

   - Interpretable OG detection.
   - Quantum chemical energy predictions of unstable systems.
   - Solar burst detection

1.5 Organisation of Thesis

The organisation of this thesis is as follows: firstly, in Chapter 2, we give a general background into DL. Secondly, in Chapter 3, we discuss the existing literature surrounding the use and integration of prior knowledge into ML models is evaluated. This literature review is not limited to only DNNs (as is the focus of this work), but instead, explores a variety of solutions that help make the wider space of ML more

trustworthy. Thirdly, the datasets and base models behind each case study are presented in Chapter 4. Fourthly, the proposed prior integration strategies are described in Chapters 5 through 8. For clarity, the different strategies have been separated into their respective chapter that consists of: (1) a general principle behind the method; (2) specific implementation styles for the case studies; and finally (3) experimental results given the selected case studies. This presentation style is intended to keep the focus on one strategy at any one point to aid the reader. In Chapter 9, many strategies are combined into a single DNN for each case study, and the performance of the DNNs were evaluated on the desired task against comparison models. Finally, our concluding remarks and recommendations are given in Chapter 10.

Chapter 2

Preliminaries

Given the scope of research and case studies presented in this work, many different concepts are introduced. To aid readers unfamiliar with these concepts, this chapter describes, in general, the application of these concepts to this thesis. Further and more specific background on topics is given at the beginning at each method chapter.

We begin with the definition of core concepts in DL. Most importantly, for DL to succeed, one needs a dataset with which to learn from. In general, a dataset is a collection of input samples, accompanied by output labels (in the context of supervised learning methods). From these samples, a DL model can learn the input-output relationships that, if the data is representative of real-world processes, can generalise to other unseen input samples.

Data A set of inputs $\mathcal{X}$ contains many observations or instances $\mathcal{X} = \{x_1, x_2, ..., x_n\}$, where each instance is composed of a set of features $x_i = \{x_1^{(i)}, x_2^{(i)}, ..., x_m^{(i)}\}$. $\mathcal{X}$ is said to contain n instances, of m dimensions with the total size $n \times m$. As is often the focus of this thesis, for the scheme of supervised learning, an additional set $\mathcal{Y}$ is the known (true) result of applying some unknown function f over an instance $x \in \mathcal{X}$ such that $f(x_i) = y_i$. The domain of data is a set of tuples from input and output sets, $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$.

Given a set of data, a learning algorithm can be used to learn to recognise patterns that occur within the data and produce a predictive model. Our learning algorithm of choice for our work is the DNN, an architectural type of ML model that has enjoyed recent success in a variety of tasks from image recognition to natural language understanding.

DNN The goal of DL is to find an approximate function $\hat{f}$ from a set of possible functions $\mathcal{F}$ that minimises an error inherent to the problem (cross entropy for classification, mean squared error for regression for example), such $\forall x_i \in \mathcal{X}, \hat{f}(x_i) \approx f(x_i)$

where the prediction made by the DNN is approximately close to the ground-truth as it may be infeasible to find the true function f that created the ground-truth. $\hat{f}$ is considered to be the DNN that takes x_i as an input, returns $\hat{y}$ as an output, with h_k the hidden representation in the *k-th* layer of the DNN.

As DNNs are architectural types of ML models, one can vary the architecture itself and the modality in which the architecture operates. The result is a different DNN that follows varying *styles* of architecture, and indeed several styles are used in this thesis. We give here a brief description of each of the different DNN architectures and in which type of learning task they may be useful. However, it is not within the scope of this thesis to comprehensively explain each of the models in great detail. Readers should refer to resources such as [36] to gain a detailed understanding of the traditional architectures and [37] for Transformer architectures.

Fully-connected networks are the most basic form of DNN where each neuron performs a simple weighted sum of its inputs, applies a non-linear activation function, and passes the output to all neurons in the proceeding layer. Activation functions allow DNNs to learn non-linear relationships and ultimately increase the complexity it can learn. Non-linear activation functions are applied to the linear output of each neuron. One typical example of an activation function is the Sigmoid activation,

$$\sigma(x) = \frac{1}{(1 + e^{-x})}$$

where σ the Sigmoid activation function and x is the output of the linear part of the neuron. One useful property of the Sigmoid function is: $\forall x \in \mathbb{R}; \sigma(x) \in [0, 1]$ and therefore is often used in a binary classification task where $\sigma(x) < 0.5$ denotes the negative class, else a positive class. Other examples include the Rectified Linear Unit (ReLU),

$$\text{ReLU}(x) = \max(x, 0)$$

in which we have two linear parts of this non-linear function. Due to its simplicity, ReLU is computationally efficient while often out-performing other, more complex, activation functions and therefore has become the initial choice for non-linear activation of hidden layers in a DNN. Another activation function that maybe used in the hidden layer is the hyperbolic tangent (tanh) functions

$$\phi(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

where the output of the function is always within the range of $[-1, 1]$.

For situations where there are more than two possible classes, a Softmax activation function may be used:

$$\text{softmax}(\mathbf{x})_i = \frac{e^{x_i}}{\sum_j^K e^{x_j}} \text{ for } i = 1, ..., K$$

where K is the number of class outputs. This function is used to normalise the distribution of outputs such that for the output of the Softmax function we have $\sum_i \text{softmax}(\mathbf{x})_i = 1$, providing a distribution where we can interpret the values as pseudo-likelihoods of the input being labelled as class i and the most likely class is $\arg\max_i \ \text{softmax}(\mathbf{x})_i$.

Unless specified, we use the ReLU activation function for hidden layers of the DNN, with the Sigmoid function in the output neuron of binary classification tasks, and a Softmax activation function for multi-class tasks.

Convolutional networks convolve over the input using a kernel, and are often used to process images due to their inherent ability to capture spatial relationships in the data. Convolutional networks are typically composed of three types of layers: (1) convolutional layers where a kernel of learnt weights convolve over the input to extract a feature representation; (2) pooling layers to down-sample the feature space thereby reducing computational size of the network; and (3) fully-connected layers to project the feature representation into a separable space for classification.

Recurrent networks, as opposed to capturing spatial relationships like Convolutional networks, instead aim to capture temporal relationships by processing input in steps. For example, by processing a sentence one word at a time. It is able to capture temporal dependencies between words by using the activation of one input in a cell's 'memory' state to be used in later inputs.

Transformer networks, while able to learn dependencies between inputs like Recurrent networks, do so by processing all the inputs at once and computing the attention between each input and every other input. Despite its original design for natural language understanding tasks, Transformer architectures have been used for many data types including audio and images.

A DNN, regardless of architecture, is trained to approximate a true function f by optimising its internal representation or weights and biases, given its accuracy on previous predictions. To measure its accuracy, or rather, its in-accuracy of the DNN's predictions, a cost function is used.

Cost Function The process of 'learning' consists of updating a DNN's weights and biases to produce increasingly accurate outputs. In order to know how to update a DNN's internal weights and biases to become better at predicting, a measure for determining its current performance is used. A cost (or loss) function uses the DNN's predictions $\hat{y}$ and the ground-truth output y to produce a *loss* L for the DNN–how badly a DNN is currently performing. The internal weights and biases of the DNN will be updated in a way that reduces the loss, often with gradient descent. The learning objective is therefore the minimisation of loss for a training set. A common cost function for regression tasks is the mean-squared error (MSE) $L = \frac{1}{n}\sum_i^n (\hat{y}_i - y_i)^2$ that measures the average distance between predictions and ground-truth values in the dataset.

This measurement of distance between the predictions and the ground-truth labelling is the same as the topological sense of distance. In the introduction of metric space, one has a set of points $\mathcal{X}$ in the metric space and a distance metric $g(x, y)$ to provide a meaning to the expression that point x lies near point y [38]. The simplest distance metric, as in Euclidean Space, is the L_2 or Euclidean distance. In this way, the MSE cost function is the average euclidean distance between predicted and ground-truth labels. Unless specified, the distance metric is assumed to be L_2. Though other distance metrics are available, such as L_1, and L_∞

As the cost function allows the DNN to update the internal weights by measuring how incorrect it was with its prediction, it is encoding and learning information from the data. To help this process of learning, in this thesis, we often give examples of how it is helpful to provide further *hints* to the network to become more accurate with its predictions. Originating from the terminology of [39], hints are the introduction indirect *'help'* to the DNN on how to solve a certain problem. For example, a hint on how to improve on a classification task can be introduced via auxiliary loss terms.

Definition 2.0.1. Hints are the method of indirectly *helping* a DNN or providing more information (such as with auxiliary loss functions) to improve it's performance on a desired task.

To learn an accurate approximate function, DNNs are typically created through three phases: training, validation, and testing.

Training The DNN is given an observation x_i and creates a prediction $\hat{y_i}$. This prediction may be a scalar value such as the case with a regression prediction; a probability distribution denoting a class assignment; or a scalar probability value that the input conforms to a positive class. Based on its prediction $\hat{y_i}$ and the true value y_i, the DNN will update its weight based on an cost function. Updating the internal weights of the DNN will allow it to iteratively predict values that are closer to the correct value. Training often consists of a number of epochs where the dataset it shown to the DNN multiple times. A single epoch is one complete iteration of a dataset.

Validation During training, it is possible to *overfit*. This concept represents the state where the DNN performs well on the training data, but badly with unseen data–it is not generalising. Often, this is due to the DNN simply 'remembering' the seen input, and not learning any useful representations that would allow it to predict well on unseen data. Detection of overfitting is done through the second phase, validation. This phase consist of checking the DNN against a very small sample (usually 10\some predictions for this validation data, but will not be able to update it's weights. This then give some interpretation of how well the DNN is generalising during training.

Testing After the DNN has been trained and before the DNN is deployed, the DNN will be finally tested on data that it has never seen before, the test set. This test set will give another representation of how well the DNN generalises, but it does not suffer from data leakage - the modification of hyper-parameters and DNN architectures in order to perform well on the validation data. The DNN's performance on the real-world data is its generalisation performance. How well it is able to learn a useful representation from the training data that can generalise to the real-world.

Despite the accuracy DNN's can achieve through this process of learning, existing work (e.g. [6]), show that DNNs can be susceptible to so-called *adversarial examples*. These are examples of input that, although being often indistinguishable from the original input, result a very different output value causing a misclassification. The prototypical example of an adversarial example where the modification of a few (selective) pixels of an image of a pandas results in the DNN outputting a classification of a gibbon. Though the example is not particularly concerning in this context, adversarial examples have been known to occur in reality due to natural occlusion or noise [9], that can affect self-driving vehicles from detecting stop-signs [40].

Definition 2.0.2. Given f, the classification implemented by a DNN, and some small perturbation ε of an input $\mathbf{x}$, an adversarial $\mathbf{x}^\star$ is ε-close to $\mathbf{x}$ with $f(\mathbf{x}^\star) \neq f(\mathbf{x})$, while $\mathbf{x}^\star$ belongs to the same class as $\mathbf{x}$.

To measure how susceptible to adversarial examples DNNs are, a *robustness* measure can be introduced. One common measurement of robustness is created by the average distance between original and adversarial examples. However, many more sophisticated, and indeed informative measures of robustness can be used. (e.g. [41])

When creating an adversarial example, modifications are made to the original input, but are usually bounded by some maximum distance, i.e. some maximum amount of perturbation can be applied to the input to create the adversarial example. If we use a bounded range of perturbation, we are specifying a neighbourhood of potential inputs surrounding the original input to which an adversarial example could be created from.

Neighbourhoods capture some information as to the set of reachable points from some point x given a maximum distance of travel away from x. We often use *neighbourhood* in this thesis to refer to the set of points constructed via perturbation to an original data point.

Definition 2.0.3. A neighbourhood to point x is the set of adherent points inside an n-ball B defined by a radius $r > 0$. The open set or neighbourhood to x is $\{\bar{x} | \bar{x} + \epsilon < B(x, r)\}$.

Chapter 3

Literature Review

Integrating prior knowledge into DNNs is not the only way to increase the trustworthiness of a system. There are many novel research methods one may use to create trustworthy ML systems. Notable examples are: improving the interoperability and explainability of DNNs [42, 43, 44]; and the formal verification of specific DNN properties concerning the problem task [45, 46]. However, it would be untenable to research all methods within a single thesis where each method is given adequate consideration for its design. Thus, for this work, we concentrate on the distillation of expert knowledge into the design and computation of DNNs. Through this focused approach, we hope to create suitable solutions of how one may create trustworthy ML systems for various other research disciplines.

In this chapter, we discuss the existing literature surrounding methods to integrate domain knowledge into ML. Though the purpose of this thesis is to focus on strategies for DL, this review of literature is not limited to only DNNs. Due to the breadth of research discussed here, we will reintroduce the most relevant works during the methodology chapters.

The structure of this chapter follows the approaches we have set out: Feature Specialisation, Specialised Information Processing, Attention on data, and Adversarial Examples. Nevertheless, it also includes other potential methods from the wider community, such as logic-based methods.

3.1 Feature Selection & Extraction

Many traditional ML methods require more data processing than DNNs. With these traditional methods and some insight into the data, practitioners can use feature engineering to improve the performance of their chosen algorithm. Feature engineering

may include feature selection where features are specifically chosen for their perceived importance in the classification, or feature extraction where new features are generated from others to create more useful ones [47]. Indeed, feature extraction can reduce the complexity of the dataset and improve the representation of the data, making it easier to learn from them.

One way of transforming the data is through automatic feature extraction. These can take the form of dimensionality reduction using: principal component analysis (PCA) [48], non-linear reduction with manifold learning [49], t-SNE [50], Laplacian eigenmap [51], locally-linear embeddings [52], or spectral methods [53]. These methods algorithmically select a subset of features that best describes the dataset, and become especially important when the data has a large number of dimensions that the ML algorithm cannot appropriately learn from.

However, if the dataset is small enough in terms of dimensionality, an alternative is transforming the original dataset through a manual process of identifying and selecting essential features. Similar to our case study of using CL analysis, this method has been used to select relevant features for the detection of grooming in chat rooms. This method is based on the assumption that a person's societal cultural context can determine which words they will use. For example, a computer scientist may use words such as *matrix* or *dimensions* more than a psychologist. Therefore it may be possible to use this particular usage of words as discriminative for the classifier. For example, Hidalgo and Díaz [54] combine both linguistic and chat-based features to improve the effectiveness of detecting OG in chat-rooms. Linguistic features can take the form of the traditional *term frequency-inverse document frequency (tf-idf)* [55] statistics as well as existing text analysis software to perform *linguistic inquiry and word count (LICW)* [56], which provides categorical relationships between the usage of words and the personality background of those who use them. Chat-based features are low-level features that consist of the average length of messages in the chat room, the time the messages were sent, the time between messages. They found that high-level linguistic features improved over the baseline methods. The best combination of features for the classifier was both linguistic features (tf/idf) and chat-based features. These results may contradict the assumption that the usage of particular words can implicate a person's context, but it perhaps is a reflection of the classification method (supper vector machine and logistic regression) not being sophisticated enough to take advantage of the high-level information these features provide. Therefore, one can manually select essential features as long as the model is capable to taking advantage of the information that is provided to it. The work in this current thesis supports this suggestion.

Instead of manually deciding upon the features, Ahmed et al. [57] use correlation-based feature selection (CFS) and minimum redundancy maximum relevance (MRMR) methods to select features from a set of magnetic features to predict solar flares. Both CFS and MRMR select a subset of features that are correlated with a class (solar flare or not solar flare), in addition to being un-correlated with other features. Through these methods, the authors are able to remove features that are redundant or irrelevant. With the removal of these features the classifier produced more precise predictions while also not having a substantial impact on the number of false-positives.

Even though the aforementioned approaches of feature engineering can benefit traditional ML techniques, there are interesting examples where feature engineering can benefit DNNs also. Best et al. [58] apply temporal transformations to audio spectrums, in addition to the mixup method of Thulasidasan et al. [59] to classify the vocalisations of orca whales using a CNN. Zhang et al. [60] use tf/idf and clustering to extract relevant logs to predict future IT system failures with an LSTM network. van den Oord et al. [61] apply a fourier transformation to music to recommend relevant songs with a CNN. In these approaches, feature transformations are used to make features more relevant, bringing forward the useful information from the raw data, to make classification easier for the DNN.

3.2 Feature Specialisation

3.2.1 Indirect Specialisation

By using indirect specialisation to augment the latent feature representation, knowledge of domain constraints and quantities can improve the trustworthiness of the DNNs by encouraging the model to learn from the prior expert knowledge to perform well at the intended task.

When training a DNN, there are two opposing methods for training:

- **Single-Task Learning** – Single-Task Learning (STL) is a more typical example of how DNNs are used to learn on a single task, such as image classification or regression.
- **Multi-Task Learning** – Multi-Task Learning (MTL) is the strategy of sharing internal DNN parameters between many auxiliary tasks as an approach to help the model generalise better than STL alone [62]. For a full review of MTL see [63].

There are two approaches for how to incorporate MTL in the training process of DNNs, which differ by how the model's parameters are shared between the auxiliary tasks:

- **Hard-parameter sharing** – In hard-parameter sharing, there are generally a set of hidden layers shared between all tasks. In order to produce the output for each of the auxiliary tasks, there are some task-specific output layers. Examples of using hard-parameter sharing in MTL tasks are: [64, 65].
- **Soft-parameter sharing** – Soft-parameter sharing differs from hard-parameter sharing in that each task has its own set of layers (and therefore its own parameters), and the distance between the representations are regularised using some distance metric, such as the euclidean distance. Usages of soft-parameter are demonstrated by Duong et al. [66], where the authors create a cross-lingual text-parser using a DNN with soft-parameter sharing. The parameters between tasks are regularised using the L_2 distance metric.

Hard-parameter sharing is the most common approach for MTL [62], and can be applicable for a most applications. Sener and Koltun [64] demonstrate its applicability to a variety of DL problems from digit classification to scene classification, and DNN architectures. Yang and Hospedales [67] use hard-parameter sharing to perform sequence tagging across different languages simultaneously.

Many standard practices for indirect specialisation, particularly in MTL, use common representations via hard- or soft-parameter sharing. This technique has been successfully demonstrated in [64, 65, 66]. However, intelligently selecting shared representations has also proposed. For example, Sun et al. [68] demonstrates an algorithms that learns to select the best DNN layers for sharing parameters between tasks. This algorithm, when applied to a DNN, can produce a model that outperforms other models using hard-parameter and soft-parameter. Intelligently selecting shared representations is also explored in Standley et al. [69], where three techniques (optimal solution, early-stopping approximation, and higher-order approximation) are introduced to determine which auxiliary tasks should share which shared representations. Although it is able to find the best parameter sharing, they find optimal solution to be computationally costly during training time. These technique of Standley et al. [69] may be more applicable to different forms of DNNs other than Sun et al. [68], as it does not require learning connections between representations in the architecture but rather just the output of the DNN.

Application of indirect specialisation can be accomplished via auxiliary tasks. These tasks are related to the main task for the DNN, but produce secondary outputs. From these secondary outputs, domain knowledge may be integrated into the DNN. Furthermore, these auxiliary tasks may assist with sparse and low-quality data. Auxiliary tasks may help improve the generalisation of the DNN and provide potentially more informative predictions with multiple outputs.

Using auxiliary tasks is a common method to estimate proprieties in physical or chemical problems. For example, Raissi et al. [70] combined a DNN module that estimates mass and momentum, and this new module constrains the solutions by implementing partial differential equations of fluid dynamics using automatic differentiation and estimating the equations' parameters. Schütt et al. [71] used a similar approach to improve SchNet's predictions of both energies and their derivatives w.r.t. atom positions into interaction forces. Yang et al. [72] use the same approach into a generative adversarial networks (GAN) framework to solve stochastic differential equations. These approaches have shown auxiliary tasks is a useful method learning.

Gülçehre and Bengio [39] integrate prior knowledge using auxiliary predictions into the hidden layer of simple Multi-layer Perceptrons (MLPs) to predict whether three Tetris sprites in a 64x64 pixel image are of the same shape. The MLP learns to predict auxiliary features such as the presence and location of the sprites. These auxiliary predictions are made from the intermediate layer of the MLP from the experimental observations that DNNs are more easily trained when they are given hints about the representation the intermediate layers should be modelling [73, 74, 75].

Indirect specialisation has also been tested as a method to encode logical formulas. Muralidhar et al. [76] penalise a DNN's output violating logical rules w.r.t. the input

features. Hu et al. [77] use the posterior regularisation framework of Ganchev et al. [78] to encode domain constraints for generative models. A teacher-student architecture in Hu et al. [79] incorporates first-order logic rules used by the teacher network to create an additional loss for the student network. Overall, losses efficiently integrate priors in several application domains, and the additional loss approach is explored in this thesis.

Overall, indirect specialisation can be a practical approach to promote trust in DNNs:

- Encoding multiple properties can reduce over-fitting on a single task.
- It may focus attention on informative inputs for predictions [62].
- Multiple output predictions resulting from auxiliary tasks may provide more information for the human to make informed decisions.

3.2.2 Direct Specialisation

While indirect specialisation implicitly encodes the prior knowledge into the DNN through the use of auxiliary tasks, direct specialisation makes direct modification to the representation following the domain knowledge. Sometimes the information encoded structures in these hidden representations is known to the human, and in these situations, it is possible to make modifications to these structures that enable the adaption to prior domain knowledge.

Word embeddings are one such DNN representation a human observer can understand. In word embeddings, each word is represented by a coordinate in an N-dimensional space. The structure of word embeddings is trained to exhibit semantic and lexical relationships [80], where the distance between embeddings is trained to be roughly equal to the semantic information of these words. For example, words that mean similar things should have embeddings that are positioned closer together within the representation, e.g. doctor and nurse may be positioned closer together, while doctor and farmer will be further apart.

Gender bias has been a critical topic for the direct specialisation of word embeddings [81, 82, 83, 84]. As many large corpora for training language-learning DNNs contain biased data concerning the male/female gender, techniques have been put forth to mitigate the effect the bias will have on the usability of a DNN trained on the data. To combat this effect, Bolukbasi et al. [82] reduces the gender bias in word embeddings through direct modification of embedding coordinates. Given an embedding space, the singular dimension is statistically related to the concept of gender. For gendered words of the same usage (such as *he* or *she*), the words are moved closer together in the embedding space. Wang et al. [85] compute a subset of dimensions using PCA of the top 500 male- and female-biased words identified in GloVe. The un-biased embeddings are computed by reducing the distance of the orthogonal direction of the gendered dimension. This appearance of bias in word embeddings is further explored for contextualised embeddings by Zhao et al. [84]. The authors analyse the contextualised word embeddings of ELMo and find two dimensions corresponding to gender,

unlike the single dimension from the GloVe representation of Bolukbasi et al. [82]. Two approaches are used to produce non-biased embeddings: 1) data-augmentation by swapping words with the opposite gender during training, following the approach of Zhao et al. [83]; 2) generating gender-swapped versions of the input text to obtain embeddings of both original and gender-swapped texts and then average the embeddings. The first method, although more successful in reducing gender bias in contextualised embeddings, requires retraining ELMo. Gender bias here has been a pivotal topic for demonstrating how prior knowledge can used to adapt learnt representations. In this example, the knowledge that words of different genders should not affect how the word is perceived by the DNN. This is incorporated directly into the representation of word embeddings.

3.3 Specialised Information Processing

Specialised information processing, as opposed to sharing representations like indirect specialisation, provide unique parameters for different concepts in the domain. These different parameters enable the learning of specialised concepts that may be helpful in improving performance on a task or improving trustworthiness.

Specialised information processing have been adapted to recurrent cells, specifically Long-short Term Memory (LSTM) cells. For instance, Lu et al. [86] combine the MTL paradigm of Feature Specialisation with a specialised information processing method to create a new LSTM cell. This specialised processing channel learns task-specific information to perform natural language tasks, such as named-entity recognition and part-of-speech tagging. The LSTM-cell proposed by Lu et al. [86], Shared-Cell LSTM (SC-LSTM), similar to our design of Specialised information processing discussed in Chapter 6, carries specialised representations for individual tasks, and performs the final task. The specialised representations are concatenated with general representation. A different method by Kim et al. [87] shares a similar design philosophy where a single general LSTM is used to capture information from many inputs, and single specialised LSTMs capture information from a single input. The representations from these general and specialised LSTM cells are additively combined to produce the final output.

These methods of Lu et al. [86] and Kim et al. [87] differ by how the specialised information processing feed back into the generalised representation for the final prediction. While both of these proposed methods use sub-networks to capture task-specific information, in Lu et al. [86], the sub-network uses: (1) the previous specialised cell state; and (2) the current cell state of the generalised cell to create a new specialised cell state. The output of this timestep is the concatenation of both specialised cell state and hidden state. In Kim et al. [87], the sub-networks are less connected during each timestep, and the final output is only the addition of all cells.

Instead of using sub-networks to capture task-specific information, Liu et al. [88] operate under the principle that many layers of stacked LSTMs each contain within itself different information for a specific task. However, in normal stacked LSTMs, the preceding layers require information from the previous layers. It, therefore, becomes

harder to use each of the layers of information independently. In their work, in an attempt to leverage the task-specific information available in each layer of the stacked LSTM, they fully connect the input of each LSTM layer with all of the preceding layers, enabling pruning of some of the subsequent layers without dramatically changing the information. This idea of task-specific representations at different layers is shared by Søgaard and Goldberg [89], who apply multi-task learning to only the lower-levels of DNNs, allowing higher-level layers to form higher semantic representations, and also Alonso and Plank [90] where different auxiliary tasks are added to different layers of a stacked-RNN.

The design of specialised information processing are not limited to recurrent networks. Other methods have been designed to incorporate specialisation communications within graph structures. For example, Schlichtkrull et al. [91] propose relational graph convolutions to handle large multi-relational graph structures. For this, unique parameters are associated with each of the different types of relations, thereby providing a specific processing channel per concept in the domain. Instead of using a single DNN for the entire graph like Schlichtkrull et al. [91], Ying et al. [92] compose the graph into hierarchical layers, and at each layer, a unique GNN is used to produce graph embeddings for the downstream task. Wang et al. [93] use specialised information processing to combine the features from different levels of attention (e.g. the attention over a single node or the attention between multiple nodes).

From these works, specialised information processing may be appropriately integrated into the design of DNNs to account for domain knowledge. In doing so, each of the concepts modelled by the specialised information processing methods may help in improving the performance of the DNN by feeding back into the generalised representation. It is also common for specialised information processing to be combined with indirect feature specialisation [86, 89, 87, 94, 90]. Combining these approaches provide hints to DNN on how each of the domain concepts should be represented, thereby helping the learning process of the DNN. This strategy is compared within this thesis.

3.4 Attention on Data

From Section 3.1, we highlight that some features are more critical or informative for the classification or regression task. When training a DNN, we would like to attend to these critical features (both input and latent representations) automatically through the supervised learning process. However, in situations where the feature importance is already known, the attention of DNNs to these features may be improved via supervised or unsupervised attentional approaches. In supervised attention, ground-truth labels are used to guide attention during training. Though sometimes called self-supervised attention, in unsupervised attention, no labels are used to train the attention mechanism, in principle, it is optimised via back-propagation with the downstream task of classification, detection, or regression. The distinction between supervised and unsupervised attention refers to how attention is optimised, i.e. with or without prior knowledge. Even if the DNN is trained via a supervised process, such as with language translation, the attention implemented in such DNN could still be unsupervised. Attention mechanisms have been adapted to various forms of

DNNs: Recurrent Neural Networks [95, 96, 97, 98, 99], Transformers [37, 100], and Convolutional Neural Networks [101, 102, 98, 99]

Unsupervised attention is one of the more popular techniques due to the recent Transformer architecture by Vaswani et al. [37], which has been a quickly emerging topic for improving the performance of many DNNs. Unsupervised attention was first introduced for language translation tasks by Bahdanau et al. [96], where the concept of optimising the hidden-state communication with attention between Encoder-Decoder architectures was introduced. This idea was further pursued for a number of applications including NLP [37, 103, 1, 104], image recognition [102, 105], image-to-image translation [106], speech recognition [107, 95], and reinforcement learning [108, 109]. Further improvements onto this design of unsupervised attention were developed by Luong et al. [1] in which two modes of attention via global optimisation (using all hidden-states) and local optimisation (considering only a subset) were presented, in addition to the Dot-Product attention that is further refined with a scaling factor to account for very small gradients. The result is the method Scaled Dot-Product Attention used as a key computation used in the Transformer architecture. These improvements have cemented the usage of attention as key component for many different topics in ML.

Saliency is a cornerstone method for incorporating attention mechanisms into the design of CNN architectures. Saliency describes the ability of an object or feature to stand out from its neighbours. Saliency estimation may help in attending to particular features within the scene [110]. Zhang et al. [101] use multiple levels of features (from different layers of the CNN) to create attention weights that can be used as saliency maps for an object detection task. These attention weights can be optimised using target saliency maps, providing ground truth annotation for supervised attention. In this work, the authors find that applying supervision to attention weights improves the performance of object detection. Wang et al. [111] use DNNs to estimate the saliency of objects using local and global information of the image. Into the global estimation DNN, the authors use known priors such as contrast and background information to improve saliency estimation performance.

Supervision may be used to guide the attention mechanism through the use of target labels. The training procedure can integrate priors without modifying the DNN's architecture. For example, Derakhshani et al. [112] use assisted excitation of CNN neurons in the images' areas of interest, thus providing localisation and semantic information to the DNN. An attention mechanism was used in a supervised way to focus a DNN on essential words in Nguyen and Nguyen [97]. Cornia et al. [99] use eye fixation of humans as the ground-truth saliency map. Saliency predictions are optimised in CNN+LSTM architectures using the ground truth annotations. For a language translation task, Mi et al. [113] add an alignment (quantification of attention the decoder is placing on each word) distance to the cost function during training. These works provide strong evidence that prior knowledge may be help improve attention onto important features.

Attention can be used to optimise the compression of large amounts of information. For tasks such as sentiment analysis with long sequences and image caption generation, compressing large amounts of information into a fixed-sized vector can be chal-

lenging, and some information will be lost in the process [114]. Luong et al. [1] addresses this with local attention that focuses only on a few words in the source language of a translation task. This local attention includes a smaller context window over which the attention algorithm is applied over. Indeed attention can provide hints to the DNN on what features to attend to when performing this compression process and possibly result in a set of features that are richer and more descriptive for their intended task.

In some cases, attention can be used to focus on essential features and improve the computational speed through this focus of attention, thus leading to a reduction of model parameters (see Mnih et al. [98]). Through the use of supervision, attention could be adapted to include prior domain knowledge following the expert's understanding of important features (see Chapter 7).

3.5 Generating Adversarial Examples

Despite the improvements in performance that DNNs bring for many ML applications, they are susceptible to so-called adversarial examples, where very small and specific changes to the input result in a different classification. These adversarial examples undermine the trustworthiness of DNNs as the adversarial example, from the perspective of the human, appears to be identical to the original input. In this section, we will highlight some of the key works that introduce methods that generate adversarial examples.

A DNNs susceptibility to adversarial examples was first shown by [6]. In this work they had demonstrated that, even for state-of-the-art image classifiers, by modifying the input with very small changes, i.e. perturbations, they were able to change the output classification of DNNs. To generate these perturbations they used a box-constrained optimisation method, that locates small-probability 'pockets' around the original image that produces the misclassification of a target class. Moreover, this work shows the susceptibility to adversarial examples is not simply an artefact of the training methods (as a model trained on a different subset of the data would produce the same miss-classification), but rather propose there is a deeper and not fully understood the meaning behind the effect. Though these adversarial examples undermine the trustworthiness of DNNs, creating methods to generate them is useful for improving the DNNs robustness to these forms of attacks.

One way to classify adversarial example generation methods is by how much model information they use in the generation process. These two types are black-box (using very little information) and white-box (where more model information is used).

White-box – White-box algorithms relies on direct information about the model architecture or how the image is processed. For example, standard white-box methods use the gradient of the DNN's loss function w.r.t. the input image to compute the adversarial example from this image. While this method can quickly and easily find adversarial examples, this kind of gradient information would not be available in real-world scenarios, i.e. deploying the DNN in an online web-application. Despite white-box

methods not necessarily being applicable for exploiting and attack DNNs in these real-world scenarios, they are still useful for improving the robustness of DNNs through the use of adversarial training. Many research methods have shown that despite the methods using targeted DNNs, exploiting the information of the image gradients, the computed adversarial examples are often transferable to other types of models, even those with very different architectures. Common types of white-box methods include: using gradient information, manifold sparsity, and selected pixel manipulation.

Black-box – Black-box methods may be more applicable in real-world scenarios as they use no information of the model they wish to attack. These methods can often perform multiple queries (i.e. repeated inputs with varying values) against the model they wish to create adversarial examples for. This process of querying can take much longer to find adversarial examples than the white-box methods. Common methods for black-box include: multiple querying, and surrogate models.

Another way to classify adversarial attacks is by how they target the resulting classification of the adversarial example. **Targeted attacks** are those methods that specify a misclassification label that should be assigned to the adversarial example. For example, given an image of a car, one may want to find the perturbation that results in the classification being a plane. While **un-targeted attacks** do not require a misclassification label of a specific type, in these cases, it is enough for any misclassification label. In the example of the image of the car, any different classification label apart from the original true label is desired.

Adversarial examples for image-classifiers are more common than object detectors as the problem task is more straightforward. For example, given an input image, a single classification label is output from the DNN. Thus to make an adversarial example for this DNN, a perturbation would need to apply to the input that results in a different classification label.

3.5.1 Creating Adversarial Examples for Image Classifiers

For image classifiers, many methods have been proposed. While Szegedy et al. [6] proposed to use the adversarial examples as part of the training set, the optimisation process was computationally expensive. Later work by Goodfellow et al. [3] creates a faster method for creating adversarial examples (Fast Gradient Sign Method, FSGM). FGSM adds the sign of the gradient w.r.t. the cost function to the original image. Doing so is purposefully pushing the pixel value of the image in the direction that increases the loss of the prediction and thus creating a misclassification. This approach is computationally quick enough such that the method can be included during training time to increase the model's robustness against these forms of attacks.

Whereas in FGSM, imperceptible changes are made to the entire image, a different method by Papernot et al. [115] makes a prominent and noticeable change to a small region, while the rest of the image remains untouched. This method works by computing the Jacobian matrix of the input image to find the most significant or salient pixels of the image in which to perturb. While showing their method can outperform many other adversarial attacks in various scenarios, its reliance on computing the Jacobian,

a potentially slow computation, restricts its usability in many applications.

Many of these previous methods perform a single computation step to modify the pixel values, though other iterative methods have been proposed. One effective iterative method comes from Carlini and Wagner [116] where they introduce the C&W attack. Their work improves upon the objective function of Szegedy et al. [6]. Moreover, to improve the optimisation of the method, in this work, they proposed many alternatives objective function candidates as well as distance metrics. The C&W method can often outperform many of the existing approaches to more easily fool DNNs.

Despite many methods being proposed to easily fool DNNs, very few quantify and compute the robustness w.r.t. adversarial examples. The *DeepFool* algorithm proposed by Moosavi-Dezfooli et al. [117] aims to compute adversarial examples and compare the robustness of various classifiers to particular types of perturbations. DeepFool iteratively perturbs the original image in the direction of the closest hyperplane and thus creates a misclassification with a minimal Euclidean distance to the original image in a potentially quicker time than Goodfellow et al. [3].

Instead of using the gradients to manipulate the image like [3], Huang et al. [118] applies small perturbations to the output of activation functions in the DNNs. By slightly manipulating the activation in each layer of the DNN, they test for the existence of an adversarial using an Satisfiability Modulo Theory (SMT) solver to check whether the perturbed activations would result in a misclassification. If these perturbations create an adversarial example, the activations are propagated backwards towards the input layer, where the image that results in the misclassification is generated. One of the discretisation methods to find adversarial examples possible–the Lipshitz continuity assumption, has been further explored for sigmoid and hyperbolic tangent nonlinear functions in Ruan et al. [119]. They use this continuity to evaluate the reachability of the network with the aim of both proving properties of the output ranges in addition to generating adversarial examples.

The benefit from using knowledge of the model, i.e. white-box methods, whether it is through activation analysis or gradients, is that adversarial examples are created quickly and with a minimal number of perturbations, enabling the knowledge of adversarial existence to be used within the training process and therefore creating a more robust model that is less susceptible to adversarial attacks. However, it is often the case that these white-box methods generate adversarial examples that may only be applicable for a small selection of DNN architectures. Black-box methods, on the other hand, may find adversarial examples that are more applicable to a broader rand of models.

One type of black-box formalisation is shown in Zheng et al. [120], where adversarial examples are constructed using realistic constructs such as JPEG compression and other image artefacts by adding Gaussian noise to all pixels. Guo et al. [121] show that given randomly picked dimensions in the input space, an ϵ can be either added or subtracted from the dimension in a way that reduces the probability of the target class. In order to find adversarial examples using this method, multiple queries must be made against the model before one can be found. However, given the simplicity of modification made, this random method minimises the number of queries posed to the black-box model and yet achieves a 100% success rate of attacking the model in most

cases.

By not using direct model information, the many approaches instead focus on salient features in the input to generate adversarial examples more quickly and with fewer queries. Wicker et al. [122] use the Scale Invariant Feature Transform (SIFT) method to detect salient features from within images. Pixels within these features are manipulated by sampling from a Gaussian Mixture Model. They show through the SIFT algorithm that realistic camera movements such as rotation can create adversarial examples, creating concerns for the use of any automated driving systems. Other techniques for detecting salient pixels has been explored by [123], where pixel-wise decomposition to detect necessary pixels for classification is used. Decomposition here involves two competing methods: 1) Taylor-type decomposition, where Taylor expansion approximates the prediction and finds the classification boundaries. 2) layer-wise relevance back-propagation where the relevance of each pixel can be computed at each layer and accumulated at the input. By modifying these particularly important pixels, they can flip the classification made by the DNNs with a high degree of confidence. These methods, though not being always successful due the limitations on the model information that can be exploited, do demonstrate that DNNs can be attacked with adversarial examples in realistic scenarios such as when they are deployed as part of a web-application, thus further highlighting the need for the improved robustness to perturbation.

While many of these existing methods aim to generate adversarial examples for the means of demonstrating their methods efficiency in doing so, while also highlighting DNNs susceptibility, research has also been conducted to quantify the robustness of DNNs. Using the adversarial attack method of Huang et al. [45], Duncan et al. [124] evaluates quantised networks' robustness. Quantisation decreases the floating-point precision of DNNs from the normal 32-bit to 16-bit or lower. The authors find that when transferring DNN weights from 32-bit to quantised forms, the robustness of the DNN is often preserved if not enhanced. This observation may coincide with the notion that half-precision DNNs generalise better due to reduced over-fitting [125]. The role of quantisation on the measure of robustness is further explored by Gorsline et al. [126], who vary the *strength* of adversarial attacks and the amount to which quantisation is applied. In this work, the authors find a point at which further quantisation does not affect the defence of adversarial attacks. However, this work considers only MLPs, leading to possible future work to consider alternate architectures to determine if this trend continues.

3.5.2 Creating Adversarial Examples for Object Detectors

Since [6] has shown DNNs are susceptible to adversarial attacks, much research has provided both exploitation of this weakness and defences. However, these methods are often developed solely for image classifiers, where a single image is given a single classification label. Little work has considered the case of object detectors and image segmentation models, where individual objects or pixels are classified within a scene. These detectors are generally more challenging to create adversarial examples due to the number of bounding boxes at various scales that could be considered for object

detection. A typical object detector could create thousands of potential regions for classification, in contrast to the single classification made by traditional image classifiers.

However, recent research has proposed methods to enable the generation of adversarial examples for object detectors, and therefore also allow the adversarial examples to be used as part of adversarial training – potentially boosting the models' robustness to these types of adversarial attacks.

These methods, much like the methods created for image classifiers, rely on bounded variations of input space. More specifically, the perturbations made to regions of the image are bounded by a single shared value. Our work on generating neighbourhoods (Chapter 8) aims to improve and augment these adversarial attack methods by making the bounded variations more specific and unique for each sample of the dataset. This method consider the density (or conversely the sparsity) of the input space and the estimated class boundaries that may be close to these samples. The complete method can be found in Chapter 8.

Adversarial examples for image classifiers can only create misclassifications, while object detectors are prone to two types of errors as a result of perturbation:

Misclassification – As object-detectors predict a classification label for each of the bounding boxes estimated to contain an object, adversarial examples can be created through the misclassification of these bounding boxes. For example, by taking an object of type dog in a scene, the object detector can place a bounding box over the dog but label it as a horse. In this situation, while it has correctly predicted that this region of the image contains an object, it is incorrect as to what object the region contains.

Mislocalisation – Mislocalisation differs from misclassification in that it does not matter what the final classification of the object is. In this type of attack, the object-detector would predict bounding boxes for superfluous objects or objects that do not exist in the scene.

Many of these types of attacks for object detectors can be categorised by how they intend to create the adversarial examples. These categories are *Disappearance attack* and *Creation attack* [9]. The disappearance attack aims to fool the object detector into missing or not recognising the ground-truth object that exists within the image, while the creation attack performs the opposite: this attack fools the detector to classify and mislocalise objects that do not appear in the image.

Disappearance attack – In disappearance attacks, the input to the detector is manipulated in a way such that the ground-truth is not detected by the model. This type of attack is often used for image-classifiers, where the image is manipulated to give the incorrect class label. To achieve this type of adversarial example for object detectors, however, pixels of the object to make disappear can be manipulated [127, 9, 40], or more salient targets within the scene can be created [128].

Creation attack – The creation attack's aim is to fool the object detector to detect non-existent objects within the scene. Examples of these types of attacks can often take

the form of 'patch' type manipulations, where a small subset of the image undergoes a large amount of perturbation while the rest of the image remains untouched. These patches could be physical stickers placed in the scene [9] or applied directly to the image before detection with the DNN [129]. While patches have been used to create mislocalisations, less strict complete image manipulations can achieve the same attack [130, 131].

A popular trend in demonstration the applicability of adversarial examples for object detectors is causing a United States-based *stop sign* to disappear [40, 9, 132, 127, 128]. This case is the epitome of a safety concern should a fully autonomous vehicle (using an object detector as its main source of checking environmental hazards) fail to spot a stop sign and continue driving without stopping safely. These signs are made to disappear by manipulating the red background of the sign [40, 9, 132, 127], or by causing the object detector to focus on a more salient part of the image [128]. Often we see these adversarial examples tested at different viewpoints: indoors/outdoors, at different distances, and different angles; before being applied to viewpoints of vehicles travelling down the road next to the sign [132, 40].

Many methods for generating adversarial examples have been proposed for the simple problem of image classification but far less for object detection. Attacking an object detector is usually more complex than an image classifier, as the detector model will produce multiple bounding boxes at varying scales. Xie et al. [131] propose Dense Adversarial Generation (DAG), an optimisation procedure that suppresses the confidence in the true correct class while also increasing the targeted incorrect class. DAG applies back-propagation from the pixels in the image for image segmentation or to the proposals. However, the authors find that, for object detection, it is necessary to increase the number of proposals made by the Region Proposal Network (RPN) in order to make proposals denser and thus have more probability of being close to the input proposals to the rest of the network.

DAG can often require many iterations to generate an adversarial example. To address this, Wang et al. [130] propose using Project Gradient Descent (PGD) to attack an object detection model, Faster R-CNN. Their method consists of using the total combined (summed) loss of the two-stage detector, which can more quickly find adversarial examples more often than the current state-of-the-art Dense Adversary Generation (DAG) method. While their method proposes using the summed loss of the network to create the examples most optimally, they show the contribution of the individual loss terms towards the success rate of fooling the detector. They find the classification loss terms are generally more susceptible than the bounding-box regression loss terms, and the Fast R-CNN loss terms more susceptible than the RPN loss terms.

As object detectors aim to detect and locate objects within a scene, some methods have been proposed to perform real-world manipulations to the scene. Chen et al. [40] propose the adaptation of the *change-of-variable* attack and *Expectation-over-Transformation. In this work, the red colouring of a stop sign was perturbed in a way that makes DNNs fail to detect the sign's presence. For instance, Song et al. [9] improve an existing /Robust Physical Perturbations* (RP_2) technique of creating physical adversarial examples for image classifiers for object detectors. More physical variation, such as object position and rotation, is added to the distribution of perturbations that can

Table 3.1: *Summary of Adversarial example generation methods for object detectors.*

		Model Information		Attack Type	
Citation	Year	Black-box	White-box	Disappearance	Creation
Lu et al. [8]	2017	x	x		
Xie et al. [131]	2017	x	x	x	x
Song et al. [9]	2018	x	x	x	x
Zhao et al. [127]	2018	x	x	x	
Chen et al. [40]	2019		x	x	
Huang et al. [128]	2019		x	x	x
Lee and Kolter [129]	2019		x	x	x
Wei et al. [133]	2019		x	x	x
Wang et al. [130]	2020	x	x	x	x
Liao et al. [134]	2020	x	x	x	

be made. An additional smoothness constraint is added to reduce the pixelated perturbations to improve the success of misclassification at various distances. Lu et al. [8] propose to create adversarial examples that are invariant to viewing conditions. This method, tested on a video scene of a stop sign, maps a base texture (e.g. red of the stop sign) to each from of the scene. Through this mapping, this method may perturb certain pixels of the stop sign that may be invariant to the viewing condition of stop sign at any one frame in the video.

In Table 3.1, each of the categories to which each method uses model information, and which attack type they perform are shown. For all cases, white-box methods are used with the majority of attacks being of disappearance. For instance, using model information to make a stop sign to disappear. Though other methods do use surrogate models to test the transferability of the generated adversarial examples to other object detectors.

In certain cases, in particular when the generation method is computationally quick enough, these adversarial examples can be included during training to improve the DNN's robustness to certain types of perturbations. This method of training, adversarial training, can increase the DNNs robustness to perturbation by including the loss w.r.t. the perturbed input in the final loss of the network, thereby enforcing the resistance against this type of perturbation while also providing a small variation on the training samples seen during training.

The existence of adversarial examples demonstrates why it can be difficult to trust decisions made by DNNs. The presented methods may improve the trustworthiness by either using fast and efficient generation methods to apply adversarial training and thus improve the DNNs robustness against possible future adversarial attacks, or aims to systematically quantify the DNN's robustness and therefore understand when a model may be more susceptible than others in certain situations.

3.6 Knowledge Representation & Logic-based Methods

It can be difficult for many of the existing DNN architectures to perform inductive reasoning and exploit existing domain knowledge encoded in symbolic forms [135]. The field of Knowledge Representation is concerned with how information can be represented to allow computer systems, and indeed ML models, to reason and make decisions given the existing knowledge base. These methods combine formal reasoning with probabilistic systems. At a high level, some of these methods consist of:

- Representation of knowledge as graphs, such as ontologies, in a way that classical DNNs can use the information.
- Fuzzy Logic to relax the domain constraints and learn the relations between concepts.
- Using Markov Logic Networks, where nodes are atoms of a formula, and edges are connectives.
- Using Logic Tensor Networks to convert the real logic formula into computational graphs using a combination of real features (vectors and matrices) and fuzzy semantics to model connectives.
- Verification of DNNs using Linear Programming or SMTs.
- Automated reasoning and theorem proving.

3.6.1 Knowledge Graphs

Knowledge graphs are representations of knowledge bases where edges interlink entities. These edges may exhibit different properties about how entities are linked depending on the usage of the knowledge graph. For instance, MUTAG [136], one such knowledge graph dataset, contains chemical compounds as entities with edges representing bonds between atoms. [91], using MUTAG and other knowledge graphs, predict edges between entities to enhance entity classification performance.

One such method of embedding knowledge within a graph data structure is with an ontology. Ontologies are the formalisation of terms or a system in a vocabulary. They can provide a mechanism to use automated reasoning over a knowledge base to derive information about the system that is not explicitly defined, for example, through axioms. Several ontologies exist within the biomedical domain. However, combining these different knowledge representations can prove difficult due to the inconsistency in the expression of classes, instances, and relations within the ontology. Moreover, contradictions within a single ontology can occur due to the use of ambiguous relations. These various factors can limit the ability to use these ontologies and perform high-level queries for knowledge extraction.

Biological data and knowledge bases are being increasingly reliant on Semantic Web Technologies and other graph representations. However, feature learning methods

within the biomedical domain are not widely applied to use these formats. To address this, Alshahrani et al. [137] propose a method to learn ontological object proprieties. This is done by inferring a graph representation of ontologies, creating graph embeddings, and creating a multiple logistic regression classifier (one for each object property) that predicts an edge between the vertices. Three ontological databases are considered: Gene Ontology (GO), the Human Phenotype Ontology, and the Disease Ontology. From these, a knowledge graph (a graph-based representation of entities in the world and how they interact with each other) is constructed. Finally, the knowledge graph is iterated over using the Deepwalk Perozzi et al. [138] algorithm to create an embedding representation of the connected edges and vertices, used by multiple logistic regression models (one for each object property in the ontology). Sufficed to say, while they use logistic regression models, it would be possible to develop to DNNs to operate over ontologies by using these embedding representations that is seen in applications such as with DeepGO[1], Kulmanov et al. [139], where a hierarchical DNN is used to encode the transitivity of the sub-class relations in the ontology. The structure of the ontology may be encoded in the DNN in different ways. While Alshahrani et al. [137] create graph embeddings, Kulmanov et al. [140] generate a geometric representation of GO using description logics. This method works by creating differentiable loss functions that mimic the set of *EL* description logic rules.

3.6.2 First Order & Fuzzy Logic

Various methods have been proposed for using logic (first-order logic, fuzzy logic) in addition to ontology knowledge bases for the construction of interpretable and accurate ML classifiers. Prior domain knowledge and the high-dimensional ML learning processes may be combined in a way that provides a benefit to the trustworthiness of the system, in addition to the overall accuracy of the final model.

Logic may be integrated into DNNs via indirect specialisation. Diligenti et al. [141] propose two strategies of integrating domain knowledge into DNNs. The first investigates the use of indirect specialisation as a means of regularisation. The second method uses a first-order logic knowledge base with fuzzy logic rules. This latter fuzzy logic representation of the knowledge base is used to compute an additional loss which is added to the training loss of the DNN. Experimental results show significant performance improvements when combining CNNs (AlexNet in particular) with these Fuzzy Logic rules for a partially labelled dataset. However, if the entire dataset is labelled, combining a CNN with logic rules only has a marginal improvement. While Diligenti et al. [141] use additional losses, more sophisticated methods may use teacher-student architectures. [79] show a method to distil prior knowledge from logic-rules (teacher network) into the weights of a DNN (student network). Here the distance between weights of these two networks are optimised to be reduced. These methods highlight the approach of using indirect specialisation as a form of knowledge distillation without fundamental changes to the architecture of DNNs.

Other methods have proposed to use DNNs to model directly the logical forms using Logic Tensor Networks (LTN) [142]. LTNs provide a framework for incorporating

[1] `https://github.com/bio-ontology-research-group/deepgoplus`

symbolic rules using fuzzy logic. In LTNs, each concept or predicate is represented as distinct points within the feature space, and the model is trained to optimise the representation of each function or predicate to satisfy known formulas. Using the concept of LTN provides an interesting method of including reasoning in DNNs. Bianchi and Hitzler [143] evaluate the deductive reasoning capabilities of these types of models by showing the model only partial knowledge, such as only true predicates and only some axioms. They show that using very small datasets, LTNs can produce decisions that can be manually inspected for where the LTN fails to produce the correct answer. A simple implementation of logic rules has been used by Krüger et al. [144] to model large latent spaces using computational state-space models. Each state space contains a probability of a predicate being true, and models are trained through an extension on the KL-divergence–Jensen-Shannon distance. They show that using partial predicates, they are not only able to increase the accuracy of the model predicting the activity, but 'spikes' in predicate probabilities increase the interpretability of the model's decision process. This method of applying the activation of simple predicate could also apply to other models that operate using states such as RNNs. As opposed to [142] that adapts the architecture of the DNN to encode logical formulas, the work of [145] encodes the logical specifications via loss functions that allows them to train DNNs to meet these logical specifications. While this work demonstrates how, like [140], logical constraints can be incorporated into the training regime with loss functions, it goes further to incorporate a declarative language for querying the DNNs, such as searching for an adversarial example. The query language is then translated into a loss to be minimised with LBFGS-B to locate a solution to the query.

Markov Logic Networks (MLN) are one such alternative to LTNs, whereas a graph, Markov Networks consist of atomic formulas as vertices and edges between atoms are first-order logical connectives. For example, Khot et al. [146] uses MLNs for question answering of extracted knowledge from a US-based 4th-grade textbook. Their method outperforms other solutions, including satisfiability with an SMT solver. However, this only results in a 55% success rate. They question whether the constriction of the acyclic nature of the graph in the MLN limits the inference that can be made in this task. Perhaps including another model into the sub-task of inference would help the MLN to perform better. Kok and Domingos [147] aims to solve the problem of learning long formulas through *structural motifs*. Learning using structural motifs assumes that re-occurring patterns are underlying the data to reduce the number of computations. However, while MLNs can combine probabilistic models with first-order formulas, initial methods are limited with the number of objects represented in the formulas due to the exponentially sized resulting MLN where inference becomes intractable.

These logic-based methods provide some interesting applications to incorporate domain knowledge through various methods. These range from those methods that implicitly encode the domain constraints with indirect specialisation, to methods that recreate the logic rules in a DNN architecture with differentiable formulas. While some are limited in success when compared with traditional DL, they do provide insight into how symbolic and probabilistic models may be combined in the future to create trustworthy hybrid-DNNs.

3.6.3 Verification of Deep Learning

Trustworthiness of DNNs may also be improved using formal verification techniques. After a DNN has been trained, formal verification can test that, under certain conditions, properties of the DNN hold. For example, using the adversarial attacks as motivation, we may want to ensure that all perturbations within a small range result in an output classification that are consistent with the class label of the original data. By using verification techniques, such as SMT solving or optimisation using Linear Programming, these properties can be tested, thereby making DNNs suitable for safety-critical systems as properties under certain conditions of the DNN can be specified and formally verified for certification.

One method to verify properties of DNNs is with SMT solvers. However, an SMT solver expects logical (and most often Boolean) formulas as input. Therefore, to enable the use of SMT solvers, an alternate logical representation of the DNN must be made. One issue in creating this representation is the non-linear activation functions present in DNNs. These non-linear functions cannot be easily translated into logical formulas, thus different approaches have been proposed. For instance, Pulina and Tacchella [148] introduce the use of abstraction of non-linear activation functions using piece-wise approximation. Ehlers [149] also uses piece-wise approximations of non-linear activation functions, but use an approximation of the SMT search process to improve the speed and scale to which DNNs can be verified. Kokke et al. [150] demonstrate a framework for verifying DNNs using SMT. Their framework, implemented in Haskell and $F^\star$, can create an encoding of a DNN by making using of linear approximation for the non-linear functions. More accurate approximations can be made with more components of the linear approximation, but it also increases the complexity of the formula to test for satisfiability.

Other methods use optimisation over a constrained formulaic representation of the DNN. Katz et al. [151] extend the Simplex method to enable the use of ReLU activation functions. This extension involves splitting each ReLU activation into two forward- and backward facing nodes in a Simplex formula. These two parts van then be satisfied independently before ensuring the lower bound of the forward facing node is equal to the maximum of the input or 0, thus replicating ReLU. Instead of Simplex, Dutta et al. [152] propose the use of a local optimisation with gradient optimisation and a global optimisation using Linear Programming. This method when compared against Katz et al. [151], can prove properties of much larger feed-forward DNNs (127 layers, with 6845 neurons per layer) more quickly.

While various methods for the formal verification of DNNs have been proposed, many of these methods do not scale well to large networks, certainly not to the scale of DNN that would be used in a system such as fully-autonomous vehicles, for example. Therefore, for this field of verification of DNNs to progress, research into allowing the verification technique to scale to large logical formulas, or abstracting DNNs in way such the output logical formula is small but still accurate, is needed. Nevertheless, verification within small DNNs, verification can give provable guarantees as to the computation of the DNN under varying conditions and therefore does improve the trustworthiness of DNNs.

3.6.4 Combining Machine Learning with Interactive Theorem Proving

In the preceding examples, the focus of the approach is ML: how does one integrate logic into the ML model? There are, however, other interesting approaches where logic is the focus and aided by the use of ML. One such example is adding an ML model to assist in the interactive theorem proving process.

Applications of the combination of ML and theorem proving can be divided into Symbolic via inductive logic programming (ILP) [153, 154] or abductive logic programming (ALP) [155], numeric with DNNs or kernels, or a hybrid approach that combines both symbolic and numerical methods [156, 157, 158, 159, 160]. Methods have been devised to combine the pattern recognition power of ML with theorem proving to either automate the proof search or aid the user in solving the proof with potential tactics to use.

Theorem proving with automated systems (automated theorem proving or ATP) and have been steadily improving in computational performance [156], while interactive theorem proof (ITP), where interactive input from the user to deduce the series of steps to prove a formula has added richer programming environments with expanded language implementations of type systems. For instance, Komendantskaya et al. [161] integrate Weka[2] into Proof General[3] to provide statistically prompted proof guidance through data-mind proof heuristics. Clustering algorithms are used to group related proof strategies and help the user to decide how to proceed with the proof. In this work, a feature extraction method is demonstrated (named the *proof trace method*) enabling the abstraction from the implementation of the ITP language. This allows the ML algorithm of choice to find statistical patters where it might otherwise have difficult and enable the use of the method for ITP languages outside of its intended use. Proof trace method works on the basis of recording properties of proof tactics to form a matrix representation, where tactics types of the tactics arguments). Using this representation for training a DNN may be complemented from our method outlined in Chapter 7, where depending on the different context of the current goal, attention of different parts of the proof trace could fluctuate as per the annotation of the expert logician training the DNN. While Komendantskaya et al. [161] 's representation provide a description of each tactic with multiple properties, a more recent attempt Gauthier et al. [158] to combine ITP and ML use a single descriptor for each tactic for their clustering algorithm. Heras and Komendantskaya [164] test the capabilities of the feature extraction method of Komendantskaya et al. [161] using different clustering algorithms (such as k-means, FarthestFirst, and Expectation maximisation). Another method that combines clustering methods from ML and ITP is presented in Heras et al. [160]. In this work, they suggest one of the limitations of Komendantskaya et al. [161] that the suggestions by ML are not fully explainable are rows, and properties of the tactics are columns (such as the type of tactic and or apparent to the user why they have been chosen as suggestions. Therefore, Heras et al. [160] use a lemma generation approach which ML aids to reduce the number of possible suggestions. Fur-

[2]Weka (`https://www.cs.waikato.ac.nz/ml/weka/`) is an open-source toolbox of ML algorithms implemented in Java, supporting a variety of different algorithms from unsupervised clustering to DL [162].

[3]Proof general (`https://proofgeneral.github.io/`) provides a generalised interface for proof assistants written for Emacs [163].

thermore, these generated lemmas are customised to the current problem to aid the user in understanding why the suggestions occur.

Chapter 4

Datasets & Base Models

For much of DL research, the concern is improving optimisation procedures, designing new architectural paradigms, or improving existing tasks and score well on leaderboards. In these situations, standard datasets are often used to compare and contrast existing methods with the proposed state-of-the-art methods by the researchers. For the case studies used in this thesis, however, data is not always readily available. Therefore, to facilitate the training of DNNs, the work has necessitated the collection, and annotation of data. With exception of dataset used for the detection of Type II bursts (see Section 4.3.1), where upon the dataset of Type II bursts was originally introduced by [2], the OG detection and crystallography datasets were created for the production of this thesis.

This chapter explains each of these datasets designed to solve a specific problem in each scientific domain. Furthermore, each of these datasets has been carefully constructed and annotated by domain experts to aid the task of training DNNs. For each dataset, this chapter describes:

- The construction process of the datasets.
- Examples of the data (with additional examples highlighted in the appendices).
- Statistical properties of the data.
- Format and available variables in the dataset that may aid other researchers in these areas.
- The base models that were used for comparison when evaluating the prior integration strategies.
- The metrics by which the experiments were evaluated.

4.1 Energy Estimation of Quantum Chemical Systems

The first case study was the use of physics knowledge to improve the estimation capabilities of DNNs in non-stable chemical systems. For this, three novel datasets of both molecular and crystalline structures at various interatomic distances were generated. Secondly, the knowledge integration techniques were evaluated on two different architectural types, using two of the state-of-the-art models for comparison.

4.1.1 Augmented-QM9 & Crystal Datasets

Many previous ML approaches and associated datasets only consider molecules and crystals at their stable configuration, thereby learning the interaction of atoms only at equilibrium. QM9 [25, 26] is one such dataset that contains a diverse set of 134k molecules using carbon, hydrogen, oxygen, nitrogen, and fluorine atoms, but at their stable configuration only. The MD17 [24, 16, 165] and ISO17 [71, 16, 166] datasets provide potential energies of OoE molecules with (independent) perturbations of their atoms' positions. However, they are limited to 8 small organic molecules (MD17) and 129 isomers (i.e. molecules of the same size and atomic composition) of C7O2H10 (ISO17). To support our work, three new datasets were created consisting of OoE molecules that includes more diverse sizes (i.e. number of atoms) and compositions. These datasets are: 1) an augmented QM9 dataset of molecules; 2) a dataset of infinite crystals; and 3) a dataset of finite crystals at various system sizes.

An augmented QM9 dataset was created by taking the first 10,000 molecules of QM9 and modifying the interatomic distances at ten regular intervals between 90 and 150% of the original stable configuration. This dataset, therefore, contains 100K different systems. At each interval, the potential energy is calculated using DFT to calculate the potential energies to serve as the target energy for the supervised learning task.

Performing this compression/dilation of the interatomic distances creates an energy curve with the stable configuration found at the lowest energy (see Figure 4.1 for an example where the interatomic distances are compressed/dilated around the 100% stable positions). This exposes the DL models trained on such datasets varying compound structures and varying stabilities.

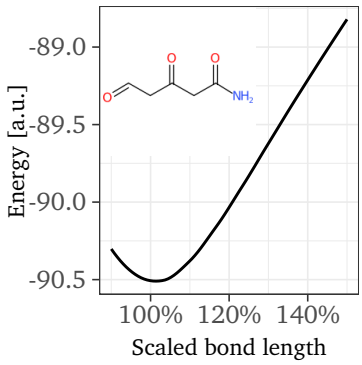


Figure 4.1: *Example energy curve where the stable configuration occurs where the energy value is at its lowest.*

The first crystal dataset, infinite crystals, has enabled the learning of regular bonding patterns that arise in periodic structures. Learning to estimate the potential energy for an infinitely sized crystal might benefit from the regular pattern in the lattice structure, possibly reducing the ML algorithm's complexity. Although not necessarily relevant from a physical point of view, a dataset of infinite-

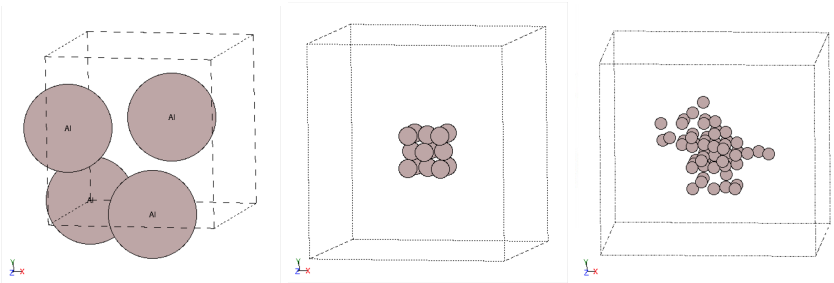


Figure 4.2: *Infinite Crystal (left) and Growing Crystal (centre: seed, right: growing) structures.*

sized crystals is an interesting case study for the design and training of ML methods.

To generate the infinite crystal, the simple Face-Centred Cubic (FCC) Bravais lattice was used as illustrated in Figure 4.2 (left) for both Al and Cu crystals following the unit cell pattern of Equation 4.1. From this base lattice structure, the interatomic distances were compressed/dilated isometrically (i.e. the same change in all spatial axes), and the potential energies were computed using DFT at 20,000 uniform intervals from 90-150% of the stable configuration.

$$[x_1, ..., x_4] = \begin{bmatrix} 0 & 0 & 0 \\ 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1/2 \\ 0 & 1/2 & 1/2 \end{bmatrix} \tag{4.1}$$

with atom x_i being at the x, y, z position from row i of the matrix w.r.t the stable lattice size.

Thirdly, growing crystals of increasing (finite) size and complexity enable experimenting with large scale atomic interactions in non-regular patterns. In contrast to the regular lattice pattern created by an infinite crystal, crystal growth allows for learning over more complex atomic interactions in non-regular systems. Starting from a basic fcc crystal seed of 14 atoms (Figure 4.2 centre), new systems were generated by iteratively placing atoms at a random location on the surface of the growing crystal following its lattice pattern (Figure 4.2 right), with sizes ranging from 15 to 114 atoms. Such a dataset enables evaluating an ML method's ability to learn how each atom contributes to the final potential energy. The DFT calculator presumes an infinite system, and thus for the finite crystals data, a 10Å vacuum was added to the system to prevent electronic interactions over duplicated images of the system.

Two subsets of this dataset were created: one that included the compression/dilation of the interatomic distances at ten regular intervals between 90 to 150% of the stable distance referred to as the Unstable Crystal Growth (UCG) subset, and another that contained only stable configurations referred to as the Stable Crystal Growth (SCG) subset. The latter aims to evaluate an ML model's capability to estimate energy values at varying system sizes (i.e. the number of atoms) without the added complexity of learning energy as a function of distance. Both subsets contain Al and Cu crystalline systems. 20 random seeds were used for each atom type for SCG, thus creating 40

Table 4.1: *Number of stable and OoE systems in our datasets*

	Augmented QM9	Infinite Crystal	Crystal Growth Stable	Crystal Growth Unstable
# stable	10k	2	4k	1k
Total #	100k	40k	4k	10k

Table 4.2: *Statistics on the Augmented QM9's molecules*

	Mean	STD	Min	Max
Number of Atoms (per molecule)	17.95	2.95	5	29
Number of Molecules (each composition)	269.54	652.82	10	4,690

varied crystal growths and 4,000 crystalline systems. For the *Unstable Crystal Growth* subset, 5 of these random seeds were selected for each atom type to consider 1,000 basic configurations, each being compressed/dilated for a total of 10,000 crystalline systems.

In all three new datasets, OoE systems were obtained through compressing/dilating all interatomic distances at regular intervals within 90-150% of stable configuration, thus generating isometric compressions/dilations of the whole system, which are referred to as *'scaling'* in the rest of the thesis. The ground-truth potential energy was calculated using CP2K[1]'s DFT at each spatial configuration. An example input to CP2K can be found in Appendix A.

The total numbers of stable and stable+OoE systems, for each dataset, are provided in Table 4.1. Statistics on the composition of the Augmented QM9's molecules are in Table 4.2 and Figure 4.3.

The properties available in the Augmented-QM9 dataset are shown in Table 4.4. In addition, for the continuous variables of this data, the summary statistics are provided in table 4.3. A concrete example of the data is shown in Appendix D.

4.1.2 Base Models

The physics knowledge integration were evaluated on a graph-DNN and CNN. The corresponding MPNN and SchNet are the current state-of-the-art for estimating potential energies. However, neither base models were used in the same conditions as in their original papers. For example, MPNN was only used on stable systems, and neither model was tested on crystals. Furthermore, in the experiments presented in this thesis, the DNNs were trained on perturbations of molecules/crystals of different sizes and compositions at once, and generalise to unseen sizes and compositions. In [71], SchNet was trained on perturbations of a single molecule (MD17) or of isomers (ISO17). The datasets created here has several compositions and the sizes are more complex, but the integration of physics knowledge allows preserving high accuracy.

[1] `https://www.cp2k.org/`

Table 4.3: *Summary statistics of continious variables in Augmented-QM9 dataset.*

Characteristic	Test set, N = 10,000	Training set, N = 80,000
simulated_energy		
Median (IQR)	-74 (-80, -68)	-74 (-80, -68)
Mean (SD)	-74 (10)	-74 (10)
Range	-124, -40	-130, -19
volume		
Median (IQR)	7,389 (6,532, 8,387)	7,394 (6,538, 8,383)
Mean (SD)	7,519 (1,268)	7,511 (1,233)
Range	4,163, 14,835	3,541, 13,426

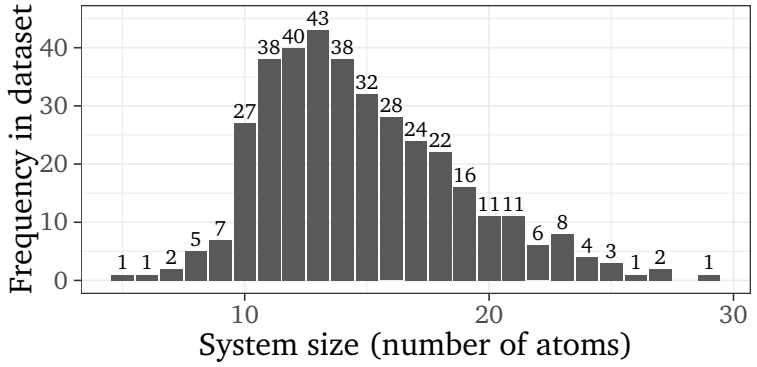


Figure 4.3: *Frequency of system sizes available in the Augmented QM9 dataset out of 10K molecules.*

Graph-DNN: MPNN learns to predict potential energy from an undirected graph $\mathcal{G}$, where atoms are nodes and edges e_{vw} describe the relation between two nodes v and w (i.e. their distance). MPNN applies three functions for 3 to 8 iterations: 1) the Message function (M_t in Equation 4.2, implemented with a perceptron) creates a *message packet* that symbolises the action of a neighbouring atom on node v. This function does not replicate the physics of atomic interactions, but processes information in a physics-inspired way. A final *message* vector m_v^t for node v combines the actions of all neighbours (Equation 4.2). 2) An Update function (U_t in Equation 4.3, implemented by a Gated Recurrent Unit (GRU)) updates the hidden state h_v^t of node v using its previous hidden state h_v^{t-1} and interaction message m_v^t. 3) A Readout function uses the set of hidden state of all nodes at all timesteps to estimate the system's potential energy through minimising a mean-squared error loss. See [167] for full details on the implementation of these functions.

$$m_v^{t+1} = \sum_{w \in N(v)} M_t \left(h_w^t, e_{vw} \right) \tag{4.2}$$

$$h_v^{t+1} = U_t \left(h_v^t, m_v^{t+1} \right) \tag{4.3}$$

CNN: SchNet implements three layers of interactions followed by atom-wise convolu-

Table 4.4: *Available properties in both the Augmented-QM9 and all Crystalline datasets.*

Characteristic	Data Type	Description
`category`	String	The molecule for which the data is calculated (Augmented-QM9 only)
`cell_lengths`	List[Float64]	Cell lengths in the xyz and rotational angles
`center_mass`	List[Float64]	Xyz position of the molecule's centre of mass
`chemical_formula`	String	Chemical formula for molecule
`chemical_symbols`	List[String]	List of chemical non-unique symbols
`smiles`	String	SMILES representation of molecule (Augmented-QM9 only)
`distances`	List[List[Float64]]	Pairwise matrix of distances from one atom to all others
`positions`	List[List[Float64]]	Xyz position of each atom
`simulated_energy`	Float64	Simulated potential energy using DFT
`input_a`	Float64	Push/Pull amount of molecule in the x direction
`input_b`	Float64	Push/Pull amount of molecule in the y direction
`input_c`	Float64	Push/Pull amount of molecule in the z direction
`pdf`	Float64	Calculated PDF of current positions from a Gaussian distribution centred at stable position.
`volume`	Float64	Volume of molecule with vacuum

tions, non-linearities, and a final sum pooling to obtain an energy estimate. Similar to Equation 4.2, each interaction layer l considers the sum of actions of all atoms on atom v (Equation 4.4). These actions are computed through element-wise multiplication of the neighbour's internal state with a radial filter W^l that depends on the distance d_{vw} between the two atoms. W^l is implemented by two softplus dense layers. This interaction 'message' is used to update atom v's internal state within a block of interaction and atom-wise convolutions (Equation 4.5). See [71] for full details on these layers' implementation.

$$m_v^{l+1} = \sum_w h_w^l \circ W^l \left(d_{vw}\right) \tag{4.4}$$

$$h_v^{l+1} = h_v^l + V_l \left(h_v^l, m_v^{l+1}\right) \tag{4.5}$$

4.1.3 Experimental Setup

All models were trained using an Nvidia Tesla V-100 with PyTorch 1.2.0, Python 3.7.2, and CUDA 10.1. Each model was trained for a maximum of 360 epochs, with 50 epochs of no validation loss improvement as the criteria for early stopping. The best model (determined by the validation loss) was saved to be used for testing during training.

The base models used the default hyper-parameters for both original and augmented versions. While, for the graph-DNNs, the number of iterations has been varied, we observed MPNN's default of three iterations worked best for all versions. A comprehensive list of the hyper-parameters used to train both MPNN and SchNet can be found in Appendix B.

The Augmented QM9 training set consists of all ten scaling operations for 80% of the molecules, so 8,000 molecules, totalling 80,000 training samples. Following the same rules, the testing and validation set contain 10% of the dataset, so 10,000 samples. As the Infinite Crystal dataset has two different crystal types (Cu and Al crystals), 8 data points were uniformly sampled from their respective range of scaling for training. The rest of the data were split evenly between validation and test. For the Crystal Growth dataset, a 60/20/20 split was used with the same procedure.

4.1.4 Evaluation Metrics

We report absolute error (AE) and relative error (RE $= \frac{|\hat{y}-y|}{|y|}$) between true y and predicted $\hat{y}$ energies in atomic units (a.u.). We also report the Distance to Stable Geometry (DSG). For molecules, it is computed as the absolute difference between the scaling where the predicted energy is minimal (optimised by multiple DNN queries for various scales) and the ground-truth at-equilibrium scaling (100%). We use the lattice size (expressed as the half diagonal of a face of the fcc lattice in Angström) instead of scaling.

Mean and standard deviation were computed over chemical systems, with std indicating the ability to handle a large variety of systems.

4.2 Online Grooming Detection

The second case study used CL knowledge with DL. To enable the design of appropriate strategies for this case study, the work relied on the annotation by experts on grooming language and linguistics. These strategies were experimentally tested using two base models to provide state-of-the-art baselines.

4.2.1 Online Grooming Chat Logs

To facilitate training of a DNN to distinguish from grooming and non-grooming in text-based chat logs, we introduced a new dataset referred to here as simply OG dataset. This dataset has built on the previous standard: PAN2012 [168], used in many studies for OG detection [169, 170, 54, 171, 172, 173]. PAN2012 was gathered from Omegle (one-to-one conversations with random users), IRC (technical discussions in groups), and the Perverted Justice (PJ) website[2] (chat logs from convicted groomers interacting with trained adult decoys, for a debate on the usability of this data, see [30, 174], with 396 groomers and 5700 / 216,121 OG / non-OG conversations. Some non-OG chat logs contain sexual wording, making the OG classification more challenging. Conversations are truncated to 150 messages each, which limits both CL and ML analyses. To resolve this limitation, the corpus was augmented with entire conversations (avg. 431 messages per conversation) and the addition of new groomers from PJ, totalling 623 groomers in 6204 OG conversations (same negatives).

This OG dataset has also included the results of a CL analysis of the new corpus using the method described in [29], which involved a heavy use of manual analysis by CL experts. As part of the data preparation for CL analysis, the groomer subset of the corpus had undergone word-standardisation, where spelling or intentional variations of words were changed to a standard form. For example, 'looool' was standardised to 'lol', and 'not comfy' was standardised to 'uncomfortable'. This process considers the context in which the variations on spellings (from this point referred to as *word variants*), and required the manual study by CL experts to determine whether a word had the same intended meaning as another, more standard word. This standardisation process had yielded a set of pairs of words with the same intended meaning.

While these variants were not specific to OG, but are valid for other scenarios of real-world chat conversations, the CL analysis then identified the variants that were most used by groomers. Word variants that were frequently used and highly dispersed (often used by many groomers) were selected to determine salient collocates. From these collocates a set of 2100 3-word collocates were annotated with a strategy employed by the groomer. These strategies, or OG processes, give an indication as to the type of entrapment tactic used by the groomer that may help in discriminating between grooming and non-grooming conversations. There are a total of seven different OG processes used in both base models. These were characterised into five main categories as seen in Table 4.5. For each of these categories, the number of annotated collocations and an example collocate (in italic text) in its context are reported. The summarised construction process of this dataset is presented in Figure 4.4.

Therefore, for evaluating the prior knowledge integration strategies with CL, two components of CL knowledge was used: (1) a set of word variants where each item in the pair has the same sentiment; (2) a set of 3-word collocates in context, and annotated by one of the seven OG processes. These CL components were used to test our prior knowledge strategies, but the DNN does not require these annotations at its testing phase.

[2] `http://perverted-justice.com`. As of 2019, the organisers behind the Perverted-Justice website have ceased further collection of newer chat logs from convicted groomers.

Table 4.5: *Processes used by groomers in order to establish a connection with a child.*

OG Process		# Coll.	Collocate Usage in Context
Approach: Reference to the groomer's intention to meet with the child.		622	"...lots more peaceful lol i know rightand i **could come over** right?"
Compliance Testing: Checking likelihood of victim agreeing to proposed behaviour.		23	"do u **like** talking to **older guys**?"
Deceptive Trust Development: Building trust with the victim with the ulterior motive of eliciting sexual activities.	Activities	61	"ok so any **plans** for **this weekend**?"
	Personal Information	33	"so can you **tell me how** me about how far it is from you to allendale"
	Relationship	357	"i couldn't **stop thinking about** u"
Isolation: Groomer distance the victim physically/emotionally from their support circle.		112	"**we meet** some **where** alone near your neighborhood..."
Sexual Gratification: Groomer's attempt to involve their victim in sexual talk/activities.		892	"just you and me touching each other ... **feeling each other**"

Table 4.6: *Breakdown of the conversations, sessions, and messages between groomer and non-groomer users.*

Stats Name	OG / Non-OG	Total
# Conversations	623 / 316500	317,123
# Sessions	6204 / 216242	222,446
# Messages	648463 / 1197875	1,846,338
# Words	27388 / 134075	161,463

This OG corpus contains chat logs from 317,123 users (623 groomer and 316,500 non-groomers). A user may be engaged into one or several *conversations* (chat logs) with other individuals, and each conversation is made up of *messages* sent during one or several *sessions*. The statistics of conversations, sessions and messages are summarised in Tables 4.6 and 4.7. The splits between train and testing sets follows the same design as PAN2012 to give a more realistic interpretation that grooming conversations are far less frequent than non-grooming conversations.

The properties available in the compiled OG chat log corpus are shown in Table 4.8. In addition to the chat log corpus, there is the complementary collocate dataset, the properties of which are shown in Table 4.9. An example conversation is shown in Appendix E.

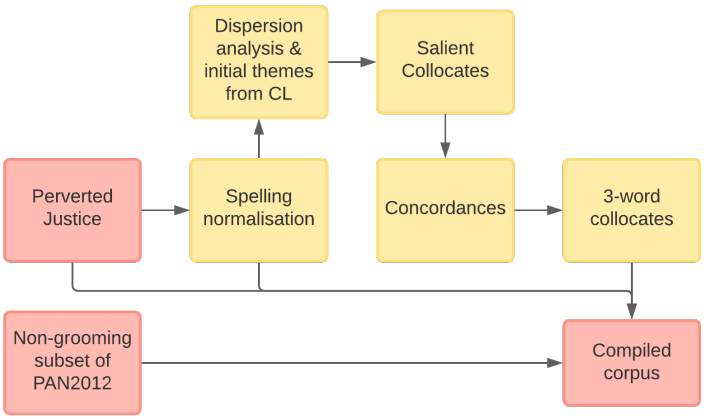


Figure 4.4: *Overview process of constructing OG chatlog corpus. CL analysis is required to generate a series of word variant pairs and 3-word collocates. The output of this analysis is combined with the original conversations from PJ and the non-grooming subset of PAN2012.*

Table 4.7: *Message statistics for the groomer subset of the corpus.*

Stats Name	Min / Max	Mean (STD)
# Mess./Groomer	2 / 28389	2,163 (3,224)
# Mess./OG Conv.	13 / 49207	4,228 (6,032)
# Mess./OG Session	2 / 32351	431 (1,414)

4.2.2 Base Models

The general applicability of the CL integration strategies has been demonstrated by applying them to two DNN architecture types representative of the two NLP standards of recurrent and transformer models. The recurrent DNN of [175] is the current state-of-the-art for OG. It is comprised of a language model that builds a Word Semantic Representation and an OG classifier with:

1. A language model: a word embedding layer and two Long-Short Term Memory (LSTM) that model word semantic and extract sentence vectors as the last hidden state of the LSTM.

2. An OG classifier: two LSTMs and a linear projection with Binary cross-entropy (BCE) loss.

In many forms of ML, an embedding is the mapping from one concept into a coordinate space. This is especially useful in language learning tasks where one can learn a the mapping of words to a unique $n-$ dimensional vector in the coordinate space, where the distance between vectors is roughly equivalent with the semantic similarity. However, distances in embedding spaces do not always carry this same notion, thus we make our definition more precise with WSR. We are stating that such embedding is encoding the semantic representation of words.

Table 4.8: *Available properties in OG Chatlog dataset*

Characteristic	Data Type	Description
`conversation_id`	String	Unique identifier for each conversation
`author`	String	Name of the user that authored the message
`is_groomer`	Bool	Truthy value indicating if current message is from a groomer
`line`	Integer	Line number in the conversation (i.e. each conversation begins at line 1)
`origin_file`	String	Where the conversation was extracted from PAN2012 or CL analysis
`text`	List[String]	Tokenised text contained within message
`time`	String	Time of message if it is known
`contains_groomer`	Bool	Indicates if a groomer is present at any point within conversation
`subset`	Categorical	Specifies if conversation should be entered into training or testing set
`grooming_themes`	List[List[Float64]]	PDF value for each of the 7 grooming themes. The value is the result of a gaussian centred on a detected grooming process.

Table 4.9: *Data properties of OG collocates*

Characteristic	Data Type	Description
`3word_collocate`	Categorical	Group for which collocate is part of
`Text ID`	String	Origin conversation ID
`Context before`	String	Message before the occurence of collocate
`Query item`	String	Search word to determine collocate
`Context after`	String	Message after the occurence of collocate
`theme`	Categorical	Category of one of the 7 grooming themes
`borderline`	Bool	Annotation confidence in usage of theme

Definition 4.2.1. A Word Semantic Representation (WSR) is an embedding space that captures the semantic relationship between words through the distance between vectors in this embedding space.

To test the prior knowledge integration strategies for Online Grooming Detection (OGD), two different base models were used based on these two sub-components. These two models are Online Grooming Detector - Recurrent (OGD-R) and Online Grooming Detector - Transformer Embedding (OGD-T).

Our OGD-R is a modified version (Figure 4.5 left) with the WSR of the language model's word embedding layer provided as input to the OG classifier in place of a sentence embedding. It may be replaced by similar WSRs. This is demonstrated in the results by also using the pre-trained GloVe WSR [176]. Further, to compensate for the

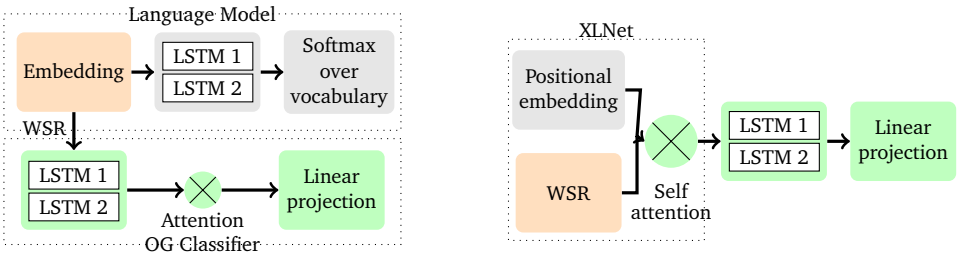


Figure 4.5: *OGD-R (left) & OGD-T (right), for integration of CL priors. Orange and green indicate where word variants and OG processes priors may be integrated, respectively.*

loss of sentence structure modelling previously provided by the sentence embedding and to account for the longer sequences of inputs into the classifier, an unsupervised attention mechanism was added [1] into the classifier.

The XLNet [100] transformer model is the state-of-the-art for NLP. It iteratively refines word embeddings, from a WSR similar to that of [175] to richer word representations to account for their relationships, to be classified by linear projection, with cross-entropy loss. This projection fails to handle the class imbalance of the OG corpus, and thus a two-layer LSTM was added to form OGD-T (Figure 4.5 right).

Both original and modified OGD-R were trained from random weights on the OG dataset. Experiments with the GloVe WSR used pre-trained weights from Common Crawl 840B [176]. The XLNet part of OGD-T was pre-trained on BookCorpus and English Wikipedia (see [100] for more information on the construction and training of XLNet).

Input to the models – The text analysis was performed on whole conversations, with conversation messages separated by the [SEP] token, so that inter-text representations can be modelled. Messages from both users were included with no distinction. The final OG/non-OG classification was obtained for the whole conversation rather than per message. All base and CL-augmented DNNs take raw text as input only. The only text preparation before the DNN was the tokenisation of named entities, as actual names are irrelevant to OG detection. Text normalisation was not applied such as with [177], since the methodological premise of this thesis is the design of prior knowledge integration strategies.

4.2.3 Experiment Environment

All DNNs were trained using an Nvidia 2080Ti with Python 3.7.5 and PyTorch 1.7.1 on a system running CentOS 7. These DNNs used best estimates for the hyper-parameters, though different values may improve the results. Where appropriate, specific hyper-parameters relating to the prior knowledge integration strategy is discussed in the experiment sections of this thesis. A comprehensive list of the hyper-parameters used to train both base models can be found in Appendix C.

The corpus was divided into 30% of users for training, 70% for testing, and 30% of training for validation, using a similar ratio as [168]. This division was based on

users ensures that the model may not recognise the specific language of a groomer but focuses on trends in OG language. Each DNN was trained continuously while tracking the validation loss at the end of each epoch. If the validation loss did not improve within a set number of epochs (OGD-R: 50 epochs, OGD-T: 100 epochs), the training was halted, the best model weights restored, and the test set predictions were made.

4.2.4 Evaluation Metrics

OG classification is evaluated by: Area Under Precision-Recall (AUPR) to account for the class label imbalance, F_1 score using the default 0.5 threshold on Sigmoid output, and the $F_{0.5}$ score used in [168] to weight the precision metric higher. The effects of selective normalisation are further measured by their proportion of distance reduction between selected variants

$$\Delta\mathcal{D} = \frac{1}{N}\sum_{k}\frac{|\mathcal{D}(v_k^i, v_k^j)^{new} - \mathcal{D}(v_k^i, v_k^j)^{old}|}{\mathcal{D}(v_k^i, v_k^j)^{old}} \tag{4.6}$$

and resulting average distance

$$\overline{\mathcal{D}} = \frac{1}{N}\sum_{k}\mathcal{D}(v_k^i, v_k^j). \tag{4.7}$$

4.3 Detection of Type-II Solar Bursts

The objective of the third case study is the automated detection of Type II solar bursts. For this, the existing WAVES dataset was used, along with an object-detection models to predict bounding boxes over the spectrograms.

4.3.1 WAVES Dataset

The WAVES dataset (originally created by [2]) was collected from instruments onboard the Wind spacecraft located between Earth and the Sun in order to measure properties of the solar wind in situ. These instruments have been collecting data on radio activity since 1994 at different frequency ranges. This data, collected by [2] from NASA's public archive, has been catalogued into samples (background, Type II, Type III), and each of the types of solar bursts present in these samples has been annotated with a class label, as a well as a pixel-wise mask over the occurrence of the burst. For the full description of the construction of this data set and the pre-processing that has been applied to the samples, readers should consult the original source of the data [2].

The events in which different bursts occur are presented in a spectrogram format with time along the x-axis and frequency (MHz) along the y-axis (Figure 4.6). For the construction of WAVES, the spectrogram was cut into positive and negative samples

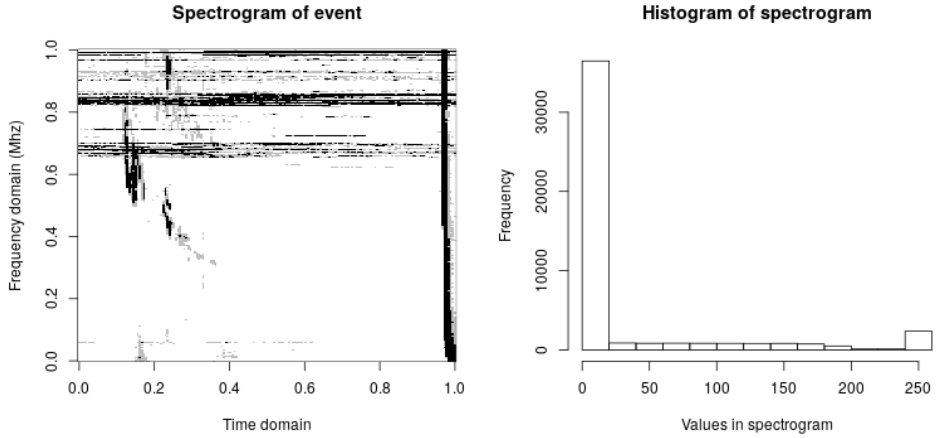


Figure 4.6: *Example of WAVES spectrogram.*

with the same duration. For each sample, each element represents the intensity values at this time (column) and frequency (row). For the positive subset of the data, these spectrograms are accompanied by a binary mask which provides a pixel-wise mask as to the location of the Type II solar burst. Binary masks were not provided for Background and Type III classes but rather focus solely on the task of detecting Type II bursts. For our task, the binary mask was used to construct a bounding box to encompass the Type II burst.

4.3.2 Base Model

The Faster R-CNN [178] was used as the base model for the experiments of this case study. Faster R-CNN performs object detection on images by first producing a feature map at various scales that can be shared between different sub-networks, thereby improving the speed at which it may make detections. The feature maps are passed to a Region Proposal Network (RPN) where potential objects are detected. These proposals are passed to the Region of Interest (ROI) sub-network, along with the feature map to regress the proposals to the original image dimensions. This model takes an image as input, and produces many outputs: (1) a bounding box; (2) a class label associated with outputs with the bounding box; and (3) a objectness score for the outputs. In our learning task, there is only 1 class label, that of the Type II burst.

The success of Faster R-CNN for object detection has spawned a number of methodological improvements, such as Mask R-CNN [179]. Mask R-CNN, in addition to performing bounding box detection, adds an additional sub-network to predict a pixel-wise mask for each object. While Mask R-CNN was not used in this thesis, the methods presented here are transferable to this network as they have an identical methodological base network. However, in future work, we shall investigate how our methods affect the pixel-wise segmentation as well as the bounding box detection.

Table 4.10: *Number of samples in the train and test set per burst and activity type. Data is shown in the format: frequency (percentage).*

		Dataset	
	All	Training	Testing
Total	731	511 (100)	220 (100)
Burst Category			
Background	245 (33.5)	172 (33.7)	73 (33.2)
Type II	244 (33.4)	170 (33.3)	74 (33.6)
Type III	242 (33.1)	169 (33.1)	73 (33.2)
Solar Activity			
Low	144 (19.7)	101 (19.8)	43 (19.5)
Medium	266 (36.4)	186 (36.4)	80 (36.4)
High	321 (43.9)	224 (43.8)	97 (44.1)

4.3.3 Experimental Setup

For all experiments presented in this work, the same training process was used. In this section, the process of training the DNNs to detect Type II solar bursts is described.

The implementation of Faster R-CNN is non-deterministic. Therefore, the DNN was trained ten times and the mean and standard deviations of the results were reported. This reduced the possibility of a perceived improvement in performance being a result of the random process. Faster R-CNN used a pre-trained ResNet-50 backbone to compute the feature maps from the input. The last 3 layers of the backbone were fine-tuned during training. Pre-trained weights (trained on the COCO dataset) were used for the rest of the Faster R-CNN architecture and were also fine-tuned. The weights were optimised using SGD (learning rate 5e-3, momentum 0.9, and weight decay of 5e-5). The learning rate were warmed up during the first epoch to minimise large weight gradients due to the pre-trained weights. The Faster R-CNN were continually trained on the training set until the loss on the validation set did not improve for ten epochs. The model weights with the lowest validation loss was saved to disk. After Faster R-CNN completed training, the model's weights that had the lowest validation error during training was re-loaded and used to create the test predictions. PyTorch 1.6.0 and a single Nvidia 2080 GPU was used to train the Faster R-CNN, which for this dataset completes training in approximately 1 hour.

To facilitate the training of a DNN, WAVES was split into 70% for training and 30% for testing. To create these splits, stratified random sampling was applied to the 3 types of burst categories (background, Type II, and Type III) and solar activity (low, medium, high). Performing random sampling in this manner ensures that the model is exposed to all types of solar activity and burst categories equally during training. During training, 15% of the training data was used for validation at the end of each epoch.

Table 4.10 shows the number of samples used for training and testing. Approximately 33% of each of the different burst categories were included in the training and test-

ing set. However, the original WAVES dataset includes different proportions of solar activity examples. Therefore, these samples were distributed among the training and testing sets with respect to these original proportions, i.e. 19.7% of the original data is of low solar activity, therefore, both training/testing splits have approximately this amount as well.

4.3.4 Evaluation Metrics

To evaluate the model's effectiveness, we follow the evaluation method of [2], where the Precision, Recall, and their harmonic mean: F_1 score was calculated from the test predictions. These classification metrics were induced from the intersection of the predicted bounding box with the ground-truth bounding box of the different segments of the Type II burst. If these bounding boxes did indeed intersect, this was counted as true-positive. False-positives occurred when a predicted bounding box did not intersect with a ground-truth bounding box. If Type II bursts were predicted where there were none, this was flagged as a true-negative. While bounding boxes were used to detect Type II bursts, due to the shape of the burst, they will often contain large portions of the background, and scores based on this bounding box may change dramatically if small and faint segments of the burst are missed. This can make the metrics extremely sensitive if such an implementation was used for a real-world solar burst detector. In future, pixel-wise masks should be computed, such as with Mask R-CNN, to reduce the sensitivity of metrics.

The Intersection over Union (IoU) score was used an additional form of comparison between experiments. This metric evaluates how closely the predicted boxes match the ground truth boxes through the ratio between the area of intersection and the area of their union. Those situations where the boxes are very similar in size and placement will result in an IoU score closer to 1 as the intersection and union areas will also be a very similar number. On the other hand, boxes that are not very similar in placement or size will result in an IoU score closer to zero. The IoU score is in the range $[0, 1]$, where higher values indicate better performance.

As a Type II burst could be made up of multiple segments, there is a possibility for the model to produce many true-positives and false-negatives per image. If the model places a single large bounding box over many segments of the burst, this will count as many true-positives, but also reduce the IoU score as much of the background is included within the predicted bounding box. Therefore, when analysing the results, both metrics should be considered. If a model misses a segment of a Type II burst, it is counted as a false-negative even if the main part of the burst has correctly been detected. Therefore to achieve a good F_1 score, all segments of the burst will need to be localised within the image.

This method of measuring the classification score follows the design presented in [2]. However, for the detections using bounding boxes, rather than pixel-wise masks, the classification metrics communicate less information individually as they require only single pixel overlaps, which may be background noise. The IoU metric will inform how well the DNN is able to align the bounding box to the ground truth, while the classification metric is a quantification of the ability to detect Type II bursts while

avoiding other misleading features such as Type III bursts. Thus, in all evaluations of this work, we show the IoU metrics in combination with the classification metrics to better investigate the relative performance of the DNNs.

A further analysis is provided in Chapter 9 with the use of Receiver Operating Characteristic (ROC). This type of analysis provides deep insight into the trade-off between sensitivity and specificity at different positive prediction threshold values of the bounding box made by Faster R-CNN.

Chapter 5

Feature Specialisation

In the process of learning, DNNs optimise a set of weight matrices and bias vectors that, when applied to the input, can create meaningful latent representations of the input for the downstream task, be it classification or regression. While through the process of learning DNNs could produce informative latent representations of the data in order to perform their task, these latent representations may be optimised by prior knowledge provided by the domain expert. This chapter, recognising this process of learning, evaluates how prior knowledge may be integrated into this process to further enhance the feature specialisation process.

These feature specialisation methods are sorted into two distinct categories. The first is *indirect specialisation*. This type of specialisation is complementary to the already existing learning process where additional constraints are placed on the objective function that can improve the model's generalisation and prediction capabilities. As this type of specialisation is implemented through the objective function, it is possible to indirectly add domain knowledge to enhance the latent feature extraction capabilities of the model through the use of back-propagation. It is *indirect* in that it does not precisely specify how the latent representations should be improved through their usage of these methods. In this chapter, indirect specialisation is used to integrate prior knowledge of OG processes into an OG detector. We also used indirect specialisation to add knowledge of Type II and background properties for the detection of solar bursts. Finally, physics knowledge of the chemical makeup of molecules and crystals was used to improve energy estimations of quantum chemical systems.

The second method is *direct specialisation*, where the DNN's latent representation is updated to reflect the known relationships in the data as told by the domain knowledge. These methods are *direct* in so far as they specify exactly how the representation should be updated to match the domain knowledge instead of being inferred through the learning process. To test these direct specialisation methods, prior knowledge of how groomers usage certain words to mean similar things is used to augmented the

base DNNs. Specifically, this knowledge of word usage is used to update a pre-trained word-embedding, for which three potential methods have been provided.

5.1 Background Study on Indirect Specialisation

DNNs are inductive learners. Using the training signal (the error between its prediction and ground-truth output), the model will infer the relationship between the input and output to improve its generalisation performance, providing there is enough variety of data to infer such relationships. To create this training signal, the DNN learning process will typically include a cost function and use the output ground-truth for the singular task the DNN is being designed for. This type of learning is called Single-task Learning (STL) [62].

Take for example the following objective function:

$$\mathcal{L}_\theta = \frac{1}{n}\sum_{i}^{n}(\hat{y}_i - y_i)^2$$

where the loss of the model is measured by the average squared difference between the model's prediction of each input $\hat{y}_i$ and the ground-truth label y_i with respect to the model's parameters θ. This objective function provides a feedback signal on how to improve θ through back-propagation to reduce the residuals on the training data.

While STL is certainly successful at producing an accurate model with enough input data, it misses potential gains in performance from the process of Multi-task Learning (MTL).

MTL, as opposed to training for a single task, can improve learning through the addition of complementary and related tasks. These related tasks help the DNN in learning by providing more *hints* (Definition 2.0.1) as to the input-output relationship in the form of prior domain knowledge, thus enhancing the training signal, and ultimately, improving it's generalisation performance on the singular intended task.

While there may be many methods to incorporate a MTL process (see Zhang and Yang [180] for a comprehensive review), one method is simply to combine many objective function in the learning process:

$$\mathcal{L}_\theta = \frac{1}{n}\sum_{j}^{J}\sum_{i}^{n}\alpha_j * (\hat{y}_i^j - y_i^j)^2$$

where the loss is measured over J outputs and modulated by an optional α value to denote the importance of the task. In this example, residuals for all tasks are measured using the mean squared errors. Nevertheless, it is perfectly reasonable to combine both classification and regression loss terms providing the magnitude of each residual does not overpower each other. This magnitude may also be controlled through the use of

the (possibly learnable) α modulation. In some cases, the use of homoscedastic uncertainty w.r.t. the auxiliary tasks may better help control each of the task weightings than manually selecting a value prior to training [181].

There are many reasons why MTL may work in favour of STL:

1. Reduces over-fitting by providing many training signals which the model must perform well at.
2. Act as a form of regularisation [182]. Indeed, methods of applying L_2 or L_1 to the objective function also serve the purpose of MTL. In this case, the regulariser term is performing the task of reducing the magnitude of the weights. Though this is not directly related to the main task, it can still provide generalisation benefits commonly seen with MTL.
3. Constrains the model's internal representation as it must be shared between tasks. These auxiliary tasks must share an internal representation, and thus to perform well on these many tasks simultaneously, it must find a representation that fits all these tasks.
4. It may act as a constraint on the hypothesis class as an inductive bias [14]. Since these tasks are related, the intersection of the respective hypothesis class for each task may contain the optimal solution for the intended usage of the model, but also reduce the search space through this intersection:

   $$\hat{f} \in \mathcal{H}_i \cap \mathcal{H}_j$$

   where $\hat{f}$ here is our optimal function for $\hat{f} \approx f$, f is the true-labelling function to approximate, and $\mathcal{H}_i, \mathcal{H}_j$ are the hypothesis classes of tasks i, j respectively.

5.2 Application of Indirect Specialisation

MTL via auxiliary tasks, through the process of indirectly specialisation the latent representation, provides a opportunity to integrate prior expert knowledge into the training process of DNNs. To evaluate the use of auxiliary tasks for integrating prior knowledge, we have used auxiliary tasks to integrate the knowledge of OG processes for the OGD case study, integrated the physics knowledge of chemical properties for the Quantum Chemistry case study, and added auxiliary losses to predict Type II and background properties in the Solar Burst detection case study.

5.2.1 Integrating Knowledge on OG Processes

To enhance an OGD detector, auxiliary tasks were used to encourage our OG detectors to better model the themes of groomers. The annotated samples of OG processes were associated to three-word collocates, which were used to identify contexts of interest. The presence of the seven OG processes (Table 4.5) were defined using continuous

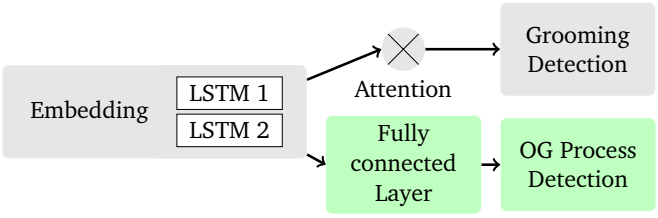


Figure 5.1: *An additional OG process detection branch (green) is added to the classifier module (in OGD-E for illustration). This additional OG process branch predicts the a pseudo-likelihood of OG process for each word in the conversation.*

representation with a Gaussian Mixture Model (GMM) whose components are centred on the associated three-word collocates and standard deviation being the span of each collocate (max. seven words as mentioned in Section 4.2.1). Seven GMMs are defined for the seven OG processes. The GMMs, $\mathcal{G}$, provide a pseudo-probability of an OG process occurring at word w_t at timestep t for each of the seven OG processes.

To predict the response of $\mathcal{G}$, a second output branch was added to the DNN after the LSTMs (Figure 5.1), where the output of the LSTM is the hidden state at each timestep. In creating the grooming prediction, an attention layer uses the state of all timesteps to create a single vector representation of the entire conversation that is used for classification. For estimating the pseudo probability of OG processes for each word, a fully-connected layer was applied at each timestep and optimised using the mean-squared error loss over all timesteps in the conversation:

$$\mathcal{L}_{GMM} = \frac{1}{T}\sum_{t}^{T}\Big[\sigma(W \times \mathbf{h}^{(t)} + b) - \mathcal{G}(w^{(t)})\Big]^2$$

where $h^{(t)}$ is the hidden state at timestep t. W and b are the weight matrix and bias vector of the fully-connected layer, σ is the ReLU non-linear activation function, and $\mathcal{G}$ is the response of the seven GMMs for the word $w^{(t)}$.

For OGD-R, we experimented with adding an attention block as in the main branch, but found that this did not help with the OG process detection, probably because this task is more local and does not need to consider as large a context as classification of the whole conversation.

The new branch also served as additional regularisation to prevent overfitting given the class imbalance between (non-)OG chat logs, where the non-OG chat logs consisted of ∼96% of the test set. Further, it allowed for an OG process-based interpretation of what the DNN considers relevant clues for OG classification. For each word in the conversation, the DNN outputs seven values corresponding to the seven grooming processes. Non-zero values indicates the model is predicting a grooming process is occurring at these words given the surrounding context. This prediction that a grooming process is occurring is indicative of the DNN classifying the conversation as grooming. Therefore, these auxiliary tasks, while indirectly specialising the latent features for the classification task, have also helped in enhancing the interpretability of the model. These interpretations will be explored and visualised in Section 5.4.1.

The loss of this secondary task was added to the original classification loss and modulated with a α term:

$$\mathcal{L}_{total} = \mathcal{L}_{class} + \alpha\mathcal{L}_{GMM}$$

For the experiments we used $\alpha = 0.3$, though other values may yield better results.

5.2.2 Estimating Auxiliary Physical Properties

Using the MTL technique, the GNN's node states are specialised to relate more to relevant physical properties. We experimented with a combination of low- and high-level physical property estimation using auxiliary tasks:

1. The number of atoms of each type present in the system. This low-level property can be correlated with the system energies, as heavier atoms will generally require more energy to bind or separate but release less net energy as part of the process.
2. The number of orbitals associated with each atom type provides clues to infer the potential energies. As DFT approximates the system energies from the electronic density, therefore by estimating the number of orbitals associated with each atom type provides some clue as to the potential energies.
3. A probability distribution for the scaling to the stable geometry, estimated as a Gaussian function as in [183]. This high-level property may encourage the model to infer the relationship between system-wise property of energy and stable Angström distances.

For each auxiliary estimation, a new output layer was added, and a mean-squared error loss term was minimised during training. Each auxiliary loss term was weighted by an empirical $\alpha = 0.3$ hyper-parameter while the original potential energy loss kept a weight of 1 to emphasise the energy loss more:

$$L_{total} = L_{MSE}(y, \hat{y}) + \alpha \sum_{a \in \mathcal{A}} L_{MSE}(a, \hat{a}) \tag{5.1}$$

with y and $\hat{y}$ the ground-truth and predicted potential energies, and $\mathcal{A}$ the set of auxiliary quantities a being used. In future works, we may investigate auxiliary quantities linked to the physics of individual atoms/nodes in addition to the system-wise properties.

5.2.3 Predicting Solar Burst Properties

Whereas in the case study of chemistry physics knowledge that estimates different levels of system-wise properties, the auxiliary predictions of the solar physics case study

differed by predicting two classes of properties: the Type II burst and background. The Type II burst properties included the frequency range and time duration of the burst. For these properties, we might expect the DNN to better understand the shape of the burst in order to jointly predict the low-level features as well as predicting locality.

To create these predictions, a small sub-network used the feature representation of the image, and predicts the frequency and time duration ranges of the burst. This feature representation was passed to both RPN and classifier sub-networks in addition to this new sub-network for Type II burst range predictions. An auxiliary mean-squared error loss was added to train this sub-network and proceeding feature extraction layers.

For the prediction of the background properties, the different levels of solar activity was used. While the solar activity varies over many months, as the WAVES dataset is segmented into fixed periods of time, the amount of solar activity in WAVES can vary from one image to the next. This level of solar activity may be indicative of how much surrounding noise[1] is present that may distract the object detector. For example, with more intense levels of solar activity, the Type II burst may be obscured by Type III bursts, thus making the detection task more difficult. This auxiliary estimate has aimed at making the DNN more aware of the varying background properties and able to account for them.

Therefore, providing hints as to the amount of solar activity (low, medium, high)[2] may help the object detector to be more accurate in its prediction of Type II bursts. To accomplish this task, an auxiliary task was added object detector to predict the level of solar activity (Equation 5.2). From the encoded feature representation of the image, the class label of the solar activity was predicted, and additionally passed to the RPN and classifier sub-networks for prediction of objects.

$$\mathcal{L}_{activity} = -\sum_{k=1}^{K} y_o^k \log(p_o^k) \tag{5.2}$$

where we have 3 K classes for the low, medium, and high solar activity for image o. $y_o^k \in \{0, 1\}$, and 1 if the image was of this solar activity. This cross-entropy loss was added to the total loss of the Faster R-CNN.

5.3 Direct Specialisation

In contrast to indirect specialisation, where the model is encouraged to find a latent representation that better matches the problem tasks, direct specialisation aims to improve the existing internal representation by applying direct and specific changes to better reflect the knowledge of the domain expert. Using domain knowledge of

[1] Noise in this context refers to the signals other than the target i.e. the Type II solar burst. Noise could therefore refer to Type III bursts and other background radiation that makes the object detection task more difficult.

[2] See [2] for details on how these categorisations of solar activity were created.

linguistics, word embedding spaces, and the semantic representation found in such spaces, are used in this work to demonstrate this process of direct specialisation.

5.3.1 Latent Representations for Word Semantics

Learning a useful representation of language can be difficult. Firstly, there are many tens-of-thousands of words and to encode each one in a one-hot representation would be impractical from a computational perspective. Secondly, this one-hot representation does not capture the knowledge of similarity between words, where some words mean very similar things.

A popular technique to represent language and overcome these challenges are word embeddings or word vectors, where every word is instead represented by a fixed-length vector [184]. These representations of words are interesting as we find each word's distance to each other roughly equivalent to its semantic similarity [185].

However, to correctly and accurately model these semantic relationships requires substantial amounts of text for training. Indeed, the larger the corpora to learn such representations, the more useful they become [186]. It is, therefore, a challenging task to correctly learn the correct semantics of words from groomers when the data of grooming conversation is very small. Therefore this work relies on specialist knowledge of how groomers communicate in order to better model this information.

5.3.2 Direct Specialisation of Usage of Grooming Language

Using the knowledge of Word Semantics Representation (WSR), our methods have been designed to improve the existing structure to better encode the domain knowledge of how groomers have used language that the DNN may not recognise due to the small amount of positive grooming data.

During the CL analysis process (Section 4.2.1), the linguistic experts have normalised the usage of words in context, based upon their intended meaning. Word variants being words with the same semantic meaning, was used by linguistics used to reduce the variance within the text. This reduction in variation of language enables the linguistics to analyse statistical patterns that may otherwise be hidden through misspellings and alternative vocabulary. Therefore, integrating this knowledge into DNNs to capture the same word semantic representation used by linguistics may allow the DNN to exploit the same statistical patterns.

Although variants have the same intended meaning, some may be discriminative of groomers' language. Hence, it may be helpful for OG classification to keep them apart in the WSR space. The significance of word w for classification is determined based on the absolute difference of empirical occurrence in OG and non-OG conversations within the training set:

$$\delta p(w) = |p(w|y_{pos}) - p(w|y_{neg})|. \,,$$

Table 5.1: *Comparison of occurrence frequencies for selected variants and all words in the corpus.*

	Mean	Standard Deviation
Word Frequency	$3.922 \cdot 10^{-05}$	0.001
Variant Frequency	0.001	0.002

where $p(w)$ gives the probability of word w occurring in the positive or negative labelled subset of the data. If $\delta p(v_k^i)$ or $\delta p(v_k^j)$ are high (i.e. within the last 5 percentiles for all words), *we do not use the pair for modification of the WSR*. We considered increasing the separation between these variants, but we found that this modifies the word embedding too much and reduces its semantics representation power. Out of 4590 pairs of variants, 2955 were retained for modification of the WSR. In effect, this selective WSR modification applies an implicit and selective text normalisation which supports the OG classification.

The final set of word variants is defined as:

$$\mathcal{S} = (v_1^i, v_1^j), (v_2^i, v_2^j), ..., (v_n^i, v_n^j)$$

The effectiveness of WSR modification using selected variants is emphasised by the frequency of usage of these variants within the corpus: modifying the WSR space around frequently used words may have a larger impact on subsequent text analysis than adjusting it around rare words. The frequency of any word w_i was calculated as: $\mathcal{F}_{w_i} = \frac{\text{count}(w_i)}{N}$, where N is the total number of words in the corpus. Occurrence frequencies for the selected variants and other words are provided in Table 5.1. We see that the average frequency of selected variants is significantly larger than for other words in the corpus by two orders of magnitude. This may explain in part the effectiveness of the selective WSR normalisation.

The mean occurrence frequency of variants in the OG corpus is significantly larger, by two orders of magnitude than that of all words (Table 5.1). Therefore, using these common words to modify the WSR may have a strong impact on classification.

Three strategies have been proposed to modify the WSR based on our set of n pairs of variants using the principle that *words with the same semantic should be moved closer to each other in the WSR space*. These three different implementations may apply to different usage scenarios, such as training a new language space (supervised WSR modification) or modifying an existing one before training a new classifier (manifold-based) or before fine-tuning an already existing classifier (elastic pulling). While these integration strategies have been evaluated using the manipulation of word embeddings to improve the WSR in our case study, these strategies can be generalised to many other forms of latent representations, provided the availability of suitable prior knowledge of which representations should be pulled together in the latent space.

5.3.3 Supervised WSR Modification

Prior to the classification of OG chat logs, two-layer LSTM language model was trained to learn semantically related word embeddings.

To incorporate domain knowledge of the word variants, a regularisation term was added with weight λ to the language modelling loss $\mathcal{L}_{WSR}$ to minimise the L_2 distance $\mathcal{D}$ between the selected word variants' word embeddings:

$$\tilde{\mathcal{L}}_{WSR} = (1-\lambda)\,\mathcal{L}_{WSR} + \alpha\left[\frac{1}{n}\sum_{k=1}^{n}\mathcal{D}(v_k^i, v_k^j)\right] .$$

5.3.4 Manifold Learning

Supervised WSR requires a training process to incorporate the knowledge of word variants. Manifold learning, on the other hand, allows the direct modification of WSR with the modification of computed pairwise distances of words.

Using the pre-trained WSR, a pairwise distance matrix was computed using the L_2 distance metric $\mathcal{D}$. The distance entry for each word variant was reduced:

$$\tilde{\mathcal{D}}(v_k^i, v_k^j) = \lambda\mathcal{D}(v_k^i, v_k^j)$$

where $\lambda \in [0, 1]$ modulates the strength of distance reduction between selected variants. While in our work, a single λ value was used for all word variants, in future work, different λ values may be used to modulate the strength of reduction with respect to how often the two word variant pairs have the same sentiment. If two words are most often used with the same sentiment, the distance between these two words in the WSR should be further decreased.

After reducing all word variants, a Robust Diffusion Map [187] was used to project from a distance matrix back into WSR space. Robust Diffusion Map was used as it conveniently computed a WSR based on the distance matrix, but other manifold learning methods could be explored.

This implementation requires re-training subsequent layers of the DNN from scratch, as words' new embeddings may be very different from initial ones.

Note that this is an unusual use of manifold learning for WSR modification rather than for dimensionality reduction. It is also possible to reduce dimensionality, which may help to combat overfitting for classification, as discussed in [188]. However, the dimensionality of the WSR is unchanged in our experiments.

5.3.5 Elastic Pulling

The third implementation modifies the existing WSR space *in place* through local movements that pull together the selected variants. This process is depicted in Figure 5.2. Elastic Pull pulls two selected word variants closer together in the WSR, while also having a small effect on these variant's neighbouring words. This mostly preserves all words' original word embeddings, thus limiting the amount of change needed for the classifier to simple fine-tuning.

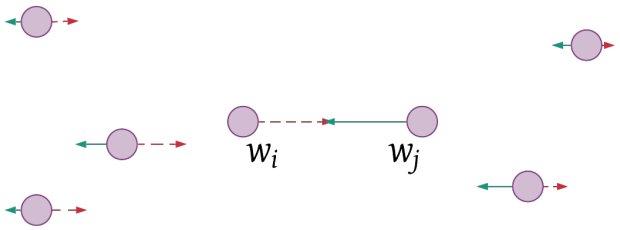


Figure 5.2: *Depiction of the Elastic Pulling process between two word-pair variants. Two variants w_i and w_j are pulled together in the WSR while each of its neighbouring words are also pulled closer to preserve the initial semantic relationship with the variants. The influence of this pulling process is determined by an RBF centred on each of the word variants. Each of the neighbouring words are pulled in the direction of each word variant.*

Two variants v_k^i and v_k^j of coordinates $\mathbf{x}_k^i$ and $\mathbf{x}_k^j$ are pulled towards their geometric centre:

$$\hat{\mathbf{x}}_k = \frac{\mathbf{x}_k^i + \mathbf{x}_k^j}{2}$$

by the amount in the direction of

$$\delta_{\mathbf{x}_k^i} = \hat{\mathbf{x}}_k - \mathbf{x}_k^i$$

where this movement towards the centre of word pairs is modulated by

$$\lambda \in [0, 1] : \tilde{\mathbf{x}}_k^i = \mathbf{x}_k^i + \lambda \delta_{\mathbf{x}_k^i}.$$

Like Manifold Learning, we used a single λ value to modulate the movement for all word variants. However, providing a λ value for each word variant would enable greater control over the movement of word variant pairs.

The pull operation propagates to neighbouring words, with strength of pull decreasing with distance (i.e. modulated by a radial basis function (RBF) ϕ_k^i centred on $\mathbf{x}_k^i$), so as to preserve the pairwise relationships between variants and their neighbours:

$$\tilde{\mathbf{x}} = \mathbf{x} + \lambda\, \phi_k^i(\mathbf{x})\, \delta_{\mathbf{x}_k^i}$$

Table 5.2: *Impact of each CL augmentation on OG classification. Bold are improved results.*

Model	Strategy	AUPR	F_{max}	F_1	$F_{0.5}$
OGD-R	No augmentation	0.867	0.832	0.829	0.851
	Aux. OG process detection	**0.873**	0.825	0.825	**0.863**
OGD-T	No augmentation	0.940	0.889	0.886	0.894
	Aux. OG process detection	**0.943**	**0.895**	**0.889**	**0.906**

For our experimentation, we used an inverse multiquadric:

$$\phi_k^i(\mathbf{x}) = (||\mathbf{x} - \mathbf{x}_k^i||^2 + \gamma^2)^{-\frac{\beta}{2}}$$

The inverse multiquadric provides global support so that all words can be considered without costly intersection tests needed, and with β and γ tuning the RBF's decay rate i.e. the locality of propagated pull. The method is not very sensitive to these values as long as the pull's reach is sufficient, within a radius of the order of magnitude of $\delta_{\mathbf{x}_k^i}$, and we set them empirically to 1.0 and 3.0 respectively.

5.4 Experiments

To verify our feature specialisation approaches, the indirect feature specialisation methods were implemented for all three case studies: OG detection in chat logs, energy estimation, and detection of Type II bursts. While for direct specialisation, we implemented all three proposed direct specialisation methods to update the WSR of the OGD base models.

5.4.1 Indirect Specialisation

The experiments begin with the indirect specialisation using the auxiliary tasks of predicting the results of GMM-based annotation. This method was compared against both base models OGD-R & OGD-T for the detection of OG. The results are presented in Table 5.2.

For both base models, we observed that the auxiliary task of predicting the GMM annotations of grooming processes helps improve the models' performance. This was more clear for OGD-T where all metrics improve. This has provided us with some indication that the auxiliary tasks are enabling the models to detect the more subtle nuances in the grooming strategies by providing more hints during the training process.

While these performance improvements are very slight in these cases of indirect specialisation, we demonstrate in a later chapter (Chapter 9) how the combination and interaction of the indirect methods with other integration strategies can lead to more significant improvements.

Table 5.3: *Impact of each physics integration strategy on energy estimation on the Augm. QM9 dataset for MPNN (top) and SchNet (bottom) base models. Results are presented in the format: mean (std). The specialised interaction methods that optimise at best energy and geometry are highlighted in bold for each model.*

	Strategy	AE	RE	DSG
	MPNN base model	0.242 (1.318)	0.0029 (0.015)	0.034 (0.074)
Aux. estimates of	# atoms of each type	0.171 (0.986)	0.0021 (0.011)	0.031 (0.046)
	# orbitals	0.195 (0.545)	0.0025 (0.006)	0.033 (0.046)
	distance to stable geom.	0.119 (0.698)	0.0015 (0.008)	0.033 (0.046)
	SchNet base model	0.038 (0.037)	0.0005 (0.0005)	0.020 (0.031)
Aux. estimates of	# atoms of each type	0.038 (0.036)	0.0005 (0.0005)	0.032 (0.033)
	# orbitals	0.036 (0.035)	0.0005 (0.0005)	0.022 (0.032)
	distance to stable geom.	0.049 (0.044)	0.0007 (0.0006)	0.037 (0.033)

Since the augmented models make use of OG processes' recognition to capture the language associated with grooming, their auxiliary OG process detection may be used to highlight, at the word level, those parts of the conversation that the model associates to OG processes. These are visualised in Figure 5.3, where the estimated likelihood of the *Compliance testing* process is indicated in shades of red. The DNN focused on questions about a personal situation and on invitations to talk over the phone as indicators of compliance testing happening, in line with our general understanding of this OG process. While these elements of discussion may seem neutral enough and may not be captured by generic OG classifiers, the DNN's understanding of OG processes and of their relation to OG made it increase the OG classification score. The words and local contexts associated with these OG process detections are indications of the language cues which supported the OG classification.

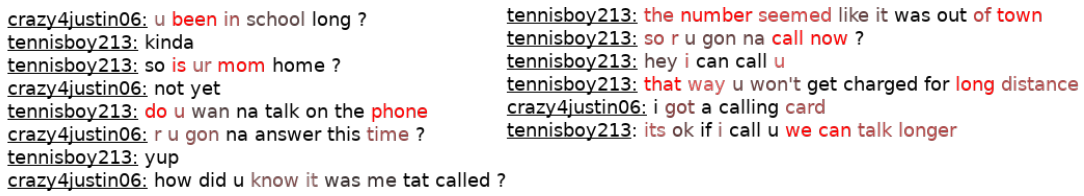


Figure 5.3: *Graphical representation of the DNN's predicted likelihood of the* Compliance testing *OG process. Higher likelihood values are indicated in red, while lower values are represented by black colours.*

The next experiment uses the quantum chemistry case study. Once more, the two base models of this study are used and each of the auxiliary proprieties were applied in turn. The results of these experiments are presented in Table 5.3.

The auxiliary estimation of physical properties has improved the performance for both MPNN and SchNet base models. Estimating the distance to stable configuration was the most effective strategy for augmenting MPNN, while SchNet benefited more from estimating the number of orbitals. We interpret the overall improvements as the models implicitly discovering and encoding useful physics representations required for accurate predictions. Indeed, the numbers of atoms of each type and of orbitals are

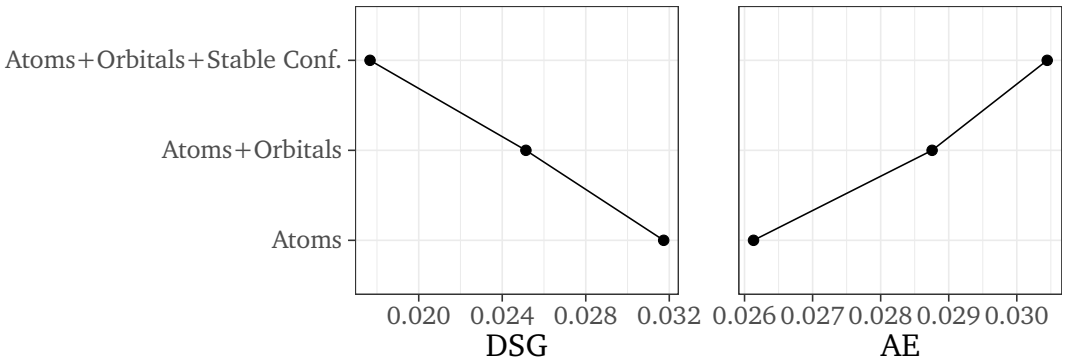


Figure 5.4: *Effect on DSG and AE performance when the auxiliary outputs are incrementally added.*

related to the global properties of the molecule and its atoms. The distance to stable configuration is related to a high-level system optimisation, with the DNN learning to estimate the energy and at the same time assess how close/far it is from the minimum. Its improved performance may indicate that the DNN's features better capture the dependency of energy on geometry. In the case of MPNN, we reckon that these more geometry-related features compensate the less complete accounting of atomic interaction in MPNN, which is due to only considering bonded (neighbour) atoms, and which hinders a complete capture of geometry in the non-augmented base model.

In order to explore the potential performance improvement to be gained through multiple auxiliary outputs, each of the outputs were added to Augmented-SchNet in turn and train the model using Augmented-QM9. First by adding the auxiliary prediction of number of atoms of each type (labelled 'Atoms'), then adding to this by including the number of orbitals prediction, and so on. The performance impact for both the DSG and AE metric are shown in Figure 5.4. From this experiment, the DSG metric has improved through the addition of more auxiliary estimations, while the AE metric has slightly worsened (from 0.026 to 0.031). This result indicates the auxiliary estimations are contributing to help with the downstream task of estimating stable configurations of molecules more accurately, even while being very slightly less accurate for the immediate task of energy estimation. This positive result encourages us to continue to use all three auxiliary tasks for the final Augmented-SchNet model.

Finally, an experiment was conducted by adding an auxiliary tasks to the Solar Burst detection model. In this experiment, the Type II auxiliary tasks (predicting frequency and time ranges) and background property task (predicting the level of solar activity) were added to the DNN. To create the solar activity prediction, a two-layer CNN sub-network was used to condense the feature representation from the ResNet101 head. After the CNN sub-network, a linear layer projected the feature representation from ResNet101 into the classification space.

In Table 5.4, the results for the base model Faster R-CNN and Faster R-CNN with the different auxiliary tasks separately are shown. Despite the hypothesis about the information regarding the level of solar activity, the background auxiliary task here

Table 5.4: *Comparative performance results between base Faster R-CNN and Faster R-CNN with auxiliary target prediction of solar activity. Results are shown over 10 trails in the format: mean (standard deviation).*

Model	Precision	Recall	F_1 Score	IoU	IoU (Type II)
Baseline	0.335 (0.090)	0.854 (0.104)	0.468 (0.071)	0.661 (0.013)	0.172 (0.009)
with Type II aux.	0.507 (0.066)	0.350 (0.043)	0.412 (0.044)	0.667 (0.003)	0.195 (0.009)
with Background aux.	0.392 (0.040)	0.481 (0.078)	0.427 (0.032)	0.668 (0.008)	0.192 (0.013)

does not help the model improve on its object detection task. This is also true for the Type II tasks. While being more precise in its prediction, the F_1 score decreases due to the significant decrease in recall. While these methods have not had a entirely positive effect on F_1 score, the IoU score has increased suggesting that with a different implementation choice, these auxiliary tasks may yield more promising results. These different implementation choices could be tested in the future for a fuller evaluation of the potential of these auxiliary tasks.

Despite the decrease in F_1 performance, when measuring the IoU scores for all experiments, we observe the IoU score for the Type II bursts increased, meaning that when the auxiliary predictions are added to the model, the model, although failing to detect some Type II bursts, has created better-aligned bounding boxes when true Type II bursts are detected. This was further demonstrated by the increased IoU scores for the Type II events for both type of auxiliary predictions.

From this case study, we see that it may be possible to use the auxiliary tasks to reduce the false-positives created by the detector, while also creating more accurate localisations for the Type II bursts. This work may be further tested in future works by considering segmentation performance.

5.4.2 Direct Specialisation

The individual effects of the different WSR algorithms are evaluated in Table 5.5 in comparison to non-augmented models. To test these algorithms, the WSR method was applied to a pre-trained word embedding. For this, two versions were used, the first was a simple two-layer LSTM language model trained using the grooming chat logs. The second was with all varieties of the pre-trained GloVE representations. As GloVe includes many pre-trained variations (these variations are from the different number of tokens, dimensionality, and training data source location), thus we trail each of the WSR algorithms with each of the variations.

The 300 dimensional language model of OGD-R was trained from random weights on the OG dataset, then the WSR was fine-tuned while the classifier was trained from random weights on OG classification. To demonstrate the general applicability of the selective text normalisation based on CL knowledge, the results are initially presented using the GloVe WSR [176] with 300 dimensions, pre-trained on Common Crawl 840B[3] and fine-tuned on OG classification. Further experiments were conducted with all vari-

[3] `http://commoncrawl.org/`

Table 5.5: *Impact of each CL augmentation on OG classification. Bold are improved results.*

Model	Strategy	Precision	Recall	AUPR	F_1	$F_{0.5}$	$\Delta\mathcal{D}/\overline{\mathcal{D}}$
OGD-R	No augmentation	0.867	0.794	0.867	0.829	0.851	– / 3.72
	Supervised modif.	0.834	0.765	0.824	0.798	0.819	0.75/0.91
	Manifold learning	**0.916**	0.753	**0.881**	0.827	**0.878**	0.65/1.29
	Elastic pulling	**0.878**	**0.808**	**0.877**	**0.841**	**0.863**	0.83/0.61
OGD-R using 840B - 300d GloVe	No augmentation	0.834	0.775	0.849	0.803	0.822	– / 5.87
	Supervised modif.	**0.836**	**0.781**	0.828	**0.808**	**0.825**	0.90/0.93
	Manifold learning	0.824	**0.796**	**0.854**	**0.810**	0.818	0.89/0.53
	Elastic pulling	**0.905**	**0.761**	**0.880**	**0.827**	**0.872**	0.92/0.73

Table 5.6: *average distance between pairs of selected variants $\overline{\mathcal{D}}_{var}$ and all other pairs of words $\overline{\mathcal{D}}_{non\,var}$ in the wsr spaces*

Method	$\overline{\mathcal{D}}_{var}$	$\overline{\mathcal{D}}_{non\,var}$
Base Model #1's original WSR	3.72	2.86
Supervised WSR modification	0.91	2.78
Manifold Learning	1.29	2.86
Elastic Pull	0.61	2.82

eties of GloVe to demonstrate the methods transferrability to different training sources as well as different dimensionality of WSR.

In Table 5.6 we show the average pairwise distances between pairs of variants and non-variants. This demonstrates each of the WSR normalisation algorithms ability to preserve the semantic power of the space.

We observed a performance improvement over the baseline when adding the direct specialisations to the WSR. Here Manifold Learning and Elastic Pulling increased the AUPR from 0.867 (baseline) to 0.877 (Elastic Pulling) and 0.881 (Manifold Learning) this improvement is considerable given the number of variants accounts for less than 6% of the overall word embeddings in the WSR.

While both Manifold Learning and Elastic Pulling have helped the model generalise better, the supervised approach has unfortunately reduced the performance of the resulting classifier. This may be explained by the new loss term conflicting with the original word embedding loss. This hypothesis may be further exhibited when Supervised modification is applied to the GloVe word embedding. In this situation, the algorithm was more successful in improving the performance over the baseline, as there could be less conflict between the original word embedding loss and the new less term as the word embedding loss is generally lower due to the word embeddings being pre-trained and only fine tuned with the supervised WSR modification.

It is worth noting, for OGD-R, that $\overline{\mathcal{D}}$ the average distance between the representations of two selected variants is at 3.72, higher than the average distance between all other pairs of words which is at 2.86. Therefore, OGD-R, even though fully trained

Table 5.7: *WSR algorithms applied to varying GloVe representations. Bold values denote the improvement over no augmentation.*

Augmentation	Precision	Recall	AUPR	F_1	$F_{0.5}$
6B - 300d					
No augmentation	0.834	0.775	0.849	0.803	0.822
Supervised modif.	**0.881**	0.745	0.838	**0.807**	**0.850**
Manifold Learning	0.799	**0.778**	**0.855**	0.788	0.794
Elastic Pulling	**0.871**	0.759	**0.851**	**0.811**	**0.846**
27B - 200d					
No augmentation	0.839	0.774	0.839	0.805	0.825
Supervised modif.	**0.865**	0.762	0.837	**0.811**	**0.843**
Manifold Learning	0.796	**0.804**	**0.854**	0.800	0.797
Elastic Pulling	**0.874**	0.763	**0.877**	**0.815**	**0.849**
42B - 300d					
No augmentation	0.870	0.759	0.847	0.811	0.845
Supervised modif.	0.861	**0.791**	0.846	**0.824**	**0.846**
Manifold Learning	0.832	0.762	**0.854**	0.796	0.817
Elastic Pulling	**0.879**	**0.763**	**0.870**	**0.817**	**0.853**
840B - 300d					
No augmentation	0.834	0.775	0.849	0.803	0.822
Supervised modif.	**0.836**	**0.781**	0.828	**0.808**	**0.825**
Manifold Learning	0.824	**0.796**	**0.857**	**0.810**	0.818
Elastic Pulling	**0.905**	0.761	**0.880**	**0.827**	**0.872**

on OG classification, was not able to discover on its own the knowledge that some variants have the same semantic meaning while not being discriminative for the OG classification task, and could therefore have the same or similar representations. This, together with the improved results from modifying the word embedding, demonstrate the usefulness of integrating this knowledge into the model.

These results for each of these various forms of GloVe WSRs are presented in Table 5.7. We have seen that for most of the variations of GloVe, Supervised modification improves the F_1 and $F_{0.5}$ score but not the AUPR and almost always improves the precision of the classifier (except for 42B - 300d), while the opposite is true for Manifold Learning. Elastic Pull has appeared to be the most consistent in improving the results. While the largest improvement in AUPR performance was applying Elastic Pulling to 27B - 200d, we observed the best classification results when using the largest GloVe word embeddings (840B - 300d). This result might be explained by the larger training corpus and dimensionality providing a richer WSR before the direct specialisation was applied.

5.5 Chapter Summary

This chapter has investigated the method of feature specialisation. DNNs, in the process of learning, already perform a method of feature specialisation - the internal representation is updated to perform well at a classification or regression task. Yet, many opportunities are missed to improve the generalisation and visualisation abilities of these features due to the concentration on a singular task.

In addition to relying on auxiliary tasks to optimise the representation, providing the intuition surrounding what it represents, it is possible to modify the representation to reflect prior knowledge directly. This was applied to word embeddings, where the representations of words are roughly equal to semantic distances. For these word embeddings, distances were reduced between words to reflect the expert's knowledge about how words are used by online groomers.

From the experimentation presented in this chapter, this direct manipulation of representation has been found to be more fruitful regarding the resulting model performance. However, the auxiliary tasks do, by their design, provide means to graphically represent potential criteria for the model's decision processes, aiding the user in making decisions based upon the predicted classification of the DNN.

The goal throughout this chapter has served to categorise and design methods to enhance these learnt features. These methods were tested on many different architectures and data types with potential successes in each case.

The contributions from this chapter are:

- A theoretical formation of indirect and direct specialisation. Where indirect specialisation can be accomplished via a multi-task learning process with auxiliary tasks that encourage the DNN to learn more informative feature relevant to the task. For direct specialisation, specific modifications guided by prior knowledge are made to the features that improve the representation.

- To verify the approach for indirect specialisation, different implementations of these key ideas were developed and tested using three case studies: detection of online groomers, energy estimation of chemical systems, and detection of Type II solar bursts. In both OGD and energy estimation studies (each consisting of very different architectures), we observed the performance of these models improved while providing an additional means of graphical representations (such as Figure 5.3) to help the users of the model understand a part of the decision process.

- We present three novel methods for applying direct feature specialisation to embedding spaces, or in the context of the OG detection study, a WSR. Each of these methods optimise the representation of word embeddings by reducing the distances between word variants to conform to the prior knowledge of how these words have been used in online grooming chat logs.

The key points from this chapter are:

- DNNs perform feature specialisation as part of their routine process of training.
- However, this process of training can be enhanced through the integration of prior knowledge, providing more hints to the DNN.
- This chapter outlines two categories of prior integration methods for feature specialisation: Indirect and direct specialisation.
- Indirect specialisation is the combination of complementary auxiliary tasks that provides the DNN with more hints during training time. These hints can possibly improve the feature representation of the model and therefore performs better at the task of classification and regression.
- Direct specialisation is the process of taking already learnt representations and augmenting them using prior knowledge of how these representations could be improved and become more accurate. Once again, we find that this method outlined here can have a positive effect on the performance of DNNs.
- Better performance improvements can be gained via direct modifications to the internal representations of the model than with indirect hints via multi-task learning and auxiliary estimates. However, it does require an intuition about the representation that is not always known to modify such a representation.

Chapter 6

Specialised Information Processing

In the previous chapter, different methods were introduced that specialise the latent representation of DNNs using domain knowledge. These methods either directly manipulate the model's latent representations of data or indirectly provide hints in auxiliary related tasks to the model during the training process. These auxiliary tasks then help the model improve in generalisation by forced sharing of internal representation between many related tasks.

As opposed to specialising the representation, this method of specialised information processing provide specific channels for processing representations. Each specialised processing channel represents a concept within the domain being modelled. These specialised information processing methods are implemented by unique weights and biases for each of the concepts. Whereas in the previous chapter the feature specialisation acts on the existing latent representations, this chapter demonstrates how separate and specialised information processing on the latent representations may better model domain concepts relating to the downstream task.

This chapter discusses the intuition behind specialised information processing, and explains the implementation of these mechanisms for recurrent and graph-based DNNs. This approach of using specialised information processing was tested using the OG detection case study, where specialised RNNs are used to learn the patterns of communicative processes. The specialised information processing in GNNs was tested through the chemical energy prediction case study.

6.1 Background on Encoding Specialised Concepts with Specialised Information Processing

In many cases, experts can leverage feature specialisation, such as indirect specialisation, to enhance the generalisability of DNNs by finding a representation that learns to fit many tasks jointly. If the tasks are closely related and are on the same semantic level, it is common to see small performance improvements [89, 90]. However, to properly use indirect specialisation, one must weight each of these different tasks to ensure that the loss term of one task does not over power another, especially the main task the DNN is being trained for, otherwise one may see optimisation conflicts [86].

Specialised information processing, in contrast to feature specialisation, provides a specialised processing channel to act on the latent representation, where channels are defined by unique and learnable parameters that process specific information/concepts in the domain. By adding a separate and unique information processing channel for each of the different concepts being modelled, the DNN has the additional capacity to find meaningful representations. Instead of enforcing the model to find a joint representation and information processing function to suit all tasks, the model can learn meaningful concepts that are added to the original and generalised representation for the final classification or regression prediction.

Definition 6.1.1. A channel is a distinct *pathway* of computation, implemented with unique weights and biases.

In specialised information processing, there are two types of information processing channels. The first is the generalised processing channel. This type of channel is created as if there were no unique specialisation through related tasks. The second representation is the specialised processing channel. This second channel learns unique parameters to process specific concepts in the domain, and there can be one or many specialised information channels, one for each concept being modelled. These specialised information channels, though distinct from the generalised representation used to create the main task, helps the model improve in performance with this main task through the combination of channels. For example, in the case study of quantum chemistry, the generalised processing channel computes messages from all atoms, while the specialised processing channels only compute messages from certain atoms bonded with particular types of bonds. Here there is a specialised processing channel for each of the bond types.

Like indirect specialisation, specialised information processing additionally allows for user exploration of the domain concepts with visualisation of the specialised information processing channels. As each channel should model a unique concept in the domain, visualisation of these should provide some insight into the model's learnt understanding of the data.

6.2 Specialised Information Processing in Recurrent Networks

In general, RNNs take a sequence as input, while for each entry in the sequence, the RNN updates an intermediate state representing the cell's memory. For the purpose of classification, this intermediate state or the hidden state, has encoded a representation of the entire sequence that may be classified with some logistic regressor. With specialised information processing, specialised cells are used to learn concepts from the domain, and feed back into the general representing according to the hierarchy of tasks. To demonstrate this method, we use the case study of OGD.

Both OGD base models use recurrent layers, specifically LSTM cells, to accumulate the word embeddings and provide a fixed-sized vector representation of the entire conversation. This fixed-sized vector can then be used for the final projection into the classification space.

LSTMs are composed of multiple probabilistic and differentiable logic gates (Equation 6.1). The LSTM cells use an input, forget, and output gate while storing an internal *memory* into a cell state and hidden state.

$$
\begin{aligned}
\mathbf{i}^{(t)} &= \sigma(W_i \mathbf{x}^{(t)} + U_i \mathbf{h}^{(t-1)}) \\
\mathbf{f}^{(t)} &= \sigma(W_f \mathbf{x}^{(t)} + U_f \mathbf{h}^{(t-1)}) \\
\mathbf{o}^{(t)} &= \sigma(W_o \mathbf{x}^{(t)} + U_o \mathbf{h}^{(t-1)}) \\
\overline{\mathbf{c}}^{(t)} &= \phi(W_c \mathbf{x}^{(t)} + U_c \mathbf{h}^{(t-1)}) \\
\mathbf{c}^{(t)} &= \mathbf{f}^{(t)} \circ \mathbf{c}^{(t-1)} + \mathbf{i}^{(t)} \circ \overline{\mathbf{c}}^{(t)} \\
\mathbf{h}^{(t)} &= \mathbf{o}^{(t)} \circ \phi(\mathbf{c}^{(t)})
\end{aligned}
\tag{6.1}
$$

where $\mathbf{i}, \mathbf{f}, \mathbf{o}, \overline{\mathbf{c}}$ are the input, forget, output, and candidate gate activations used in the creating of a new cell state $\mathbf{c}$ and the hidden state $\mathbf{h}$. W, U are the learnt parameters for each of the logic gates. σ, ϕ are the sigmoid and hyperbolic tangent activation functions. $\circ$ represents the Hadamard product between vectors.

A potential limitation of using this traditional LSTM cell structure is that the cell must accurately model an internal hidden state $\mathbf{h}$ that is useful for predicting the classification of an entire conversation. This is made difficult if there is a lot of important information needed to be encoded into the memory of the cell. For example, if there are several sub-tasks that contribute to the main task that can be expressed in different ways. The LSTM may have difficulty compressing this information into a fixed vector size. Specialised information processing may overcome this challenge by providing a unique processing channel for each of the different concepts. In the case of OG, this is each of the different grooming processes.

In the work of Lu et al. [86], a separate LSTM was used to learn task-specific information on language understanding tasks. However, the information learnt from these specific tasks does not interact with the generalised LSTM cell. In our specialised in-

formation processing method, the generalised representation is created from both the previous generalised representation state, and the representation processed by the specialised processing. We argue this enables more frequently communication between the general and specialised channels that better leverages both low-level representations.

The specialised LSTM closely follows the processes of Equation 6.1, though with its own parameters:

$$
\begin{aligned}
\mathbf{i}_s^{(t)} &= \sigma(W_i^s \mathbf{x}^{(t)} + U_i^s \mathbf{h}_s^{(t-1)}) \\
\mathbf{f}_s^{(t)} &= \sigma(W_f^s \mathbf{x}^{(t)} + U_f^s \mathbf{h}_s^{(t-1)}) \\
\mathbf{o}_s^{(t)} &= \sigma(W_o^s \mathbf{x}^{(t)} + U_o^s \mathbf{h}_s^{(t-1)}) \\
\overline{\mathbf{c}}_s^{(t)} &= \phi(W_c^s \mathbf{x}^{(t)} + U_c^s \mathbf{h}_s^{(t-1)}) \\
\mathbf{c}_s^{(t)} &= \mathbf{f}_s^{(t)} \circ \mathbf{c}_s^{(t-1)} + \mathbf{i}_s^{(t)} \circ \overline{\mathbf{c}}_s^{(t)} \\
\mathbf{h}_s^{(t)} &= \mathbf{o}_s^{(t)} \circ \phi(\mathbf{c}_s^{(t)})
\end{aligned}
$$

At each time-step t the general hidden state takes into account the detection of the OG processes and the general hidden state from the previous timestep (Equation 6.2). At this stage, the model is encouraged to find a relation between the conversation semantic representation and the OG processes.

$$
\begin{aligned}
\mathbf{o}_r^{(t)} &= \sigma(W_r^s \mathbf{x}^{(t)} + U_r^s \mathbf{h}^{(t-1)}) \\
\mathbf{h}^{(t)} &= \mathbf{o}_r^{(t)} \circ \phi(\mathbf{c}_s^{(t)}) + \mathbf{o}^{(t)} \circ \phi(\mathbf{c}^{(t)})
\end{aligned}
\tag{6.2}
$$

The additional activation $\mathbf{o}_r^{(t)}$ modulates the amount of information used from the specialised representation to produce a new hidden state candidate.

To help the specialised channel in learning to detect grooming processes, hints (Definition 2.0.1) are provided to the specialised channel through an auxiliary loss:

$$
\mathcal{L}_{GMM} = \frac{1}{T} \sum_{t}^{T} (W \mathbf{h}_s^{(t)} + b - \mathcal{G}(w^{(t)}))^2 \tag{6.3}
$$

The design of specialised information processing for recurrent networks is not limited to LSTMs, however. This concept can be generalised to add a specialised channel to the Gated Recurrent Unit (GRU) cell. This cell, instead of having a separate cell state and hidden state, fuses these two states together into a single hidden state (Equation 6.4). It also uses fewer parameters as it lacks an output gate, further reducing its probability of over-fitting on smaller datasets [189].

$$
\begin{aligned}
\mathbf{z}^{(t)} &= \sigma(W_z \mathbf{x}^{(t)} + U_z \mathbf{h}^{(t-1)}) \\
\mathbf{r}^{(t)} &= \sigma(W_r \mathbf{x}^{(t)} + U_r \mathbf{h}^{(t-1)}) \\
\hat{\mathbf{h}}^{(t)} &= \phi(W_h \mathbf{x}^{(t)} + U_h(\mathbf{r}^{(t)} \circ \mathbf{h}^{(t-1)})) \\
\mathbf{h}^{(t)} &= (1 - \mathbf{z}^{(t)}) \circ \mathbf{h}^{(t-1)} + \mathbf{z}^{(t)} \circ \hat{\mathbf{h}}^{(t)}
\end{aligned}
\tag{6.4}
$$

where $\mathbf{z}, \mathbf{r}, \hat{\mathbf{h}}$ are the update, reset activations, and candidate hidden state, respectively.

To add a specialised processing channel for this type of cell, a specialised candidate state $\hat{\mathbf{h}}_s$ is created and a relation gate $\mathbf{o}_r$ (Equation 6.5) is used to update the general state from both general and specialised processes.

$$
\begin{aligned}
\mathbf{z}_s^{(t)} &= \sigma(W_z^s \mathbf{x}^{(t)} + U_z^s \mathbf{h}_s^{(t-1)}) \\
\mathbf{r}_s^{(t)} &= \sigma(W_r^s \mathbf{x}^{(t)} + U_r^s \mathbf{h}_s^{(t-1)}) \\
\hat{\mathbf{h}}_s^{(t)} &= \phi(W_h^s \mathbf{x}^{(t)} + U_h^s(\mathbf{r}^{(t)} \circ \mathbf{h}_s^{(t-1)})) \\
\mathbf{h}_s^{(t)} &= (1 - \mathbf{z}_s^{(t)}) \circ \mathbf{h}_s^{(t-1)} + \mathbf{z}_s^{(t)} \circ \hat{\mathbf{h}}_s^{(t)}
\end{aligned}
\tag{6.5}
$$

and condition the generalised hidden state on the specialised candidate state:

$$
\begin{aligned}
\mathbf{o}_r^{(t)} &= \sigma(W_r^s \mathbf{x}^{(t)} + U_r^s \mathbf{h}^{(t-1)}) \\
\mathbf{h}^{(t)} &= (1 - \mathbf{z}^{(t)} - \mathbf{o}_r^{(t)}) \circ \mathbf{h}^{(t-1)} + \mathbf{z}^{(t)} \circ \hat{\mathbf{h}}^{(t)} + \mathbf{o}_r^{(t)} \circ \hat{\mathbf{h}}_s^{(t)}
\end{aligned}
\tag{6.6}
$$

A pictorial representation of the communication between the generalised and specialised channels is shown in Figure 6.1.

6.3 Specialised information processing in Graphs

For graph representations where edges can be used to denote different relational concepts between nodes, specialised information processing can be used to process the messages between nodes in different ways according to the type of edges between nodes. To demonstrate the use of this method, we use the quantum chemistry case study.

The bonds between atoms are an important factor when considering atomic interactions and their contribution to the system's energy. Therefore, when estimating the potential energy of a system as a function of geometry, it may be beneficial to account for the contribution of different Bond Type (BT). Accordingly, Gilmer et al. [190] introduced BT information as edge input features, combined with inter-atomic distance when computing interaction messages (Equation 4.2). This additional feature improved energy estimates over using distance alone.

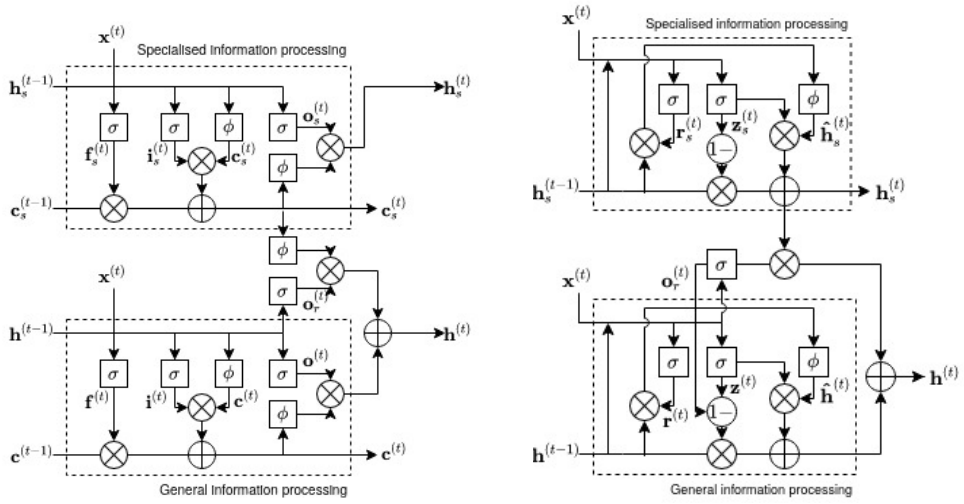


Figure 6.1: *Visualisation of the OG process information processing for LSTM (left) and GRU (right). Activation gates are shown in squares, with addition, subtraction, and multiplication of vectors in circles.*

These encouraging results have inspired us to take this physics integration principle further and to design the information exchange within the GNN to reflect the physics of atomic interactions. Similar to Schlichtkrull et al. [191], specialised communication channels are introduced based on BT[1]. While [191] focused on the production of messages that are specialised to the edge relation, two strategies are explored for specialising either the message production (Equations 4.2 & 4.4), or the message usage in updating node states (Equations 4.3 & 4.5).

6.3.1 Specialised Message Production

Separate messages $\mathbf{m}_{vw}^{r}$ (t or l are omitted for readability) for each type of relation r between nodes (in our application scenario r denotes the BT) are produced by relation-specialised mechanisms, i.e. with relation-specialised M_r (MPNN) and R_r^l (SchNet). In the case of a perceptron-based M_r, this is equivalent to the method of [191]. The new message production equations for MPNN and SchNet become respectively:

$$\mathbf{m}_v^{t+1} = \alpha \sum_{w \in \mathcal{N}_v} M\left(\mathbf{h}_w^t, \mathbf{x}_{vw}^e\right) + (1-\alpha) \sum_{r \in \mathcal{R}} \sum_{w \in \mathcal{N}_v^r} M_r\left(\mathbf{h}_w^t, \mathbf{x}_{vw}^e\right) \tag{6.7}$$

$$\mathbf{m}_v^{l+1} = \alpha \sum_{w \in \mathcal{N}_v} \mathbf{h}_w^l \circ R^l\left(d_{vw}\right) + (1-\alpha) \sum_{r \in \mathcal{R}} \sum_{w \in \mathcal{N}_v^r} \mathbf{h}_w^l \circ R_r^l\left(d_{vw}\right) \tag{6.8}$$

with α modulating the strength of the relation-specialised message production with regards to the original generic message production. $\mathcal{R}$ is the set of relation types.

[1] BT is predetermined using RDKit (`https://www.rdkit.org/`), and Antechamber [192], for molecules with Canonical SMILES representation and others, respectively.

In this application, it is the set of bond types, that may include a 'no bond' element for pairs of atoms that do not share a bond but that we still want to consider (fully connected graph), as in SchNet. $\mathcal{N}_v^r$ is the set of neighbour nodes that share a relation of type r with node v.

6.3.2 Specialised Node Update

Another possible strategy for specialised interaction is to produce generic messages $\mathbf{m}_{vw}$ using a single message function but to handle the messages in a specialised manner when updating node states. We explore following different implementations:

Specialised weighting of the messages

A simple approach to differentiating the messages coming from nodes of different relation types is to weight them by a relation-specialised and learnable (scalar or vector) weight λ_r:

$$\mathbf{m}_v = \alpha \sum_{w \in \mathcal{N}_v} \mathbf{m}_{vw} + (1-\alpha) \sum_{r \in \mathcal{R}} \lambda_r \sum_{w \in \mathcal{N}_v^r} \mathbf{m}_{vw} \tag{6.9}$$

In the case of a perceptron-based message function M (MPNN), this approach is equivalent to the basis-decomposition regularisation proposed by [191] as a means to reduce the number of learnable parameters associated with specialised message production.

Specialised update functions

The second approach is to implement separate relation-specialised update functions and sum their contributions. The node update equations for MPNN and SchNet thus become respectively:

$$\mathbf{h}_v^{t+1} = \alpha U\left(\mathbf{h}_v^t, \mathbf{m}_v^{t+1}\right) + (1-\alpha) \sum_{r \in \mathcal{R}} U_r \left(\mathbf{h}_v^t, \sum_{w \in \mathcal{N}_v^r} \mathbf{m}_{vw}^{t+1}\right) \tag{6.10}$$

$$\mathbf{h}_v^{l+1} = \mathbf{h}_v^l + \alpha V^l\left(\mathbf{h}_v^l, \mathbf{m}_v^{l+1}\right) + (1-\alpha) \sum_{r \in \mathcal{R}} V_r^l \left(\mathbf{h}_v^l, \sum_{w \in \mathcal{N}_v^r} \mathbf{m}_{vw}^{l+1}\right) \tag{6.11}$$

In the case of MPNN (Equation 6.10), where the update function is implemented by a GRU unit, we experiment with three different ways of specialising U_r:

Option 1) A separate GRU cell, one per relation type, to implement the different U_r.

Option 2) a single GRU cell, so that all relation channels share the GRU's internal state. The GRU's $\mathbf{W}_z$, $\mathbf{W}_r$ and $\mathbf{W}_h$ weight matrices contain weights that are specialised for

the different relation types. In practice, this amounts to concatenating the different messages from each relation type before providing them to the GRU cell:

$$\begin{bmatrix} \sum_{w \in \mathcal{N}_v^1} \mathbf{m}_{vw} \\ \vdots \\ \sum_{w \in \mathcal{N}_v^{|\mathcal{R}|}} \mathbf{m}_{vw} \end{bmatrix} \tag{6.12}$$

Option 3) a single GRU cell with concatenated messages from different relation types, but the number of free GRU parameters are reduced in $\mathbf{W}_z$, $\mathbf{W}_r$ and $\mathbf{W}_h$ by sharing them between relation channels:

$$\mathbf{W}_{z/r/h} = \begin{bmatrix} \mathbf{Q}_{z/r/h} \\ \vdots \\ \mathbf{Q}_{z/r/h} \end{bmatrix} \tag{6.13}$$

with $\mathbf{Q}_z$, $\mathbf{Q}_r$ and $\mathbf{Q}_h$ being matrices.

6.4 Experiments

The first experiments examined how effective the specialised communications were for detecting OG in chat logs. This experiment used the methods of Section 6.2 to integrate the prior knowledge groomer communicative processes with specialised RNN processors.

The type of recurrent cell was varied for each of the base models, and whether specialised information processing was used. The results are shown in Table 6.1. In this table, the specialised information processing methods are compared with their non-augmented counterpart to detect OG. Specialised communication channels or not, we observed the GRU cell generally performed better than LSTM on this dataset, perhaps due to less over-fitting on the small number of positive examples as the GRU cell has fewer parameters than the LSTM cell. When a specialised processing channel was added to the recurrent cell, the classifier was more precise with its predictions, indicating the specialised channel was enabling the DNN to attend to the labelled groomer processes. As not all positive cases of grooming were labelled with groomer processes, this also has had an effect of slightly reducing the recall of the classifier by possibly making the DNN reliant on the annotations. However, with some exceptions, we have generally seen a higher AUPR, F_1, and $F_{0.5}$ metrics for the specialised information processing. These encouraging results prompt us to use this prior integration strategy when building our final OG detector in Chapter 9.

Our next experiment has evaluated the methods presented in Section 6.3. The individual effects of the different knowledge integration methods in comparison to non-augmented base models were evaluated on Augmented QM9. Results of (non-augmented) base models are provided in rows 1 and 8 of Table 6.2. When compared against these

Table 6.1: *Performance of both base models OGD-R and OGD-T when using Specialised information processing with LSTM and GRU cells. Metrics are compared against their baseline cells.*

Base Model	Recurrent Cell	Type	Precision	Recall	AUPR	F_1	$F_{0.5}$
OGD-R	LSTM	Baseline	0.867	**0.794**	0.867	0.829	0.851
		Specialised	**0.912**	0.789	**0.899**	**0.846**	**0.884**
	GRU	Baseline	0.875	0.772	**0.893**	0.820	0.852
		Specialised	**0.899**	**0.775**	0.889	**0.832**	**0.871**
OGD-T	LSTM	Baseline	0.900	**0.871**	**0.940**	0.886	0.894
		Specialised	**0.954**	0.835	0.936	**0.891**	**0.928**
	GRU	Baseline	**0.943**	0.847	0.936	**0.892**	**0.922**
		Specialised	0.914	**0.863**	**0.938**	0.888	0.903

base models, and considering only the best performing methods for specialised interactions, all integration strategies of domain knowledge tend to have improved the energy estimation and/or finding stable geometries. The impact of BT information was the strongest for both architecture types. This confirms [167] 's observation that BT is relevant for estimating energy. This suggests that specialised interactions better capture the physics of atomic interactions. For MPNN, we found the specialised update function to be the best performing method of integrating different BTs into the model, while the specialised weighted messages are better for SchNet. This observation comes from the lower AE metric. However, after evaluating the DSG metric, we have seen that for both architectures, the specialised update function was the best performer. This appears to coincide with the effect shown in Figure 5.4 in which the best predictor of the stable configuration was not necessarily the model with the lowest residual errors.

In our implementations of the specialised processing for different BTs, we included an learnt α (trained via back-propagation) value that determines how much information from the specialised process to include in the node state. The α value is learnt through the back-propagation of the network while training to estimate energy. Interestingly, the model learnt a similar α value for each of the BT specialisations, indicating that it perhaps was able to find an intrinsic relationship in the specialised messages and general messages that does not change with how specialised messages are created.

For the previous experiment, the α value was learnable, and the model can optimise the modulation between specialised and general messages. To evaluate the effectiveness of the specialised messages, α was set to 0 and the models were re-trained. Table 6.3 presents this further assessment of the different specialised interaction strategies. While the AE was lower for these experiments, the DSG and RE metric was generally higher. With a learnable α we found that all metrics are usually improved, indicating there is a benefit to allowing the network to find a suitable modulation between specialised and general messages.

Given these results, when combing augmentations for this case study, we shall continue a learnable α with Equation 6.10 (implementation 2) for MPNN and Equation 6.9 for SchNet.

Table 6.2: *Impact of each domain knowledge integration strategy on MPNN (top) and SchNet (bottom) for Augm. QM9. Results are in the format: mean (std). The specialised interaction methods that optimise at best energy and geometry are highlighted in bold for each GNN.*

	Strategy	AE	RE	DSG
	MPNN base model with no BT information	0.242 (1.318)	0.0029 (0.015)	0.034 (0.074)
Specialised interactions	(Eq. 6.7) ($\alpha = 0.624$)	0.272 (1.293)	0.0034 (0.014)	0.051 (0.080)
	(Eq. 6.9) scalar λ_r ($\alpha = 0.628$)	0.122 (0.431)	0.0016 (0.005)	0.032 (0.057)
	(Eq. 6.9) vector λ_r ($\alpha = 0.627$)	0.105 (0.502)	0.0013 (0.005)	0.050 (0.049)
	(Eq. 6.10) impl. 1) ($\alpha = 0.615$)	**0.106 (0.253)**	**0.0014 (0.003)**	**0.023 (0.048)**
	(Eq. 6.10) impl. 2) ($\alpha = 0.635$)	**0.073 (0.170)**	**0.0010 (0.002)**	**0.030 (0.048)**
	(Eq. 6.10) impl. 3) ($\alpha = 0.618$)	0.131 (0.436)	0.0017 (0.005)	0.025 (0.055)
	SchNet base model	0.038 (0.037)	0.0005 (0.0005)	0.020 (0.031)
Specialised interactions	(Eq. 6.8) ($\alpha = 0.734$)	0.036 (0.035)	0.0005 (0.0005)	0.022 (0.032)
	(Eq. 6.9) scalar λ_r ($\alpha = 0.529$)	**0.020 (0.018)**	**0.0003 (0.0002)**	**0.025 (0.032)**
	(Eq. 6.9) vector λ_r ($\alpha = 0.570$)	0.024 (0.028)	0.0003 (0.0003)	0.034 (0.034)
	(Eq. 6.11) ($\alpha = 0.727$)	0.031 (0.032)	0.0004 (0.0004)	0.015 (0.028)

Table 6.3: *Impact of each specialised interaction strategy for MPNN with $\alpha = 0$ (no generic interaction used) on Augmented QM9. Results are in the format: mean (std) over molecules*

Strategy	AE	RE	DSG
(Eq. 6.7)	0.171 (0.513)	0.0022 (0.006)	0.046 (0.055)
(Eq. 6.9) scalar λ_r	0.067 (0.141)	0.0009 (0.002)	0.037 (0.048)
(Eq. 6.9) vector λ_r	0.121 (0.295)	0.0016 (0.004)	0.044 (0.050)
(Eq. 6.10) impl. 1)	0.114 (0.142)	0.0015 (0.002)	0.048 (0.056)
(Eq. 6.10) impl. 2)	0.182 (0.582)	0.0023 (0.006)	0.047 (0.080)
(Eq. 6.10) impl. 3)	0.108 (0.121)	0.0012 (0.002)	0.046 (0.074)

Using specialised information processing may also provide an opportunity to explore the use of visualisations for DNNs' learnt 'understanding' of atomic principles. In Figure 6.2, the different BT messages were extracted during the specialised update function of Augmented-MPNN (Equation 6.10 implementation 2). The t-Distributed Stochastic Neighbour Embedding (t-SNE) [50] algorithm was used to reduce the dimensionality of the messages from the initial 73 to 2 dimensions for the purpose of visualisation. From this visualisation, we observed the different messages of different BTs lay in a different space, meaning that messages of different types were sufficiently different for the TSNE visualisation to place them in different locations.

For each of the messages, the colour value was mapped to the normalised contribution of each atom during the readout function of MPNN. During this readout function, the contribution of each atom was summed to create the final energy estimation for the molecule. Therefore, the observation is that most single-bonded messages contributed to a higher energy value, while double and triple bonded messages have reduced the energy during the readout phase. From this visualisation, we observed MPNN, through prior knowledge augmentations, was correctly demonstrating physical properties.

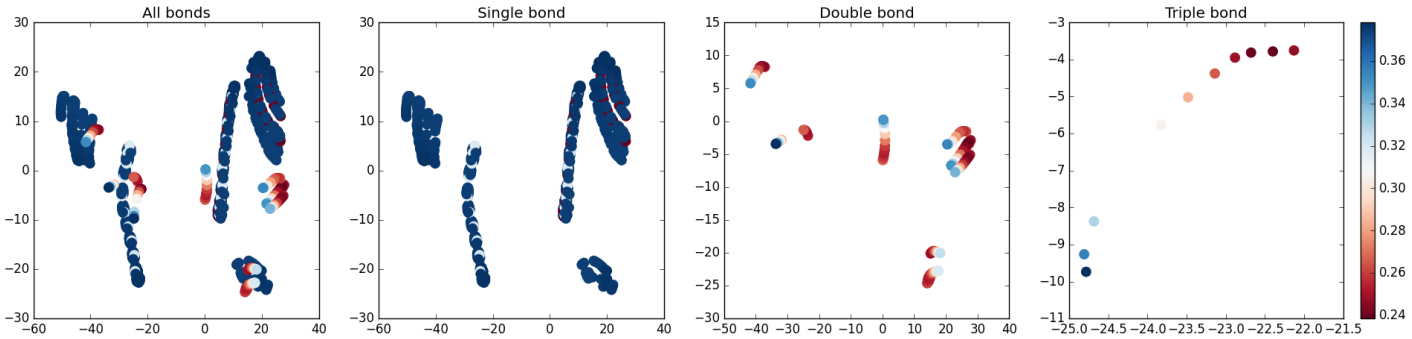


Figure 6.2: *One TSNE visualisation of different BT messages produced by Augm-MPNN for first 1000 molecules in the testing data. For clarity, the different BT messages (single, double, triple bonds) are split into three separate plots, with a fourth plot showing all BT messages together in a single plot. Messages were extracted from the output of the final BT specialised update function (Equation 6.10 implmentation 2) before the final scalar energy prediction is made.*

This visualisation was uniquely driven by the interrogation of visualising the strategies of prior knowledge integration. In this case, we have seen the double and triple bonds contributes to the node state that produces a lower energy state more than single bonds.

6.5 Chapter Summary

In this chapter, we discussed how specialised information processing helped DNNs internalise and represent domain knowledge. Specialised information processing provides specialised computations resembling the concepts within the domain. This method is implemented by unique parameters of the network to act on the representation as computed by the general processing channel. These specialised channels feedback into the general representation.

For OG detection, we used specialised channels of OG processes for recurrent cells that contribute back into the generalised representation. The specialised representation was modelled in a supervised manner, where the expert's annotation of groomer processes was used to update the specialised channel's parameters during back-propagation. The specialised channel can then additionally output a visualisation of the predicted groomer processes as in Figure 5.3. For quantum chemical property predictions, the GNNs have been augmented by combining the general messages and their specialised BT messages to update the node states in graphs. Unlike the specialised recurrent cells, an auxiliary loss was not used to provide hints the specialised BT processing channels. Despite this, the specialised BT processing channels helps estimate the system energies without the need for more expert annotations.

The contributions of this chapter are:

- A theoretical formulation of specialised information processing of distinct channels of concepts, where the addition of separate and unique information process-

ing channel for the concept within the domain being modelled. For example, in OG detection, a specialised channel of the OG processes was used. In quantum chemical systems, these specialised information processing channels represented the different chemical BTs.

- A method to incorporate specialised information process in recurrent networks. This method can generalise to both LSTM and GRU cells, and has been tested on the OG detection case study.
- A method to adapt graph-based DNNs with specialised information processing. The estimation of chemical system energies was used to test this method.

The key points from this chapter are:

- Specialised information processing channels can be added to DNNs. The purpose of these processing channels was to allow the DNN to learn specialised parameters to represent and improve the latent representations of the concepts without competition with the downstream tasks.
- Hints with auxiliary losses can help the DNN to create specialised information processing channels for the domain concepts.
- A method was shown to apply specialised information processing to recurrent cells (LSTM and GRU). When these specialist cells were used to model OG processes, there was an improved performance of detecting OG, while also providing an additional means of visualisation.
- Another method has been explained to create specialised channels for GNNs. To test this method the quantum chemical case study was used. In these tests, specialised information processing was used to represent each of the different BTs present in molecular and crystalline systems.
- We have seen an improved performance when using specialised information processing channels as opposed to only using the generalised representations.

Chapter 7

Attention on Data

DNNs typically require less feature engineering than other ML techniques [193], where the model automatically learns the importance and relations between features. However, when starting the learning process, the DNN must process and consider each of the input features with equal or random weighting, despite any prior known importance of features. Therefore, there is a unique opportunity to encode the expert's domain knowledge of the importance of certain features for the classification or regression task.

Intuitively, this feature importance corresponds to the attentiveness of the DNN, where the model learns to *focus* or pay attention to selective parts of the input to produce its output. For instance, a simple MLP may increase the weight of certain features, while the weights of other features remain relatively close to 0. In NLP, a recurrent network may learn to attend to certain words by increasing the activation of it's input gate during the context of its occurrence. Or in image recognition, where the presence of certain patterns is more conducive for prediction of one class over other classes. Despite the DNN already performing an implicit form of attention during the process of learning, the attentiveness of DNNs can be improved with additive attention mechanisms, thereby improving its performance in many domains.

An Attention mechanism was first introduced by [96], where the performance on language translation tasks is improved for Encoder-Decoder architectures. The authors hypothesise that the architecture is limited by the amount of information that can be passed from the Encoder to the Decoder in a fixed-sized vector between these two submodels. With their proposed attentive mechanism, the important features relevant to language translation can be further enhanced in this vector and therefore providing more useful information to Decoder.

Though originally designed for language translation, attention has become one of most important concepts in DL, with applications ranging from NLP [37, 103, 1, 104], im-

age recognition [102, 105], speech recognition [107, 95], and reinforcement learning [108, 109].

To specify feature importance during the training process, attention can be used to help the DNN to *focus* on particular features or parts of the input, as identified by the prior expert knowledge, that may help with the classification or regression task. In this chapter, the method of how one may use prior domain knowledge to provide added attention on data is discussed. Two methods are introduced to integrate attention on data using both recurrent and the attention mechanisms of transformer models. These methods were tested using the OG detection case study where the presence of OG processes was used to increase focus of the DNN on particular phrases, indicative of grooming strategies, present in the chat logs.

7.1 Background on Attention

Certain features are more important than others. For instance, in a language understanding tasks, certain words are more relevant to understanding of what meaning the sentence is attempting to convey, other words such as *stop-words* may be omitted but yet the meaning is still understood. In the task of image recognition, the presence of certain objects in an image is more relevant to determine what class label this image has – more relevant than the background. To incorporate this notion of relevance, attention mechanisms are designed to learn feature importance and shift the focus of DNNs towards these features.

The concept of attention in DNNs is often inspired by the human psychological processes [98, 194, 195, 196, 197]. Humans have the ability to attend to more features in sounds and images. For example, a human does not process an entire image, but instead attends to selective parts of the scene and combines this information in order to make an assessment [98]. Moreover, they would recognise an object in a scene as well as the object isolated from the surrounding environment [198]. Likewise, a DNN will learn to give some features certain weight, but will, however, evaluate the entire scene.

For tasks such as sentiment analysis with long sequences and image caption generation, compressing large amounts of information into a fixed-sized vector can be challenging, and some information will be lost in the process [114]. To overcome this challenge, Bahdanau et al. [96] allows the Decoder network of an Encoder-Decoder architecture to access the entire sequence of hidden states from the Encoder. By introducing attention weights over these hidden states, an optimised and more informative fixed-sized vector can be computed as input for the Decoder network. Luong et al. [1] provides two approaches of this method with global and local attention. Global attention, like [96], applies attention weights over all state positions, while local attention considers a subset of positions. From these methods, both have been successful in improving the performance in language translation tasks.

The method of attention can be generalised as Equation 7.1 in which the optimised feature vector z is created from the weighted sum of the hidden states of all time-steps

t and the learnt attention weights a.

$$z_i = \sum_{t=1}^{T} a_i^{(t)} h^{(t)} \tag{7.1}$$

where $\sum a_i = 1$.

Scaled Dot-Product Attention is an alternative method of computing attention that is used in the state-of-the-art Transformer architectures [37] and builds on the dot-product score method of [1] by adding a scaling factor to account for very small gradients during training. In this form of attention a (Equation 7.2), three representations are used to compute the optimised vector: a **K** or *key* from a linear projection hidden state of the Encoder; **Q** or *query* from a linear projection the hidden state of the Decoder; and **V** or *value* another linear projection of the same hidden state of the Encoder.

$$a(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{n}})\mathbf{V} \tag{7.2}$$

where n is the length of the sequence. $\mathbf{Q}\mathbf{K}^\top$ calculates the self-attention score for the relevance of each word for every other word. This score is scaled by the square root of the length of the sequence and Softmax is applied to ensure the weights sum to 1. The final attention scores are obtained by multiplying these weights against **V**.

Attention helps DNNs to attend to important features when performing the compression process of converting an entire sequence of states into a fixed-length vector, resulting in a vector that are richer and more descriptive for their intended task. In addition to just improving performance, the learnt attention weights in NLP provides insight into the importance of words for the natural language understanding task [199]. Such an example is constructing an $N \times N$ matrix with English words as columns and German words as rows (Figure 7.1). Each entry in this matrix shows the attention weights between two-word pairs and can be interpreted as the relevance of each English word for the prediction of each translated German word [96, 1].

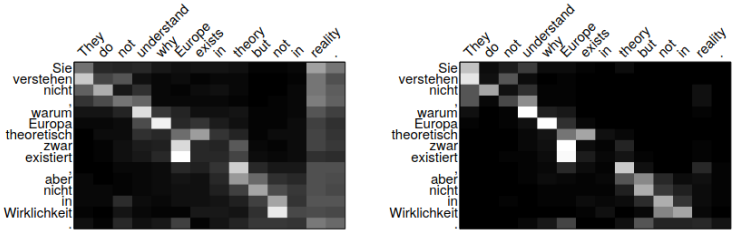


Figure 7.1: *Visualisation of attention weights from [1] for local attention (left) and global attention (right). Entries in these matrices demonstrate the importance of each English word (column) for its German translation (row).*

With attention on data, one can also avoid situations where the DNN is using features that occur in the data due to misrepresentation or errors, creating a classifier that, although performances with high accuracy on the test data, is not useful for its intended task. One prime example is in the *husky-vs-wolf* task [44], where the DNN trained to recognise the difference between an image of a husky and a wolf, would often use the presence of a snowy background to determine if the image contains a husky. This example highlights the ease with which the DNN can attend to spurious features in data. Furthermore, the analysis of feature importance is one method by which practitioners may understand some decision processes made by a trained DNN [200].

7.2 Stimulating Attention

We propose to use this idea of attention as a method for shifting the focus of DNNs to relevant features or inputs as determined by the domain expert. Like some of the existing methods for attention, this prior knowledge integration of domain knowledge is not only applicable for NLP, but may also help in making DNNs in domains such as image or speech recognition more attentive to prior knowledge.

In principle, we would like to allow the DNN to continue to learn attention in an unsupervised manner, as the method has already proved to be successful for many domains. To help the DNN to succeed in recognising important features, especially early on in the learning process, however, we may augment this unsupervised attention through a *direct* action that boosts the attention weights at strategic inputs using the prior knowledge of the expert. This direction of attention helps to highlight the important features for the DNN, even if its unsupervised attention has not yet learnt this importance.

One may use an alternative method of stimulating attention following the design principle of Indirect Specialisation (Chapter 5). Instead of *directly* stimulating the attention, one may *supervise* the process of learning attention and provide hints as to the important features. This auxiliary task may still shift the focus of attention like the excitation method but requires selecting an appropriate task weighting to ensure all tasks are not conflicting, which may result in reduction the performance of the DNN.

While this method of supervising attention uses an auxiliary task, it is not indirect in the same way as indirect feature specialisation that acts indirectly on learnt features through the auxiliary output task. Rather, supervised attention directly acts on attention weights which are considered in the loss function. Nevertheless, both indirect feature specialisation and supervised attention can be implemented via auxiliary tasks, highlighting the generalisability auxiliary tasks has in helping to train DNN to great effect.

Using OG detection as a case study for integrating stimulated attention, two implementations methods of stimulating attention were created. These implementations were designed to increase the OG detectors' focus on particular phrases used in the context of grooming. These phrases have been identified by the expert annotator as constituting of a grooming communication process. By using these methods of stimulating attention on these grooming processes, the DNN may be more sensitive to

the occurrence, expression, and context of these processes, despite their implicit and unassuming verbiage.

7.2.1 Direct Attention Stimulation

For many applications of DNNs in NLP, these DNNs use attention mechanisms weights to attend to important words/features. In transformer architectures, attention energy $\mathbf{e_t}$ are explicitly computed for each word at position t. In recurrent cells, update gates in effect implement an implicit attention mechanism that selects important input features to update the latent cell state of the RNN. Therefore, the RNN cell input/update activation i_t may be considered an attention weight and thus our methods will also be applicable.

To compute an attention energy $\mathbf{e_t}$ for word at position t, it may be stimulated during training to guide the DNN's attention on occurrences of OG processes[1]. Using the prior knowledge of the grooming processes via the GMMs, we have directly augmented the attention of the DNNs. We show two examples of how direct attention stimulation can be implemented for different forms of attention: self-attention, and input-gate activation in RNNs. We use this method to stimulate the activation of both self-attention and recurrent modules in locations containing OG processes indicated by $\mathcal{G}$, encouraging the DNN to recognise the contexts of OG processes to improve the information carried in its hidden state during sequence modelling.

Direct stimulation of attention mechanisms inspired by Derakhshani et al. [112] that augmented CNN's activations to speed up localisation learning in images. This work inspired us to augment the attention weights by the grooming communication processes regulated by the attention energies:

$$\tilde{\mathbf{e}}_\mathbf{t} = \mathbf{e_t} + \mathcal{G}(t)\, \mathbf{e_t} \tag{7.3}$$

In OGD-T, there are several attention energies for each word. In this circumstance, the direct stimulation is applied independently over each attention energy for each word.

In addition to Equation 7.3, we provided an alternative direct attention stimulation (Equation 7.4) with an unregulated prior on $\tilde{\mathbf{e}}_\mathbf{t}$ that may be more useful during the start of training when the attention weights are random. The effectiveness of this modification will be tested in Section 7.3. While in this work we consider Equations 7.3 and 7.4 to be separate methods, future work could explore using Equation 7.4 at the start of the training process, before adding the regulation term after the unsupervised attention has had time to learn from the data.

$$\tilde{\mathbf{e}}_\mathbf{t} = \mathbf{e_t} + \mathcal{G}(t). \tag{7.4}$$

Direct input-gate activations stimulation of recurrent cells. The input/update gate

[1] No annotation of OG processes (i.e. GMM) is required at testing time.

activation $\mathbf{i_t}$ of the RNN is augmented during OG processes, indicated by a peak of $\mathcal{G}$, through:

$$\tilde{\mathbf{i}}_\mathbf{t} = \mathbf{i_t} + \frac{\mathcal{G}(t)}{D} \sum_{d=1}^{D} \mathbf{h_t^d} \, . \tag{7.5}$$

The D components of $\mathbf{h_t^d}$ of the hidden state are averaged, by analogy to the work of [112] that averaged over all channels of a CNN's activation map. In addition, $\mathcal{G}(t)$ is scaled by the dimensionality D of the hidden state.

7.2.2 Supervised Attention Stimulation

An alternative (or complement) to directly augmenting existing attention weights is to supervise the attention weights of the DNN. Like the previous method of directly stimulating the attention of DNNs, the method of supervised stimulation can apply to both self-attention and RNN cells.

Supervision of self-attention by the sum $\mathcal{G}$ of GMMs used as ground-truth distribution of the salient locations and attention energies:

$$\mathcal{L}_{\text{attention}} = \frac{1}{T} \sum_{t=1}^{T} (\mathbf{e_t} - \mathcal{G}(t))^2 \, , \tag{7.6}$$

with T the length of messages from both users.

Nguyen and Nguyen [97] employ a similar tactic to encourage their DNN to learn similar attention weights. For their method, the occurrence of single and low-level target words are used as prior knowledge of the classification. Using a Gaussian distribution centred on this target word, the authors eliminate the DNN learning binary attention values, where the attention has the value 1 at the target word, and 0 elsewhere. Similar to [97], our work used the sum of GMM, but instead of a single target word, this GMM highlights seven higher-level OG processes.

Supervision of input-gate activations by minimising the loss between the average input/update gates' activation i_t and the combined GMMs:

$$\mathcal{L}_{\text{stimulation}} = \frac{1}{T} \sum_{t=1}^{T} (\mathbf{i_t} - \mathcal{G}(t))^2 \, ; \tag{7.7}$$

These two methods of augmenting the attentiveness of the DNN may be both employed at the same time. For instance, the attention energies of self-attention may be directly augmented by G while also being supervised by the auxiliary task using the same prior. Through this process, the self-attention energies may benefit more during inference (where the annotations to compute G are not available) than either method applied singularly.

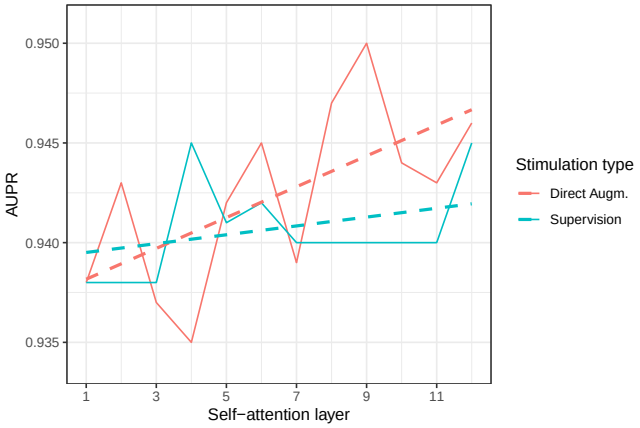


Figure 7.2: *AUPR results of OGD-T with direct augmentation and stimulation using different layers of self-attention. Performance is shown by solid lines and dotted lines are the trend lines.*

7.3 Experiments

To better understand how the direct stimulation affects the self-attention of OGD-T we varied the self-attention layer to which the stimulation methods were applied. The results of this experiment are shown in Figure 7.2. Though there were very small fluctuations in the performance of the model from using one layer to the next, we observed a general trend of increasing performance when using the higher layers. The increasing trend was more noticeable for direct stimulation than for supervision perhaps due to the direct augmentation being more direct with its attention processes. Using higher layers for attention may correspond with the higher level of OG processes where instead of the occurrence of single target words, the OG processes operate at a higher cognitive level. As we have seen generally better performance, though not significantly, improvements, when using the highest layers of self-attention for this method, we were motivated to continue to use the final layer of attention to applying both supervision and direct stimulation.

To compare all stimulation strategies, they were added to the base models (OGD-R and OGD-T) independently, and combined together to understand their interaction. Both of these base models include an unsupervised attention mechanisms to optimise the hidden states of the conversation. OGD-R uses the unsupervised attention of [1], and OGD-T uses self-attention layers of [100]. For this experimental setup, XLNet were used as the self-attention as it is the best performing transformer for our data. LSTM cells were used to test the stimulated RNN method in both models, but the method can also apply to various forms of recurrent cells, e.g. GRU [189]. The stimulated attention methods were applied to the final layers of the mechanisms, i.e. the final layer of XLNet's self-attention and the final LSTM layer.

The results are presented in Table 7.1. All methods of augmenting attention using knowledge of OG processes improved the performance of the DNNs. This has demonstrated that focusing the DNNs' attention on the language associated with OG processes does help capture the subtleties of grooming language.

Table 7.1: *Impact of each CL knowledge integration on OG classification. Bold are improved results with respect to no augmentation, i.e. base models.*

Model		Strategy	Precision	Recall	AUPR	F_1	$F_{0.5}$
OGD-R	No augmentation		0.867	0.794	0.867	0.829	0.851
	Unsupervised attention	supervision	0.839	0.804	**0.879**	0.821	0.832
		direct stim. (Eq. 7.3)	0.822	0.817	**0.877**	0.820	0.821
		direct stim. (Eq. 7.4)	0.870	0.808	**0.906**	**0.838**	**0.857**
		supervision+direct stim. (Eq. 7.3)	0.859	0.819	**0.897**	**0.838**	0.851
		supervision+direct stim. (Eq. 7.4)	0.929	0.741	**0.908**	0.824	**0.884**
	Last LSTM layer	supervision	0.924	0.752	**0.891**	0.829	**0.883**
		direct stim.	0.856	0.797	0.863	0.825	0.843
		supervision+direct stim.	0.906	0.781	**0.901**	**0.839**	**0.878**
OGD-T	No augmentation		0.900	0.871	0.940	0.886	0.894
	Last self-attention layer	supervision	0.919	0.862	**0.943**	**0.890**	**0.907**
		direct stim. (Eq. 7.3)	0.894	0.885	**0.945**	**0.889**	0.892
		direct stim. (Eq. 7.4)	0.916	0.866	0.940	**0.891**	**0.906**
		supervision+direct stim. (Eq. 7.3)	0.891	0.881	**0.941**	0.886	0.889
		supervision+direct stim. (Eq. 7.4)	0.918	0.862	**0.941**	**0.889**	**0.906**
	Last LSTM layer	supervision	0.938	0.857	**0.944**	**0.896**	**0.921**
		direct stim.	0.896	0.896	**0.944**	**0.887**	0.892
		supervision+direct stim.	0.960	0.846	**0.945**	**0.899**	**0.935**

When exploring the attention energies on the test set for (non-augmented) OGD-R, we observed that the contexts the model learnt to focus on are not related to our labelled instances of OG processes: the average (std) attention energy for these instances is 0.0009 (0.0002), lower than energy across all conversations at 0.0016 (0.0128). A similar observation is made for OGD-T, where tokens' energies are obtained from the last self-attention layer similarly by Sood et al. [201] through retaining the max pairwise energy for each token (row) normalising by the sum of retained energies. This was done for each attention head before averaging across heads. The resulting average (std) energy for our instances of OG processes was 0.110 (0.072), slightly lower than the energy across all conversations at 0.120 (0.088). Thus, neither models were able to discover on their own the sub-goals the CL analysis of Lorenzo-Dus et al. [29] identified and their associated language. This knowledge is, therefore, an added value for the models, as also demonstrated by the improved results.

In comparing the method of direct stimulation with and without regulation by learnt attentions (Equations 7.3 and 7.4), we observed the method generally performing better without the regulation term (Equation 7.4), possibly due to the random attention weighting during the start of training. This improvement was seen both with direct stimulation applied singularly and when it was combined with supervision.

The two stimulation strategies seem roughly equally helpful at focusing the DNN's attention and capturing the subtleties of grooming language. In addition, improvements are more consistent for AUPR and precision (and consequently $F_{0.5}$), thanks to fewer false positives. This reduction in false positives may be due to a more straightforward distinction of OG conversations from neutral but sexually-oriented ones.

In Figure 7.3, we show the confusion matrices for the base models and the combination

of supervision and direct stimulation in LSTMs. Here we observed a reduction in the number of false-positive predictions made by the DNN if the attention augmentations are applied during training. This was true for both base models. For OGD-R we have seen a 33.66% reduction in the number of false-positives, while for OGD-T there was a 49.14% reduction.

Furthermore, these cases of false-positives appeared to be somewhat sexually explicit in nature. Using the presence of sexual wording from the vocabulary list of [202] as a criterion for determining sexually explicit conversations, the number of false-positives with this explicit behaviour decreased from 131 of the OGD-R base model to 97 using augmented attention (25.95% reduction). While for OGD-T, the base model flags 76 false-positive sexually explicit conversations, whereas it only produced 36 sexually explicit false-positives with the augmentation (52.63% reduction). This suggested the attention was shifting the focus away from misleading sexual language to more implicit and discrete strategies of groomers.

While there is a reduction in the number of false-positives created by the DNN, we also experienced a small increase in the number of false-negatives as a result of the augmentation. However, these increases are far smaller than the reduction in false-positives. For example, while we observed a 33.66% reduction in false-positives, the false-negatives only increased by 3.55% for OGD-T, likewise for OGD-R where the increase due to augmentation is 6.3%.

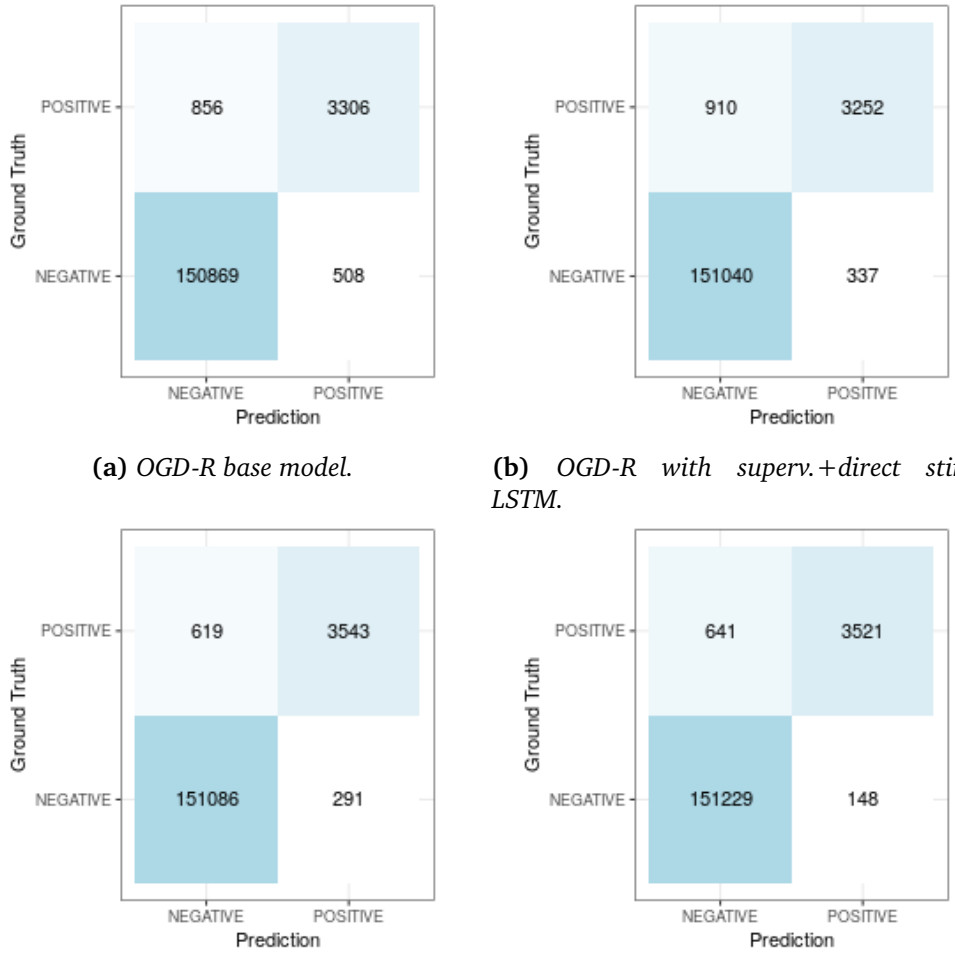
Finally, we observed both self-attention and LSTM input gates, combining the supervision and direct stimulation approaches provided better results than using them individually. This suggested that these two processes supported each other during optimisation. Indeed, improved DNN's attention (expressed in $\mathbf{e_t}$ and $\mathbf{i_t}$) from direct stimulation may assist with the supervised attention task. In addition, improved attention from supervision may also reinforce the direct augmentation and work at its best.

7.4 Conclusion

For classification or regression, different features have a different order of importance. The role of the DNN is to learn this feature importance and weight them appropriately. However, instead of solely relying on the DNN to find important features in data, and if the importance is already known, an expert may integrate this known knowledge via stimulation of attention. This stimulation is designed to improve the model's focus and shift focus away from potentially misleading or erroneous features.

To demonstrate the use of stimulation of attention, the case study of OG detection was used. In this study, the OG processes (collocation of words) was used to promote the feature importance in the DNN during training. This increase of feature importance was integrated with two methods: the first was via the stimulation or direct stimulation of attention, and the second was a supervised auxiliary task.

We observed that attention is a useful strategy of improving DNN performance and reducing the number of false-positives due to potentially misleading features. Fur-



(a) *OGD-R base model.*

(b) *OGD-R with superv.+direct stim. LSTM.*

(c) *OGD-T base model.*

(d) *OGD-T with superv.+direct stim. LSTM.*

Figure 7.3: *Comparison of confusion plots of OGD-R/OGD-T without prior knowledge augmentations and with supervision+direct stimulation LSTM. Colour intensity of cells denotes the log scale of number of conversations for readability.*

thermore, we have seen the best method for increasing performance to be combining both supervision and direct stimulation of attention as each approach can support each other during the learning process.

The contributions of this chapter are:

- A formulation of how attention networks (transformer- and recurrent-based) can be stimulated to shift focus onto important features using prior knowledge via stimulation of the existing (and self-learning) attention mechanisms.
- A method to stimulate the attention of RNN cells (tested with LSTM and GRU cells) to important features of the input, and another method to stimulate the learnt self-attention mechanisms of transformer networks.

The key points for this chapter are:

- DNNs learn feature importance throughout the training process. However, one does not need to rely solely on this process if the important features are known before training. It is then possible to encode the importance of these known features with stimulated attention, thus providing an increased focus of the model and helping with the learning process.
- This stimulation of attention may also help to shift the DNN's focus off of potentially misleading or erroneous features, leading to a more trustworthy model overall.
- We created methods of applying stimulated attention to our OG detector to help focus the detector onto the grooming communication processes.
- In case study for OGD, these stimulation methods are applied to both LSTM/GRU recurrent cells and for attention mechanisms in transformers and unsupervised attention mechanisms.
- In experimenting with these different methods for stimulating attention in DNNs, some methods have been found to be more effective than others. For instance, LSTMs can benefit from both supervision and direct stimulation than either method applied singularly.

Chapter 8

Augmenting Training Data

To successfully train DNNs, typically, a large and diverse set of training data is needed. This data, in being diverse, exposes the model to wide variety of inputs from which it can learn and estimate class boundaries to inputs. In many scientific domains, however, the existence of readily-annotated datasets with which to train DNNs is not usually available.

In some cases, annotated datasets in these scientific domains exist, but may be too small to train large and complex DNNs without over-fitting to the small amount of data. To overcome these issues, researchers have relied on techniques that include augmentation of the dataset, where specific changes are applied to the existing dataset in order to enrich and expand the variety of data available. However, the synthetic samples as a result of the augmentation need to be realistic and simulate the process to which the data was originally recorded. Otherwise, the DNN may *shortcut* the learning process and rely on artefacts in the synthesised data to make predictions. Therefore, we advocate for the use of prior knowledge in making these augmentations realistic and meaningful to the domain of data in which they are being applied.

To avoid misleading features and become more robust to small changes in input, this chapter is dedicated to the topic of performing data augmentations in a domain-meaningful way as to enhance the training data. The first method, manual crafting of data-specific augmentations, focuses on the type of data being augmented and uses the prior knowledge of the data to construct domain-meaningful augmentations that provide the DNN with a larger variety of inputs.

Even if a DNN is trained with a large and diverse dataset, the DNN may still be susceptible to adversarial examples. This phenomenon is a concern for the use of DNNs in many safety-critical systems, as the existence of adversarial examples may indeed occur in the real world [9]. While the stimulated attention methods of the previous chapter can be used to provide focus for the DNN using prior knowledge and po-

tentially avoiding misleading features, DNNs, despite the shifted focus, may still be susceptible to these adversarial examples.

Our second method, adversarial training, provides automatic perturbations that are intended to fool the DNN to produce a misclassification, and through the process of including these perturbations with the correct classifications during training, potentially make the DNN more robust to small changes in input. In order to enhance and be more data-conscious with automated process of augmenting data through adversarial attacks, our method determines what constitutes as a *small* change to create an adversarial example, where *small* depends on the data, and each specific sample of this data. This prior knowledge then of what are small changes is used to inform the DNN during training to improve the robustness of the DNN with respect to small changes in input. Our method consists of generating neighbourhoods (Definition 2.0.3) surrounding each data point that defines the search region for adversarial examples and can be used to sample additional training data in the process of adversarial training. The size of each neighbourhood is adapted and unique for each data point by estimating local class boundaries relative to the centre of the neighbourhood, while also accounting for lack of information (due to small amount of sampling data) to estimate class boundaries.

While recent work by [41] illuminates the distinction between different forms of training DNNs to be more robust, and therefore data augmentation and adversarial training are both separate methods, we present them together in this chapter under the notion that both these methods are simply modifying the training data. The first method, with data-meaningful transformations, is the modification of data prior to training. The second method, via adversarial training, is the modification of data during the process of training. Though each method may *modify* the data, as [41] points out, these methods may indeed provide different definitions of robustness to the DNN, an advantage we can exploit by combining them in later chapters.

To evaluate both of these methods, the case study of detecting Type II solar bursts was used. This case study was chosen due to its issue concerning the small training set for DNNs. To enhance this dataset and make it viable for training DNNs, we implemented both methods and compared them with the original dataset.

8.1 Data-meaningful Transformations

Our first method of data augmentations considers the manual process of crafting augmentations that are domain-meaningful to the data they are being applied to. While a variety of common augmentations exist (e.g. image cropping, colour shifting, and affine transformations), more specific augmentations should be applied to cover the variety of possible signals and backgrounds that may occur in the real data. These augmentations should be informed by the prior knowledge of how the data was generated and then recorded. For example, in the classic MNIST dataset where handwritten digits from 0 to 9 were recorded to train an ML model to recognise these digits, one augmentation may be the reduction or increase of the stroke width of the digit to simulate different pen pressures when writing the digit. Another augmentation may be

a shearing of part of the digit to replicate an italicised style of writing that originates from the inclination of the hand with regards to the text line. Both of these example augmentations manipulate the properties of the digit in a way that simulate different conditions of the pen stroke in the process of handwriting.

These augmentations enrich the training data and provide the DNN with a larger variety of input, thereby potentially leading to improved generalisation performance. Moreover, specific augmentations could be designed to alleviate potentially misleading features that could occur in the data. By exposing these misleading features to the DNN during training, the DNN will be given an opportunity to learn to avoid these features when making its predictions.

To demonstrate this method, we have used the WAVES dataset (Section 4.3.1). These augmentations were designed from the use of prior knowledge of solar physics to be physically-meaningful to how Type II bursts can appear on spectrograms in the WAVES dataset.

8.1.1 Enhancing WAVES Spectrograms with Data Augmentations

The amount of annotated data available in the WAVES dataset for training and ML model is limited, especially so for DNNs. While we used pre-trained weights to mitigate this issue of limited data in the WAVES dataset, further measures can be taken to further alleviate this issue. Four augmentations were designed to improve this variation in the data. These augmentations were:

- **D1** – Removal of burst segments.
- **D2** – Dilation/erosion of bursts.
- **D3** – Horizontal/vertical lines of noise at varying widths.
- **D4** – Addition of Type III bursts.

The first augmentation, **D1**, randomly removes segments of the Type II burst. First, individual or separate parts of the burst were identified. Then, with these burst segments, zero or one of them was removed (while also being careful to remove the positive mask label) from the image. **D1** is shown in Figure 8.1. This type of augmentations aims to replicate the physical property of the full burst rarely being visible.

D2 either dilates or erodes the Type II burst. Erosion lessens the intensity of the burst, in this case the pixel values, while the dilation increases the burst intensity. The erosion and dilation simulates the physics-related property of the burst varying in intensity and width where, for example, the natural occurrence of erosion Type II bursts occurs at higher frequencies. Equation 8.1 describes the formula for both erosion and dilation of the Type II burst with respect to the frequency used in this data augmentation:

$$\overline{x} = x \pm \lambda x \frac{f}{f_{max}} \tag{8.1}$$

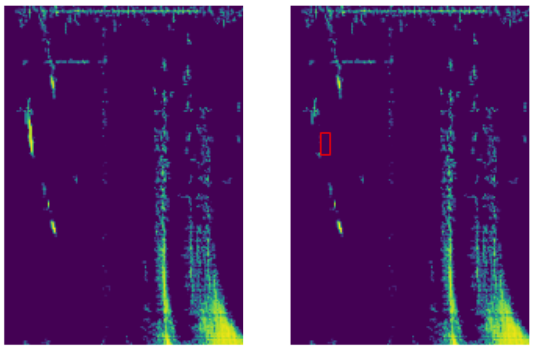


Figure 8.1: *Original image (left) and augmented version with the removal of a random segment (right). The red rectangle highlights the difference between the original and augmented image.*

where $\overline{x}$ is the augmented pixel value and λ is a user-defined modulation parameter that increases/decreases the amount of erosion/dilation. f and f_{max} is the frequency of the burst and maximum frequency in the image. An example of both erosion and dilation can be seen in Figure 8.2.

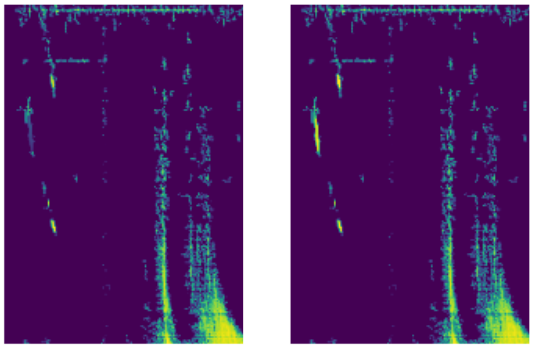


Figure 8.2: *Example of **D4** augmentation. Left figure shows the erosion of Type II burst, where the higher frequencies are more eroded than those at lower frequencies. Right figure dilates the same burst, increasing the dilation for lower frequencies.*

D3 adds horizontal or vertical lines to the image to represent noise from varying sources of interference (Figure 8.3). The horizontal noise was varied on the frequency range by increasing the height of the noise, while the vertical noise covers all frequencies for a varied amount of time by increasing the width. This type of augmentation aims to replicate the natural occurrence of interference.

From the preliminary experiments, it was shown that Faster R-CNN had often mistaken a Type III burst for a Type II. The **D4** augmentation aimed to combat this effect by increasing the number of Type III bursts the model is exposed to during training. From this, Faster R-CNN may better discern between these two different types of bursts more easily. To create this augmentation, ten different examples of Type III bursts in the training set were found and isolated, and then randomly inserted into other images.

It is relatively frequent for both Type II and Type III bursts to occur simultaneously or just before/after. Therefore, in order to insert additional Type III bursts into images, we used two different methods. The additional Type III bursts were either randomly

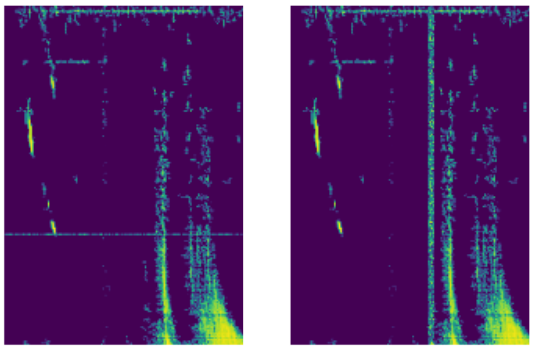


Figure 8.3: *Augmented image with horizontal noise (left) and vertical noise (right).*

placed along the time domain or correlated with the Type II burst to appear just before or just after (by 20 pixels) the occurrence of the Type II burst. An example of the additional Type III bursts are shown in Figure 8.4.

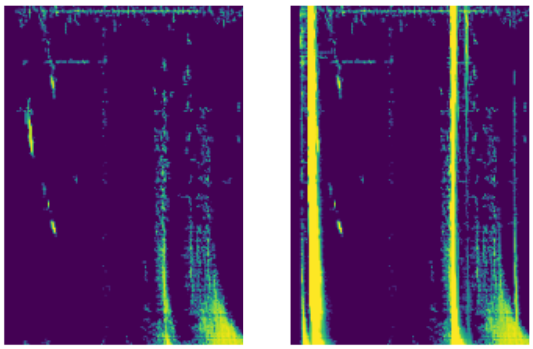


Figure 8.4: *Original (left) and **D4** augmented (right) event.*

From the combination of these four augmentations, the training subset of the WAVES dataset can be augmented to provide a richer variety of examples for training Faster R-CNN to detect Type II solar bursts. In Section 8.3.2, we experiment with each of these four augmentations to demonstrate how they affect the test performance in the object detection task.

8.2 Adversarial Examples

To make DNNs more robust against adversarial examples, it is common to use these adversarial attacks as a form of defence by introducing a measure of the DNN's susceptibility to the adversarial examples in the loss function. Through this process, the DNN is exposed to samples of the training set with small changes that could potentially lead to adversarial examples. This form of adversarial training can improve the DNNs robustness.

8.2.1 A Background on Adversarial Examples

Adversarial examples (Definition 2.0.2) occur when small changes or perturbations to the input result in a change of class output. These small changes can often be unnoticeable from the perspective of a human observer, as is the case in the work of Goodfellow et al. [3], or the perturbations result in a large amount of recognisable change but in an unimportant part of the input, as can be found in Papernot et al. [203].

It is well known that adversarial examples exist within small regions around the training data. Szegedy et al. [6] showed how the presence of adversarial examples contradicts a general belief that DNN's complexity makes them good at generalising to unseen examples. Later work by Goodfellow et al. [3] proposed the Fast Gradient Sign Method (FGSM) to generate adversarial examples from a closed $n-$ ball around the input. This method perturbed the input pixel values in the direction of the cost function's gradient. While [3] employed gradient information of the model, [204] used a black box assumption to find adversarial examples by choosing a random dimension (or pixel) on the input space [204] or on activation maps [118] for which an ε is added and subtracted from the original value. The model is repeatedly queried to determine if the perturbation would result in a misclassification. [205] applied a diffusion map algorithm to generate a reduced data space. The authors synthesised new points within sparse regions of the data space under the assumption that more adversarial examples will occur due to the lack of information in these areas. While we also used the assumption that adversarial examples may preferably appear in under-sampled areas where DNNs are under-trained, we also considered these areas very carefully, as the location of class boundaries are more uncertain there. Hence, we argue these areas should not be used blindly to search for adversarial examples, but the areas should be restricted based on uncertainty on class boundary location. We address this question of estimating uncertainty on class boundaries in Section 8.2.3.

Indeed, while in the context of these studies, the proposed algorithms are targeted at image-based classifiers in which small perturbations don't generally cause a change of class, and can be visually inspected for class type, the same algorithms may pose problems for other types of data. In datasets with jagged class boundaries, small changes may inadvertently push data across true class decision boundaries and thus incorrectly label the data. The focus of our study is therefore to provide a mechanism to quantify the amount of perturbation that can be safely applied (without change of class) to a dataset. This quantification may enable the use of existing adversarial generation algorithms, and allow their use for non-image types of data. Our proposed method estimates the density of samples within the data manifold to identify areas where true class boundaries may be uncertain. This allows defining regions where adversarial generation algorithms may be safely used in future work.

Many approaches for the automated construction of adversarial examples use a fixed-sized ε for the local neighbourhoods around training points [4, 3, 36]. To ensure these local neighbourhoods are within class boundaries, we propose the use of dataset complexity and of density analysis of the data manifold to provide an adaptive ε for each sample.

The adaptive definition of neighbourhoods relies on the properties of the data manifold, detailed in Section 8.2.2. In particular, they involve the notion of sampling density of the data manifold (Section 8.2.3) to assist in iteratively building neighbourhoods that may remain within class boundaries according to the available class information.

8.2.2 Manifold Properties

Many ML applications and learning techniques operate under the assumption of a manifold hypothesis [206, 207], where real/natural high-dimensional data lie on a low-dimensional manifold embedded within their high-dimensional space. The manifold hypothesis is often interesting for ML as it provides some explanation for the success of DNNs, as, in the process of accurately classifying data, DNNs may transform and twist the manifold on which the data lies into a form that can be separated by a hyperplane [208].

The consequence of this manifold assumption is that data has a local homeomorphism with a Euclidean space of lower dimensionality that is a local approximation to the manifold [207]. The data, when observed locally share the same properties as Euclidean space, but this is not the case when globally considering the entirety of the manifold. Therefore when analysing points relatively close together, i.e. the local neighbourhood of data points, the distance between points may be approximately measured with an Euclidean-based metric.

Definition 8.2.1. [38] Let $(\mathcal{X}, d)$ be a metric space with $\mathcal{X}$ the space of data points X and d the Euclidean distance. The ε-neighbourhood of a point $x_i \in X$ is defined as:

$$\mathcal{N}_\varepsilon(\mathbf{x}_i) := \{\mathbf{x} \,|\, d(\mathbf{x}, \mathbf{x}_i) < \varepsilon\} \tag{8.2}$$

Under the assumption of the manifold hypothesis, our method of creating adaptive neighbourhoods considered two properties of the dataset:

- **M1** The geometric complexity of the class boundaries.
- **M2** The sparsity/density of sampling from the data manifold that constitutes the training data.

M1 refers to situations where differently labelled data points lay close together in the topological space, and therefore any perturbation of the data points could result in passing the class boundaries, while wrongly labelling the perturbation the same as the original (Figure 8.5). This example may be mistakenly labelled as an adversarial example, despite the DNN having correctly learnt the true class boundaries. In this case, the DNN would produce a different output label for the perturbed input as the perturbation results in an input past the true class boundary.

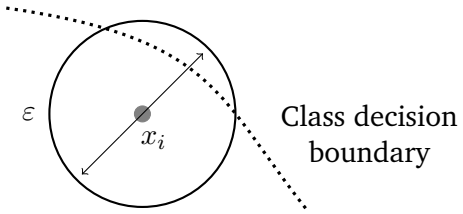


Figure 8.5: *Example where a data point x_i lies close to the class decision boundary. In these situations, too large ε values may push the synthetically generated point over true class boundaries.*

(a) *Sparse regions of the manifold may appear simple due to the lack of information.*

(b) *More data points enable more precise estimation of the class boundary.*

Figure 8.6: *Example scenario where true class boundaries are revealed when more data is collected.*

M2 concerns the number of samples from different regions of the data manifold. In sparse regions (i.e. a small numbers of samples from a region of the manifold), estimated class boundaries may seem deceivingly simple, e.g. linear with a wide margin [209] (Figure 8.6a). By increasing the number of examples (i.e. collecting more data), the true complexity of the classification task may become apparent (Figure 8.6b). Therefore, we should be cautious when estimating the class boundaries, and take into consideration how much information there is to gain from the surrounding data points to estimate such boundaries.

8.2.3 Estimating Sparsity/Density

The sparsity/density of the manifold provides some indication as to the amount of information to estimate class boundaries. If there are many points sampled within a small region of the manifold, then we can be more confident in estimating the class boundaries. Therefore, in regions of the manifold space where sampling density is low, i.e. sparse regions of the manifold, we will want to be cautious in expanding the adaptive neighbourhoods as there is less information to estimate these boundaries.

To measure the sparsity of the manifold, a radial basis function (RBF) φ is used. This is a function of the distance between some point $\overline{x}$ (the centre or origin) and another point x. This function results in a value in range of $[0, 1]$, where values approaching 1 indicates that $\overline{x}, x$ are sampled very close together within the manifold. Though many RBF functions may be applicable, we use the inverse-multiquadric function (Equation 8.3) as it has a non-shrinking value away from the origin. The RBF provides an appropriate measure of the sampling density/sparsity with respect to each data point in the data.

$$\varphi(\mathbf{x};\overline{\mathbf{x}}) = \frac{1}{\sqrt{1+(\gamma r)^2}},\ \text{where } r = \parallel \overline{\mathbf{x}} - \mathbf{x} \parallel \tag{8.3}$$

and γ controls the width of the RBF. Providing the RBF's width parameter is suitably chosen, we achieve a good measure of the density through the sum of the RBFs centred on all data points X^c of class c (Equation 8.4) producing a kernel density estimation (KDE) of the sampling density.

$$\rho_c(\mathbf{x}) = \sum_{\mathbf{x_j} \in X^c} \varphi(\mathbf{x};\mathbf{x_j}) \tag{8.4}$$

8.2.4 Constructing Neighbourhoods

Adaptive neighbourhoods are constructed for each point of our dataset, as a sphere of finely tuned radius. The neighbourhoods for all data points are created by jointly maximising the individual volumes of the spheres, under the constraint that sphere of different classes do not overlap. In addition, to account for lack of knowledge in under-sampled areas, the sphere's volume is limited to a linear function of the local sampling density $\rho_c(x)$ for the class. This is expressed by the following Lagrangian function, where the problem of finding the region size for each data point is expressed as an optimisation problem. This Lagrangian function (Equation 8.5) determines the maximum size of each adaptive neighbourhood with respect to our two constraints.

$$\begin{aligned} L(\varepsilon_1,\cdots,\varepsilon_n,\mathbf{x}_1,\cdots,\mathbf{x}_n,\lambda) &= v(\varepsilon_1,\cdots,\varepsilon_n) \\ &+ \lambda_g g(\varepsilon_1,\cdots,\varepsilon_n,\mathbf{x}_1,\cdots,\mathbf{x}_n) \\ &+ \lambda_h h(\varepsilon_1,\cdots,\varepsilon_n,\mathbf{x}_1,\cdots,\mathbf{x}_n) \end{aligned} \tag{8.5}$$

where v is the volume function, from R^n to R^n, to be maximised:

$$v(\varepsilon_1,\cdots,\varepsilon_n) = \begin{pmatrix} \varepsilon_1^D \\ \vdots \\ \varepsilon_n^D \end{pmatrix} \tag{8.6}$$

and

$$g(\varepsilon_1,\cdots,\varepsilon_n,\mathbf{x}_1,\cdots,\mathbf{x}_n) \;=\; \begin{pmatrix} \sum_{\substack{j\neq 1 \\ c(j)\neq c(1)}} \min(\parallel d\,(\mathbf{x}_1,\mathbf{x}_j) - (\varepsilon_1+\varepsilon_j)\ \parallel, 0) \\ \vdots \\ \sum_{\substack{j\neq n \\ c(j)\neq c(n)}} \min(\parallel d\,(\mathbf{x}_n,\mathbf{x}_j) - (\varepsilon_n+\varepsilon_j)\ \parallel, 0) \end{pmatrix} \tag{8.7}$$

Algorithm 1 Calculate ε_i for data point x_i

$\Delta\varepsilon^{min} \leftarrow 1e-20$ {Stop condition for $\Delta\varepsilon$ decay}
$n \leftarrow 1$ {Iteration number}
for all $\mathbf{x}_i \in X$ **do**
  $\varepsilon_i \leftarrow 0$
  $stop_i \leftarrow false$
end for
while $\exists i$ such that $stop_i = false$ **do**
  for all $\mathbf{x}_i \in X$ such that $stop_i = false$ **do**
    for all $\mathbf{x}_j \notin X^c(i)$ **do**
      if $d(\mathbf{x}_i, \mathbf{x}_j) \leq \varepsilon_i + \varepsilon_j$ **then**
        $stop_i \leftarrow true$
      end if
    end for
    if $stop_i = false$ **then**
      $\Delta\varepsilon_i \leftarrow e^{-\rho_{c(i)}(x_i)n}$
      if $\Delta\varepsilon_i \leq \Delta\varepsilon^{min}$ **then**
        $stop_i \leftarrow true$
      else
        $\varepsilon_i \leftarrow \varepsilon_i + \Delta\varepsilon_i$
      end if
    end if
  end for
  $n \leftarrow n + 1$
end while

$$h(\varepsilon_1, \cdots, \varepsilon_n, \mathbf{x}_1, \cdots, \mathbf{x}_n) = \begin{pmatrix} \varepsilon_1^D \\ \vdots \\ \varepsilon_n^D \end{pmatrix} - \alpha \begin{pmatrix} \rho_{c(1)}(\mathbf{x}_1) \\ \vdots \\ \rho_{c(n)}(\mathbf{x}_n) \end{pmatrix} + \beta \quad (8.8)$$

are the Lagrangian constraints for no intersection and volume depending linearly on density, respectively, which should be both equal to zero. $c(i)$ is the class label of point i. Note that when two spheres of different classes are too close to each other, they may not simultaneously respect both constraints of not intersecting while attaining their full size depends on local density for their respective classes. Therefore, the optimisation problem needs to be relaxed.

We developed an iterative algorithm which yields approximate results of the relaxed optimisation (Figure 8.7). This iterative version (Algorithm 1) is reminiscent of classification algorithms by [210] where an iterative algorithm continually expands neighbourhoods centred at each data point until it meets a neighbourhood of a different class. These neighbourhoods are used to make class predictions by determining under which neighbourhood does the unseen point adhere to. Other works [209, 211] perform an iterative method for complexity analysis algorithms. However, we further develop on these approaches to incorporate a decay function that limits the expansion

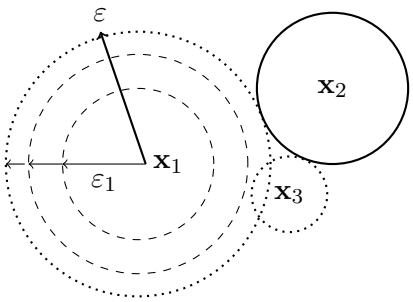


Figure 8.7: *Iterative ε-expansion process in a binary class scenario. The two classes are distinguished by the dotted and solid circles.*

of the neighbourhood based on the local density.

Sufficiently small initial neighbourhoods (e.g. neighbourhoods of size $1e-20$) are progressively expanded, with their radius at iteration n being $\varepsilon_i^n = \varepsilon_i^{n-1} + \Delta\varepsilon_i^n$, subject to avoiding overlap of neighbourhoods from different classes (Equation 8.7), and with an exponentially decreasing expansion that further depends on the local density of samples for the related class (Equation 8.8):

$$\Delta\varepsilon_i^n = e^{-\rho_{c(i)}(\mathbf{x}_i)\cdot n} \tag{8.9}$$

In areas of low density, so with an insufficient number of samples to safely determine the location of class boundaries, the expansion is slower and generates a conservative small final neighbourhood. The expansion stops when it reaches a low threshold $\Delta\varepsilon^{min}$ making it insignificant. This method may be further improved in future work by also accounting for the complexity of class boundaries in Equation 8.9, e.g. using complexity metrics of Ho and Basu [209].

8.3 Experiments

To evaluate the effectiveness that augmenting training data has on the generalisation performance and robustness of DNNs, we investigate each method in turn. We begin by first experimenting with the use of automated augmentation with adversarial attacks and adversarial training to improve the robustness of DNNs. Later experiments evaluate how effective the manual data-specific augmentations are for improving the generalisation performance of the Type II object detector.

8.3.1 Adversarial Training

Two experiments were performed to test the approach of adaptive neighbourhoods, where an ε value was created for each data point. The first was a simple experiment by using the Iris dataset. This experiment aimed to demonstrate how existing adversarial attacks can be generalised to non-image datasets where the potential adversarial

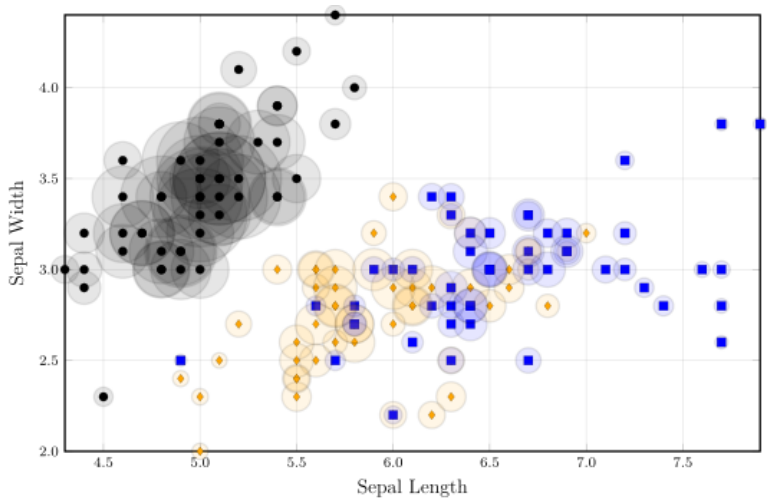


Figure 8.8: *Proposed adaptive neighbourhoods for the Iris dataset. The three classes of flower are represented by different shaped markers. The size of the neighbourhood for each sample is indicated with a circle centred on the data point. Intersections between neighbourhoods of different classes are not real but are visualisation artefacts coming from the 2D projection of 4 dimensions.*

examples cannot be visually inspected to determine if it is in-fact a true adversarial example. The second was a more complicated learning task using our solar burst detection case study. This case study tested the scalability of our approach of adaptive neighbourhoods to a more complicated dataset with a high dimensionality, and further aims to show the usefulness of the method for adversarial training. In both experiments, various forms of adversarial attacks were used to test the robustness of DNNs, in addition to using these same attacks as part of an adversarial training process. To evaluate the effectiveness of the adaptive neighbourhoods, we replaced the bound constraints of each of the adversarial attack algorithms with the learnt neighbourhoods for each data point. We show the results for adversarial attacks/defences with and without the adaptive neighbourhoods.

Adversarial Training with Iris Dataset

The experimentation begins with a non-image based dataset, the classic Iris dataset. This dataset contains 150 samples of three types of flowers (Virginica, Setosa, and Versicolour). Each sample of the dataset contains four features (Petal/Sepal's length-/width) with which a classification label may be induced. While this dataset is small (in terms of dimensionality and number of samples), it showcases the usage of adversarial training with adaptive neighbourhoods for non-image data. Later experiments focus on more complex datasets like using WAVES for solar burst detection.

Using this dataset, the iterative adaptive neighbourhood method (Algorithm 1) is applied to the Iris dataset to generate a unique ε value for every data point (Figure 8.8). These ε values enable the adaption of existing adversarial generation algorithms for non-image data. For this, FGSM [3], and un-targeted PGD [212] were selected. Though many other existing algorithms can be applicable, these algorithms were selected due to their coverage of approaches. For instance, FGSM performs a single step in the direction of the gradients, and un-targeted PGD is an iterative method in the

Table 8.1: *F_1 score of DNN for the Iris dataset using various adversarial defence methods. Scores are in the format: mean (standard deviation) over 10 k-folds. Bold font face indicates the best form of attack for each type of defence method.*

		Attack			
Defence	None	FGSM	PGD	FGSM+AN	PGD+AN
None	0.9745 (0.0413)	0.9278 (0.0618)	0.8572 (0.1036)	**0.7764 (0.0813)**	0.8461 (0.0968)
FGSM	0.9811 (0.0396)	0.9408 (0.0757)	0.8468 (0.1080)	**0.7873 (0.0785)**	0.8448 (0.0698)
PGD	0.9867 (0.0400)	0.9462 (0.0740)	0.8680 (0.0740)	**0.8508 (0.0746)**	0.8759 (0.0823)
Random+AN	0.9936 (0.0193)	0.9272 (0.0620)	0.8274 (0.0918)	**0.7935 (0.0822)**	0.8454 (0.0864)
FGSM+AN	0.9936 (0.0193)	0.9406 (0.0745)	0.8420 (0.0987)	**0.8140 (0.1085)**	0.8588 (0.1157)
PGD+AN	0.9936 (0.0193)	0.9472 (0.0642)	0.9472 (0.0642)	**0.8679 (0.0899)**	0.8753 (0.0864)

path of the steepest gradient. To combine these algorithms with the adaptive neighbourhoods, the ε bound constraint in the original algorithm was replaced with the learnt adaptive ε for each data point.

To learn a classifier from the Iris dataset, a simple MLP was trained with a single hidden layer of 10 neurons. This DNN was trained until the validation loss did not improve within 100 epochs, at which point, the model weights of the best network were reloaded to create the test predictions. The model's parameters were optimised using the Adam optimiser with a learning rate empirically selected at $5e-2$. We ran training procedure over 10 k-folds and report the mean and standard deviation of the F_1 scores averaged over the three classes. The results are shown for both adversarial generation algorithms with and without adaptive neighbourhoods (labelled AN for conciseness).

When using adversarial attack algorithms as a form of defence by using them for adversarial training, an additional loss term was added for the adversarial loss with a weighting of $\alpha = 0.5$ to equally prioritise the learning of perturbed and un-perturbed data:

$$\mathcal{L}_{total} = (1-\alpha)\mathcal{L}_{cls} + \alpha\mathcal{L}_{adv}$$

where $\mathcal{L}_{cls}$ and $\mathcal{L}_{adv}$ are the cross-entropy losses of the un-perturbed and perturbed data, respectively[1]. We used the following hyper-parameters[2] for FGSM: $\varepsilon = 0.1$, and for PGD: $\alpha = 2/255$, $\varepsilon = 0.1$, with 100 iterations. For the random perturbation with adaptive neighbourhoods (labelled *Random+AN*), the training data was perturbed by a sample of a normal distribution bounded by the adaptive neighbourhoods.

The cross-tabulation of results for the various forms of adversarial attacks and defences

[1]This method of adversarial training closely follows [3].

[2]These hyper-parameters are either set as the default for the algorithm, or, as in the case with the ε bounds, are suitably selected given the close proximity of some data points within the Iris dataset. For the adversarial attacks/defence without the adaptive neighbourhoods, the ε parameter is fixed for the entire dataset. Optimisation of these values may produce better results for with and without adaptive neighbourhoods.

are shown in Table 8.1. Firstly, we observed that PGD is a more effective adversarial attack than FGSM further decreasing the results from 0.9278 (no defence, FGSM attack) to 0.8572 (no defence, PGD attack). This trend continued for all forms of defences. When adversarial attacks were combined with the adaptive neighbourhoods (AN), the attack success was more substantial. For instance, we have seen with FGSM attack the F_1 score lowers from 0.9278 to 0.7764 (no defence, FGSM+AN attack). Though the advantage that adaptive neighbourhoods brings is stronger in FGSM, we observed that it helps both attack algorithms. As the neighbourhood sizes are larger than the default ε in some areas of the training data, the consequence is the adversarial generation algorithm can use a stronger attack (i.e. closer to the estimated decision boundary) in order to attempt to create an adversarial example. This illustrates the clear advantage that adaptive neighbourhoods has for using adversarial attacks in non-image data, as it can allow the attack strength to modulate according to the true classes of the surrounding data points and sparsity of the data.

When we used the adversarial generation as a form of defence (i.e. adversarial training), we have seen the F_1 score increase from 0.9475 (no defence, no attack) to 0.9867 (PGD defence, no attack), and to 0.9811 (FGSM defence, no attack), indicating the adversarial defence was not only helping to increase the robustness but also helped with over-fitting through the refinement of the classification boundary due to more data. This effect was made stronger with the inclusion of adaptive neighbourhoods, raising the F_1 score from 0.9867 (PGD defence, no attack) to 0.9936 (all neighbourhood variants defence).

Using PGD for defence also seemed to improve the robustness against both FGSM and PGD attacks, this was perhaps due to the fact that an un-targeted PGD attack operates as an iterative form of FGSM. This improved robustness also continued for defence against the AN variants of these attacks, where for example, the F_1 raised from 0.7764 (no defence, FGSM+AN attack) to 0.8505 (PGD defence, FGSM+AN attack).

However, when using the AN variants as a form of defence, the robustness against adversarial attacks is further improved. For example, the largest improvement was where the F_1 score after PGD attack raise from 0.8680 (PGD defence, PGD attack) to 0.9472 (PGD+AN defence, PGD attack).

The Random+AN defence had little impact on the defence of the network as it was not using any model information, other than providing some protection against FGSM+NBS, we did not see any significant change from performing no defence.

From these results, we may conclude that adaptive neighbourhoods are an effective adaptation for existing adversarial attacks, enabling an automatic modulation of attack intensity. There is also some benefit to using the adaptive neighbourhoods for a defence also, as we have observed an increased in robustness against adversarial attacks, perhaps due to the same modulation of attack intensity during training of the DNN.

Adversarial Training for Solar Burst Detection

To create adversarial examples and test the effectiveness of our adaptive neighbourhood algorithm for object detection, complementary adversarial example generation

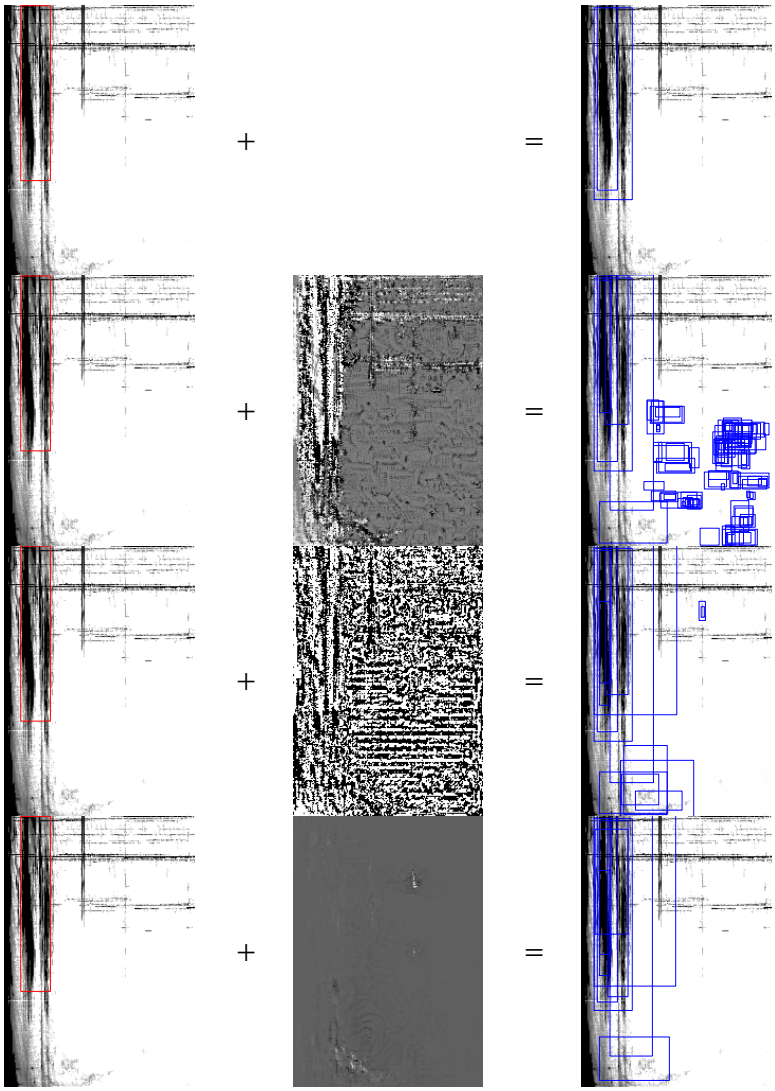


Figure 8.9: *Adversarial example generation methods applied to an solar event containing both Type II and Type III bursts. Top row shows the original predictions made without applying any adversarial attacks. The second row shows PGD, third FGSM, and bottom row DAG method. Left column images are the original event with a red rectangle to highlight the Type II burst, with the centre column being the amount of pixel-wise perturbation. The result of the perturbation is shown in the right column with predicted detections made by Faster R-CNN in blue rectangles.*

algorithms were selected. These algorithms were PGD [130], DAG [131], and a modified method of [130] to use the more simple FGSM instead of PGD. We used the following hyper-parameters for FGSM: $\varepsilon = 0.1$; for PGD: $\varepsilon = 0.1$ $\alpha = 2/255$, for 40 iterations; and for DAG: $\gamma = 0.07$ for 200 iterations. Similar to our previous experiment, we replaced the bound constraints of these algorithms with the calculated adaptive neighbourhoods to form the *+AN* variant of each algorithm. Due the computational time required to compute adversarial examples with DAG, this algorithm is used for attack only, and not adversarial training.

The effectiveness and amount of perturbation can be seen in Figure 8.9. In this example, PGD had mostly perturbed the edges of the Type II and Type III bursts, while

Table 8.2: *F_1 score performance on the WAVES dataset using Faster R-CNN. Numbers highlighted in a bold font face indicate the best achieving adversarial attack for each form of defence.*

		Attack					
Defence	None	FGSM	FGSM+AN	PGD	PGD+AN	DAG	DAG+AN
None	0.568	0.539	0.486	0.198	**0.105**	0.399	0.251
FGSM	0.463	0.458	0.178	0.013	**0.012**	0.055	0.028
FGSM+AN	0.480	0.465	0.462	**0.007**	**0.007**	0.043	0.023
PGD	0.421	0.425	0.379	0.391	0.359	0.378	**0.259**
PGD+AN	0.364	0.359	0.330	0.339	0.324	0.330	**0.212**

Table 8.3: *IoU performance on the WAVES dataset using Faster R-CNN. Numbers highlighted in bold indicate the best achieving type of adversarial attack for each form of defence.*

		Attack					
Defence	None	FGSM	FGSM+AN	PGD	PGD+AN	DAG	DAG+AN
None	0.611	0.589	0.527	0.438	**0.318**	0.556	0.485
FGSM	0.679	0.676	0.520	0.456	0.412	0.390	**0.343**
FGSM+AN	0.684	0.676	0.667	0.315	**0.302**	0.350	0.316
PGD	0.681	0.676	0.658	0.665	0.645	0.636	**0.565**
PGD+AN	0.678	0.673	0.655	0.663	0.648	0.639	**0.565**

adding perturbations to the background. This appears to have caused the object detector to create many mislocalisations of Type II bursts in the background of the event. FGSM, despite also modifying the background, caused fewer mislocalisations compared with PGD, but was still successful in creating adversarial examples. These examples were very similar to those created by DAG, which mostly perturbed particular objects in the event, i.e. the Type III burst in the bottom of the image and some of the horizontal/vertical noise. These perturbations were again enough to fool Faster R-CNN.

Similar to [130], adversarial examples were created via all loss terms of the object detector, thereby perturbing both objectness scores and bounding box detection. Therefore, to compare the methods, we show the resulting IoU score and F_1 score of Faster R-CNN with these perturbations that demonstrates the methods ability to create adversarial examples for both objectness and bounding box detection.

The results of applying adversarial attacks and defences to Faster R-CNN is shown in Table 8.3. Similar to the results of Table 8.1, we have seen PGD is a more successful attack than FGSM, lowering the IoU score from 0.611 (No defence, no attack) to 0.438 (No defence, PGD attack). With the inclusion of adaptive neighbourhoods, both FGSM and PGD were more effective. For instance, we have seen FGSM reduce the IoU score from 0.589 (No defence, FGSM attack) to 0.527 (No defence, FGSM+AN attack), and from 0.438 with (No defence, PGD attack) to 0.318 (No defence, PGD+AN attack).

By using FGSM for adversarial training, we observed the same increase in generalisation performance as we did with the Iris set. Here we have seen the IoU score raise

from 0.611 (No defence, no attack) to 0.679 (FGSM defence, no attack). Using FGSM as a form of defence, the original adversarial attack algorithms were less successful, while their adaptive neighbourhood variants were more successful. For example, the IoU score after a PGD+AN attack with no defence was 0.486, while when using FGSM for adversarial training, the IoU score was 0.390.

However, when evaluating the F_1 score, we have observed the score decrease upon using adversarial training. For example, without adversarial training, the F_1 score was 0.568, while using FGSM for adversarial training reduced the resulting score to 0.468. The reduced F_1 scores are further enhanced by the stronger forms of attack, such as PGD and PGD+AN. The contrast between the IoU and F_1 scores suggest adversarial training was making the object detector fail to detect Type II bursts, but when they were detected, they are better localised, thus improving the IoU score. Furthermore, using FGSM adversarial training with this object-detector had also made the model more susceptible to different types of attacks. For instance when using FGSM+AN as a defence, we observed the F_1 drop to 0.007 with the PGD attack, where it was previously higher. This further highlights adversarial training for this dataset and this model type may not produce a more robust model to different forms of adversarial attacks.

In the majority of cases, by using the adaptive neighbourhoods with the adversarial attacks, the attack success was increased for both F_1 and IoU scores. This was for the case for all forms of adversarial attacks: FGSM, PGD, and DAG. However, contrary to the results in Table 8.3, we did not observe the same generalisation benefit from the adaptive neighbourhoods. Only in one case did the adaptive neighbourhoods increase the generalisation IoU performance from 0.679 (FGSM defence, No attack) to 0.684 (FGSM+AN defence, No attack). This further highlights that potentially stronger adversarial training does not help the model to generalise better.

These results have demonstrated that adaptive neighbourhoods are an effective method for improving the attack success of existing adversarial attacks. However, in the case study of solar burst detection, attack success may not be entirely helpful for improving generalisation with adversarial training, as it serves to only decrease the F_1 score, even if the IoU is improved.

8.3.2 Contribution of Data-specific Augmentations

Our next set of experiments evaluate the effectiveness of the data-specific augmentations for the generalisation performance of Faster R-CNN to detect Type II bursts. We first investigate the individual performance of each of these augmentations, before combining the best performing augmentations and comparing the performance with the original WAVES dataset.

Faster R-CNN was trained on the WAVES dataset without applying any augmentations to provide a baseline measure to compare the augmentations. This baseline measure can be found in Table 8.4. We report the precision, recall, and F_1 score for predictions made on the test set. Before creating these predictions, we filter out any bounding boxes with an objectness score less than 0.3. This score was created by finding an

Table 8.4: *Performance results for Faster R-CNN trained with different augmentations applied to the training data. Results are in the format: Mean (Standard Deviation) of 10 trails. Best performance over 10 trails is highlighted using a bold font face.*

Experiment	Precision	Recall	F_1 Score
Baseline	0.335 (0.090)	**0.854 (0.104)**	0.468 (0.071)
Remove Segment (D1)	**0.480 (0.147)**	0.694 (0.164)	0.533 (0.059)
Dilation (D2)	0.351 (0.071)	0.850 (0.058)	0.492 (0.068)
Erosion (D2)	0.397 (0.143)	0.839 (0.107)	0.515 (0.089)
Horizontal Line (D3)	0.420 (0.131)	0.835 (0.122)	0.536 (0.073)
Vertical Line (D3)	0.429 (0.103)	0.801 (0.103)	0.544 (0.063)
Type III Bursts (D4)	0.468 (0.103)	0.791 (0.128)	**0.571 (0.072)**

optimal threshold hold for filtering based on the objectness score. These classification scores are made with respect to each of the different segments of the bursts, i.e. a burst will have multiple segments (Section 4.3.4).

Individual Augmentations

In this experiment, augmentations were individually applied. Experimenting in this way indicates the contribution of each type of augmentation when they are all applied to the image in later experiments.

From these results, we observed all augmentations increased the mean performance of the F_1 score slightly while often reducing the standard deviations. This effect was made more clear with aid of a box plot (Figure 8.10). When individually applied, the augmentations did not drastically improve the mean performance of the model but did help in increase the lower quantile of results. In later experiments augmentations are combined to create a greater statistical change.

The **D4** augmented (additional Type III bursts) appeared to be the best performing augmentation, with each metric gaining the largest increase over the baseline. This helped the model differentiate between the Type III and Type II bursts (a major cause for failure in the baseline model). The second-best performing augmentation was the Erosion of the Type II bursts, as the real Type II bursts are often not fully visible and some segments of the burst tend to be very faint, the DNN is getting more experience of this as a result of this augmentation. Furthermore, in these results, the removal of segment augmentation lead to generally worse performing results. Interestingly though, if the erosion effect is made strong enough, it would converge to the same result as entirely removing the segment. Therefore, by modulating the strength of the erosion and increasing it further, slightly worse results from erosion should be shown.

We tested the effect that increasing the amount of erosion has on the test results by increasing $\lambda = 0.5$ to $\lambda = 2.0$ when augmenting the training data. The difference between these two parameters are demonstrated in Figure 8.11.

In Table 8.5 and Figure 8.12, there are slightly different results between the two set λ parameters. With increased erosion, the resulting model was much more precise in its predictions and but also failed to detect more Type II bursts. This effect follows that

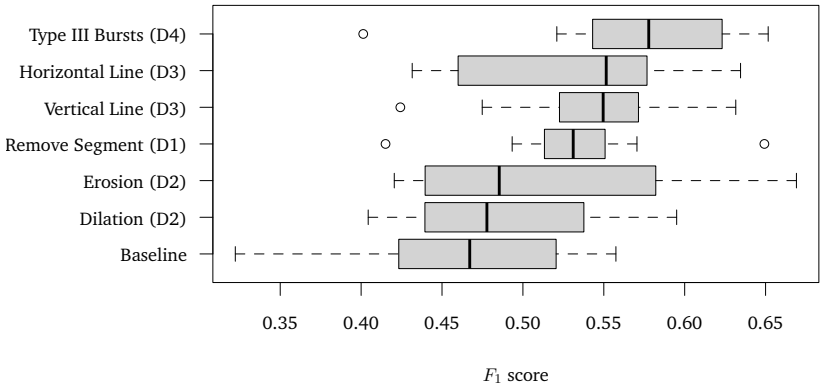


Figure 8.10: *Boxplot of F_1 score for Faster R-CNN trained with varying data augmentations.*

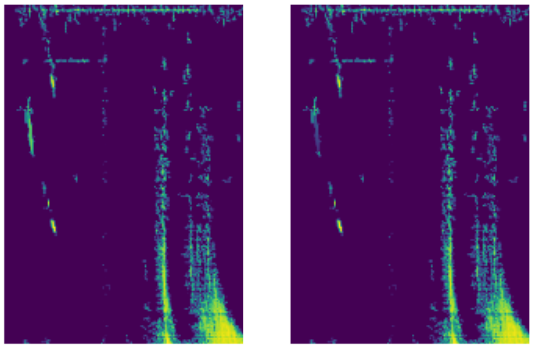


Figure 8.11: *Effect of λ parameter on the amount erosion. In the left figure $\lambda = 0.5$ and right $\lambda = 2.0$. Figure (right) demonstrates an increased erosion of burst segments in higher frequencies than those of left figure.*

of the removal segment augmentation where the precision increased while the recall also decreases. In both of these augmentations the DNN is perhaps requiring stronger signals to confirm the presence of Type II bursts given the small visible segments, thereby leading to increased precision of the results.

Augmentation vs Non-Augmentation

In this experiment, multiple augmentations were applied simultaneously for comparison against the baseline model in Table 8.4. From the analysis in Section 8.3.2, the following augmentations were observed to be the most helpful in improving the detection performance of Faster R-CNN:

- **D2** – Erosion of Type II bursts.
- **D3** – Horizontal and Vertical lines of noise.
- **D4** – Additional Type III bursts.

Therefore only these three types of augmentations were used when training Faster R-CNN as one experiment while also applying them all regardless of performance in

Table 8.5: *Comparitive performance for the effect of λ in Equation 8.1.*

λ	Precision	Recall	F_1 Score
0.5	0.397 (0.143)	0.839 (0.107)	0.515 (0.089)
2.0	0.677 (0.140)	0.518 (0.127)	0.563 (0.083)

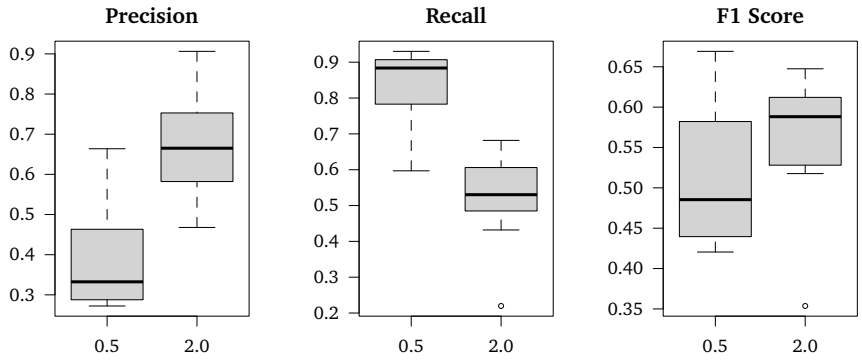


Figure 8.12: *Boxplot of Precision, Recall, and F_1 score between two different forms of erosion.*

previous experiments as another experiment. The results of this comparison are presented in Table 8.6. The best-performing augmentations (labelled "Best Performing Augm."), when applied to the data, improved the performance of Faster R-CNN from 0.468 ("Baseline") to 0.562 F_1 score. This score, while improving on the baseline, is not as performant as the additional Type III bursts individually applied. This may be due to too many augmentations being applied in parallel, as when all augmentations were applied (labelled "All Augm."), a slight mean increase (from 0.468 to 0.525) was shown. We may experience larger improvements over the baseline by only applying one augmentation per image.

From these scores, applying the augmentations may result in a positive outcome. However, when analysed in a different manner, such as with a Receiver Operating Characteristic (ROC) curve, the advantage is not so clear. Figure 8.13 shows the PR and ROC curves for the best performing model out of the ten trails. Here we see a different interpretation of the performance of the models. While in Table 8.6 we see augmentations slightly increasing the mean values for the F_1 score, the ROC and PR curves show the model trained on the non-augmented data to be generally better performing. Figures 8.13 (a) and (c) show Faster R-CNN trained on augmented data to be more '/conservative/' with its predictions, i.e. requiring more evidence of the existence of Type II burst before predicting the positive class.

In Figure 8.14 we show the PR Gain curves for all ten trails of training the base model (a Faster R-CNN trained using the original WAVES dataset), and ten Faster R-CNN models trained using the augmentations. From this visualisation, a model trained using augmented data generally predicts Type II bursts with lower scores, indicating its decreased in confidence in the decision. Generally, there was tighter grouping of PR Gain curves when using the augmented data, indicating the augmentations are helping the DNN to predict correctly from samples that would otherwise reduce the PR Gain scores. However, the area under these curves usually were lower than the baseline

Table 8.6: *Comparative results between training Faster R-CNN with and without Augmentations.*

Experiment	Precision	Recall	F_1 Score
Baseline	0.335 (0.090)	0.854 (0.104)	0.468 (0.071)
Best Performing Augm.	0.514 (0.124)	0.715 (0.200)	0.562 (0.097)
All Augm.	0.500 (0.109)	0.628 (0.196)	0.525 (0.074)

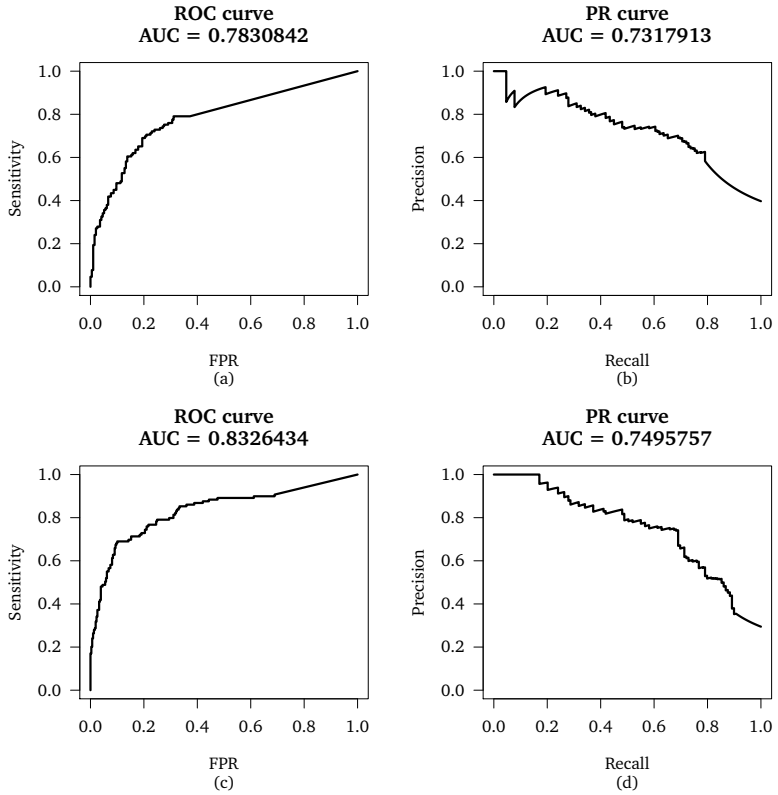


Figure 8.13: *PR and ROC curves for Faster R-CNN trained on augmented and non-augmented data. Figures (a) and (b) are for the augmented data, while (c) and (d) are for non-augmented data. For each plot, the area under curve (AUC) score is shown where higher numbers are better.*

results.

We visually interrogate the predictions made by each model. In Figure 8.15a we show the non-augmented input. Figure 8.15b shows the predictions (in blue) and ground truth (red) bounding boxes surrounding the Type II bursts. In this case, the burst itself has been detected. While the Faster R-CNN trained on augmented data misses the burst (Figure 8.15c). This is due to the model generally requiring more evidence to predict Type II.

In other examples, we found that the additional Type III bursts help Faster R-CNN differentiate between Type II and Type III bursts. Figure 8.15h shows predictions intersecting with the much more salient Type III burst, while in Figure 8.15i the predictions were isolated to the Type II burst. In Figures 8.15e and 8.15i, in both cases, the Type II burst has been detected, yet there were many more bounding boxes created with the

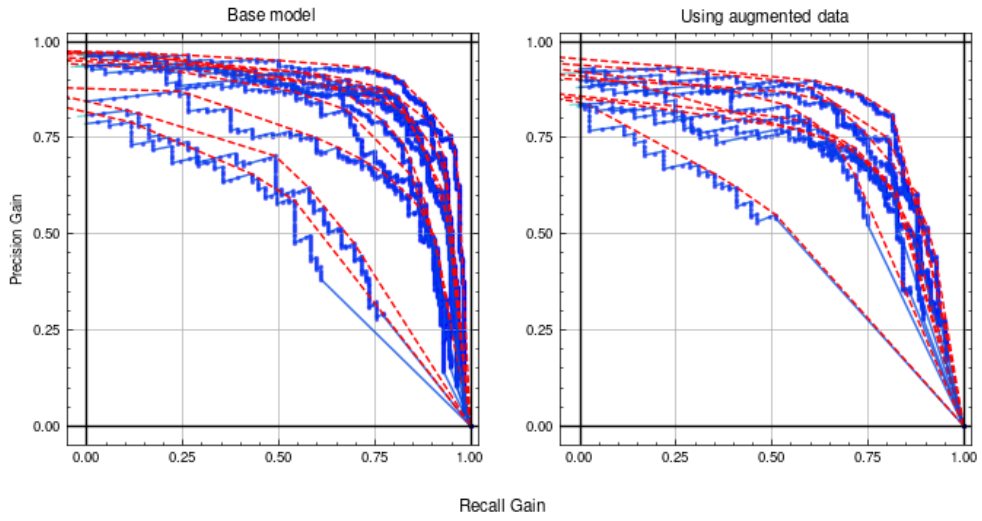


Figure 8.14: *PR Gain curves for Faster R-CNN trained with non-augmented data ('Base model', left) and trained with augmented data ('Using augmented data', right).*

Table 8.7: *IoU scores for best performing Faster R-CNN trained with (non)-augmented WAVES data.*

Dataset	IoU Score
Non-augmented	0.6743
Augmented	0.7119
Non-augmented (Type II only)	0.2478
Augmented (Type II only)	0.2786

non-augmented data. Faster R-CNN trained with augmented data was undoubtedly more precise in its prediction for this example, yet both models failed to detect the small segment to the Type II burst, perhaps due to the more salient Type III burst.

From these visualisations, we might expect the model trained from augmented data to be more precise. To further investigate the differences between these models, we evaluate the bounding box detection using the IoU metric. Taking once more the most accurate Type II detectors from our baseline and best augmentations, mean IoU scores are calculated. These results are presented in Table 8.7. From this metric, there was an improvement over the non-augmented data from 0.6743 to 0.7119. We also calculated the IoU scores only for the images that contain Type II bursts (the positive case). Once more, we see an improvement in IoU performance when using the augmented data. These scores suggest that while augmenting the dataset did not drastically improve the ability of Faster R-CNN to detect Type II bursts, its positive predictions, when made, were closer to the ground truth in shape and size. In addition, Faster R-CNN trained with augmentation data was also observed to be more robust to salient features such as Type III bursts and noise that might otherwise be incorrectly labelled as Type II bursts.

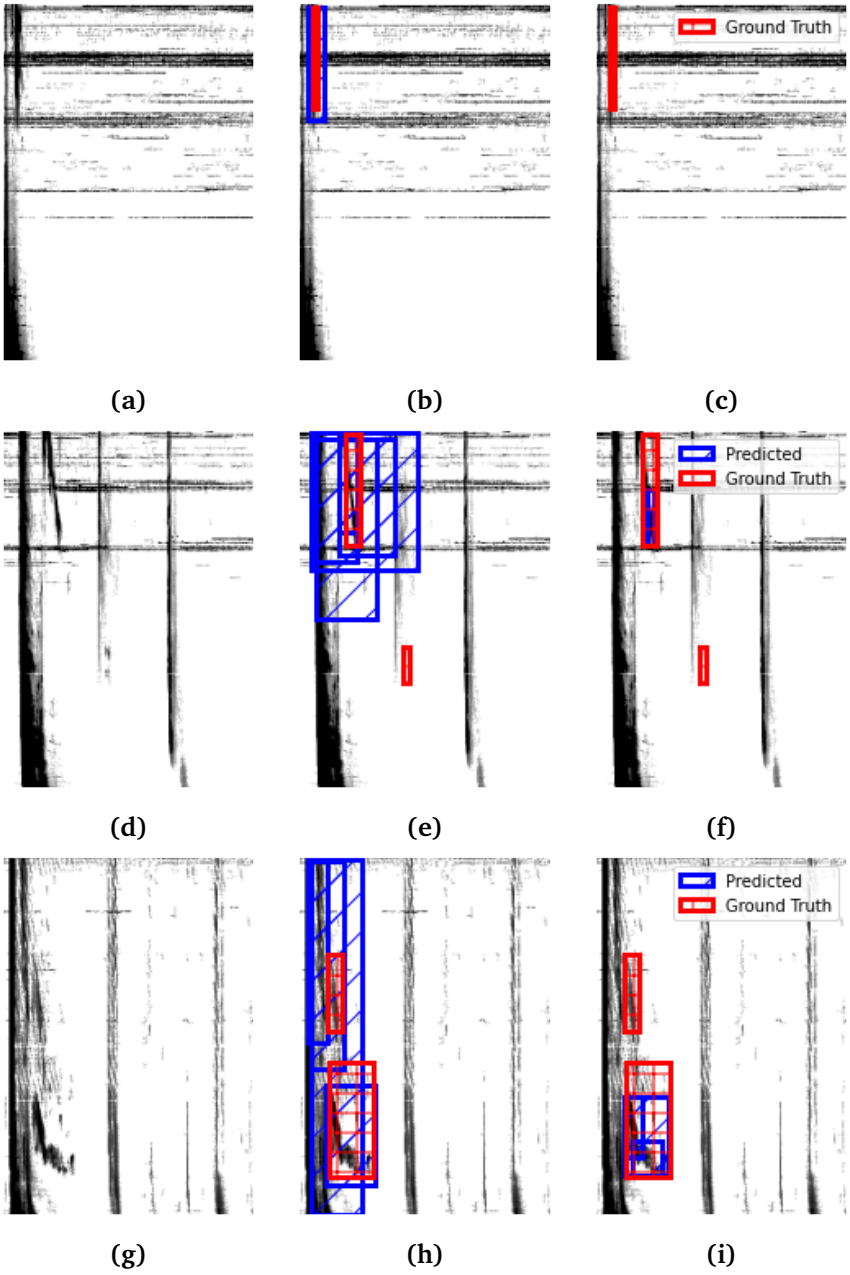


Figure 8.15: *Three example inputs to the Faster R-CNN. Images in the first column are the input to the trained models. Middle column images are the predictions by Faster R-CNN trained on non-augmented data. Right column are the predictions made by Faster R-CNN trained with augmented data.*

8.4 Chapter Summary

Through the construction of meaningful augmentations, DNNs can improve in performance as well as robustness to avoid potentially misleading or noisy features. In this chapter, we have demonstrated an automatic and manual method of creating augmentations. This first manual method consists of crafting domain-meaningful data

augmentations using the prior knowledge of how the data was created, measured, and the environment in which it occurs.

In creating data-specific augmentations, we used the WAVES dataset from the solar burst detection case study. In this study, a series of four augmentations were created using the knowledge of the physical properties of how the bursts are measured (i.e. the types of interference patterns that can occur), how the bursts can appear in the spectrograms (i.e. the erosion or dilation at different frequency ranges), and how DNNs can be mislead into detecting a Type III burst as a Type II (i.e. additional Type III bursts being correlated with the appearance of Type II bursts). These data-specific augmentations were used to enhance and effectively improve the detection performance of Faster R-CNN in localising Type II bursts.

To improve the automated process of adversarial training, our second method constructs adaptive neighbourhoods to each sample of the data. These adaptive neighbourhoods allow the adversarial generation method to be more data-specific and allows the adversarial attack to modulate their strength to each individual data point. By doing so, the attacks can more effectively generate adversarial examples in regions of the manifold space where the sampling density is higher, thereby increasing the information with which to estimate class boundaries. The adversarial attack, by increasing the intensity of perturbation, can explore a larger neighbourhood around the data point to find an adversarial example that fools the classifier or detector. Ultimately, this enhanced modulation can increase the robustness of DNNs while also not inadvertently confusing the detector with false adversarial examples that are past true class boundaries.

To evaluate the adaptive neighbourhoods, two experiments were performed. The first used a simple non-image dataset, the Iris data. At this point, we observed using adaptive neighbourhoods may help the test performance of the classifier, while also boosting the robustness against forms of adversarial attacks. Our second experiment, using the solar burst detection case study, has further demonstrated the usefulness of adaptive neighbourhoods for increasing the intensity of perturbation w.r.t. the data point being perturbed. In this study, we have observed an increase in attack success for these forms of adversarial attacks.

The contributions of this chapter are:

- A formulation of how augmentation of data can incorporate prior knowledge. Furthermore, adversarial training can aid in this process by using a fixed-sized region to limit the amount of perturbation that can be made as a result of a adversarial algorithm.
- A implementation of adaptive neighbourhoods to calculate the maximum size of perturbation for each data point. This method was tested using the detection of Type II bursts case study.

The key points for this chapter are:

- Augmenting the training data is an effective method of improving the perfor-

mance and robustness of DNNs. By the introduction of prior expert knowledge, these augmentations can be made domain-meaningful w.r.t. the data being used to train DNNs.

- Adversarial training, while already an automated process of augmentation, can be made more data conscious through the addition of adaptive neighbourhoods.
- Adaptive neighbourhoods is the process of expanding the neighbourhood of adherent points around each data point. This process takes into consideration both the estimated class boundaries and the sampling density with which to make such an estimation.
- The neighbourhoods can be used to adapt existing adversarial generation algorithms to enhance the success of these algorithms.
- The use of adaptive neighbourhoods may improve both the generalisation performance of the classifier or detector by providing more input with more variety thanks to the increased range of perturbation that can be applied to each data point in certain parts of the manifold.
- When combined with adversarial generation algorithms, the attack success is increased. This helps both the attacker, but also the defence, by using the same algorithm during the training process.
- Data-specific augmentations can be created by considering the properties of how data was created. From this knowledge, more meaningful data augmentations can be created to improve the task performance of DNNs.

Chapter 9

Combining Prior Integration Strategies

In previous chapters, methods to integrate domain knowledge into DNNs were characterised and tested using three different case studies: (1) detection of OG in chat logs; (2) estimation of energies in molecular and crystalline systems; and (3) detection of Type II solar bursts. For these case studies, each prior integration strategy was considered in its isolation, and no strategy was combined with another to improve it's effectiveness. In this chapter, however, the interaction of different strategies are evaluated to investigate their combined ability to improve performance on the objective of the case study.

Furthermore, in this chapter, we shall create the final augmented classifiers and detectors using the most effective prior integration strategies. Taking each of the base models from the different case studies, the best performing integration strategies are added to the DNNs and compared to the original non-augmented base models. For a thorough comparison with existing research, these final augmented DNNs are evaluated against the current state-of-the-art models, and other non-DL approaches.

9.1 Online Grooming Detection

A summary of the results presented in previous chapters is shown in Table 9.1. The most effective methods for integrating domain knowledge of the discourse of groomers were:

1. Indirect feature specialisation from estimating the OG processes using a shared representation. In applying this method, the seven different OG processes were estimated using the hidden states of the LSTM that was also used for classification.

2. Direct feature specialisation using the Elastic Pulling technique, where the distance between OG word variants were reduced within the WSR. In this case study, both Manifold Learning and Elastic Pulling could be effective for improving the performance of the OG detector. However, as Elastic Pulling was the best performer in the majority of experiments, this technique was chosen for the final OG classifier.

3. Combined supervision and direct stimulation (B) of both self-attention, unsupervised and recurrent cells. In both strategies the attention mechanisms were stimulated during training upon the occurrence of OG processes.

Therefore, to create the final classifier, these best performing integration strategies were combined to create the final augmented OGR-R and OGD-T DNNs[1]. The OGD-R and OGD-T base models were retrained with the prior integration strategies to form their augmented counterpart, Augmented OGD-R and Augmented OGD-T. The performance results of the base, augmented, non-DL baselines, and state-of-the-art models are presented in Table 9.2. Although the base model OGD-R performed slightly worse than [175], its augmented version outperformed it by large difference. XLNet of OGD-T was the best performing non-CL-augmented model. Through the combination of CL knowledge on word variants and OG processes, the results were improved and produced the new state-of-the-art.

In order to verify that the improved results come from a better understanding of language provided by CL knowledge, rather than merely from additional regularisation, we also compared against L_1 and L_2 regularised version of both base models. Although regularisation did improve the results, the performance gains from integrating CL knowledge were superior for both models.

Using the area under the PR-Gain curves (AUPRG) [213] metric for both base and augmented models (Table 9.3), we observed that due to the large number of true negatives predicted by the classifiers, the performance of each model was relatively close. Despite the closeness of the metrics, we still observed that the DNNs slightly improve when using the prior knowledge integrations.

In Table 9.4, the individual augmentations were applied to the simple LSTM model to determine the effect each augmentation had on the final result. At each successive stage of augmentation, the performance of the classifier had improved, with the largest increase being the first augmentation of supervision and direct stimulation of the LSTM cells. The improvements have highlighted that these methods are complementary, even those that use similar priors, such as the case with stimulated LSTM and stimulated attention. We observed that at each additional stage of augmentation, the model improved in performance.

Swapping of discriminative word variants – When the two components of all dis-

[1]OGD-R base model consists of a standard embedding layer followed by two layers of LSTM cells. All hidden states from the final layer of the LSTM's are used in an attention mechanism to optimise the encoding of the conversation for the final classification using a linear projection into the classification space. OGD-T follows the same logical process as OGD-T, but we use a transformer to build potentially more meaningful word-representations. For full details on the architectural design of these two base models, we refer the reader to Section 4.2.2.

Table 9.1: *Summary of knowledge integration strategies applied singularly to the base models. Bold are improved results with respect to no augmentation, i.e. the base model.*

Model	Strategy		Precision	Recall	AUPR	F_1	$F_{0.5}$	$\Delta\mathcal{D}/\overline{\mathcal{D}}$
OGD-R	No augmentation		0.867	0.794	0.867	0.829	0.851	– / 3.72
	Supervised word embed. modif.		0.834	0.765	0.824	0.798	0.819	0.75/0.91
	Manifold learning		0.916	0.753	**0.881**	0.827	**0.878**	0.65/1.29
	Elastic pulling		0.878	0.808	**0.877**	**0.841**	**0.863**	0.83/0.61
	Aux. OG process detection		0.890	0.768	**0.873**	0.825	**0.863**	–
	Specialised proc. channels (LSTM)		0.912	0.789	**0.899**	**0.846**	**0.884**	–
	Stim. attention	supervised	0.839	0.804	**0.879**	0.821	0.832	–
		direct stim. (A)	0.822	0.817	**0.877**	0.820	0.821	–
		direct stim. (B)	0.870	0.808	**0.906**	**0.838**	**0.857**	–
		superv.+direct stim. (A)	0.859	0.819	**0.897**	**0.838**	0.851	–
		superv.+direct stim. (B)	0.929	0.741	**0.908**	0.824	**0.884**	–
	Stim. LSTM	supervised	0.924	0.752	**0.891**	0.829	**0.883**	–
		direct stim.	0.856	0.797	0.863	0.825	0.843	–
		superv.+direct stim.	0.906	0.781	**0.901**	**0.839**	**0.878**	–
OGD-R w. GloVe	No augmentation		0.834	0.775	0.849	0.803	0.822	– / 5.87
	Supervised word embed. modif.		0.836	0.781	0.828	**0.808**	**0.825**	0.90/0.93
	Manifold learning		0.824	0.796	**0.854**	**0.810**	0.818	0.89/0.53
	Elastic pulling		0.905	0.761	**0.880**	**0.827**	**0.872**	0.92/0.73
OGD-T	No augmentation		0.900	0.871	0.940	0.886	0.894	–
	Aux. OG process detection		0.918	0.861	**0.943**	**0.889**	**0.906**	–
	Specialised proc. channels (LSTM)		0.954	0.835	0.936	**0.891**	**0.928**	–
	Stim. attention	supervised	0.919	0.862	**0.943**	**0.890**	**0.907**	–
		direct stim. (A)	0.894	0.885	**0.945**	**0.889**	0.892	–
		direct stim. (B)	0.916	0.866	0.940	**0.891**	**0.906**	–
		superv.+direct stim. (A)	0.891	0.881	**0.941**	0.886	0.889	–
		superv.+direct stim. (B)	0.918	0.862	**0.941**	**0.889**	**0.906**	–
	Stim. LSTM	supervised	0.938	0.857	**0.944**	**0.896**	**0.921**	–
		direct stim.	0.896	0.896	**0.944**	**0.887**	0.892	–
		superv.+direct stim.	0.960	0.846	**0.945**	**0.899**	**0.935**	–

criminative variants pairs were swapped (i.e. those that were not moved closer in the word embedding) within the testing corpus, there was a decrease in performance of the base models (Table 9.5). This drop in performance indicates that the language bias captured by these variants is indeed indicative of OG. However, it is important to note that any such difference in language could represent the socio-economic background rather than specific to grooming online. Therefore, it would be dangerous to classify a conversation based solely on this language bias. Both the LSTM and XLNet models may exploit this language bias, but they also account for the context of words, which may help alleviate these risks, especially for XLNet, which was less affected by the variants' swap. In the OG corpus, positive and negative conversations come from similar or identical online platforms and should therefore contain an equal representation for each socio-economic group present. Furthermore, upon examination, it was estimated that the variants used preferably by groomers may be primarily related to the goal of online grooming, with attempts to look more friendly and laid back to a young audi-

Table 9.2: *Comparative evaluation of OG classification methods. Best performing metrics are presented in a bold font face.*

Method	Precision	Recall	F_1	$F_{0.5}$	AUPR
Naive Bayes	0.240	**0.974**	0.385	0.283	0.727
SVM	**0.997**	0.337	0.504	0.716	0.748
Decision Tree	0.693	0.642	0.667	0.682	0.637
Random Forest	0.987	0.400	0.569	0.763	0.718
Liu et al. [175]	0.919	0.735	0.817	0.875	0.885
BERT	0.837	0.711	0.711	0.808	0.815
OGD-R	0.867	0.794	0.829	0.851	0.867
OGD-R + L1 Regularisation	0.880	0.759	0.815	0.853	0.857
OGD-R + L2 Regularisation	0.896	0.783	0.835	0.871	0.890
OGD-T	0.900	0.871	0.886	0.894	0.940
OGD-T + L1 Regularisation	0.885	0.881	0.883	0.883	0.940
OGD-T + L2 Regularisation	0.913	0.865	0.888	0.903	0.941
Augm. OGD-R	0.930	0.777	0.847	0.895	0.924
Augm. OGD-T	0.953	0.853	**0.900**	**0.931**	**0.948**

Table 9.3: *Area under PR Gain curves for both base and augmented models.*

Model	Base	Augmented
OGD-R	0.99903	0.99983
OGD-T	0.99987	0.99988

ence[2]. This examination is vital to understand the grounds of language bias captured by the word variants and should be performed for any new application of the selective word normalisation.

In the CL knowledge integration paradigm, the goal was not to reinforce any language bias but to exploit it towards a selective text normalisation that reduces variance within the corpus while preserving the bias. In effect, this may indeed make the bias more prominent. When swapping the word variants, the models augmented by selective word normalisation (i.e. knowledge on word variants) only see a more substantial drop in performance than base models, indicating that this CL knowledge integration makes the models more sensitive to the language used.

However, in this study, the proposed strategies capitalise on two key aspects: the low-level language used (through knowledge on variants) and the high-level use of conversational strategies (through knowledge on themes). Therefore, the fully augmented models are sensitive to both aspects, which helps counterbalance possible bias to a language associated with a socio-economic group. Indeed, when evaluated on the variant-swapped testing corpus, the fully augmented models are less sensitive to the effect of swapping variants than when using selective text normalisation only.

[2]Common examples of this occurs through intentional misspellings, such as *wowww* $\rightarrow$ *wow*, *whaaaaat* $\rightarrow$ *what*, *wayyyy* $\rightarrow$ *way*, *thnx* $\rightarrow$ *thanks*, and *thingz* $\rightarrow$ *things*.

Table 9.4: *Progressive additions of CL-augmentations to a simple LSTM model similar to OGD-R with no pre-training of WSR*

Method	Accuracy	Precision	Recall	AUPR	F_1	$F_{0.5}$
Standard LSTM	98.96	0.850	0.741	0.808	0.792	0.826
+ Superv. & excit. LSTM	99.2	0.933	0.757	0.872	0.836	0.891
+ Elastic pulling	99.22	0.913	0.783	0.883	0.843	0.884
+ Superv. & excit. attn	99.21	0.915	0.779	0.913	0.841	0.884

Table 9.5: *Performance of base models and their augmented counterpart when swapping word variants in the testing data.*

Architecture	Precision	Recall	AUPR	F_1	$F_{0.5}$
OGD-R	0.921	0.497	0.779	0.646	0.787
Augmented OGD-R	0.943	0.713	0.898	0.812	0.885
OGD-T	0.924	0.851	0.940	0.886	0.909
Augmented OGD-T	0.953	0.853	0.947	0.900	0.931

To further reduce the risk of bias towards a socio-economic group, for use in a given socio-economic context, it would be possible to train the models using conversations from this group only, at least for the negative class.

9.1.1 Conclusion of Combining Corpus Linguistics and Deep Learning

This case study has explored the integration of prior CL knowledge into a DNN. It considered two types of CL knowledge: 1) variants of semantically equivalent words that are, or not, discriminative of OG, and which were used to perform a selective text normalisation in support of the classification. Existing normalisation methods would apply to full text with no such distinction, thus failing to provide this support. 2) The identification of some OG processes and their associated language, which was used to focus the attention of the DNN on subtle language clues. This work compared several integration approaches, including a new method for stimulating an LSTM's attention directly through its input gates, without the need for an external attention mechanism. For the final augmented model, a gentle pulling method was selected for selective text normalisation as well as a combination of auxiliary tasks and supervision and direct stimulation for stimulated attention and stimulated LSTM, whose benefits combined to produce the state-of-the-art for OG detection. The general applicability of our approaches was demonstrated using two architecture types of recurrent and transformer neural networks and two-word embeddings of different complexities. The results have shown the performance improvements over the base and state-of-the-art models for both architectures while allowing for a CL based interpretation of the classification decision through visualisation of predicted OG processes and DNN's attention.

Using the integration strategies for prior knowledge improves both base models OGD-R and OGD-T, highlighting the generalisability of said strategies for different architectural types of DNNs. The extensive range of experiments highlight:

1. Stimulated attention generally performs better with both supervision and direct stimulation.

2. Auxiliary outputs improve performance while providing an additional means of visualisation.

3. Elastic Pull is an effective method for modifying the positions of word vectors in existing or pre-trained WSRs. While this has been shown for OG detection, we encourage the use of these methods for other language tasks where applicable.

While the applicability of these methodologies have been demonstrated for CL knowledge on OG, there is a potential for other domains that use similar representations or model architectures. For instance, the selective text normalisation that we propose is more generally applicable to other classification tasks from chat conversations, and its proposed implementations are usable on other DNNs that use word embeddings to capture word semantic. The decomposition of conversations into sub-goals may be obtained from CL studies on other applications. For example, a large corpus of works have identified strategies for persuasion and manipulation in extreme ideology groups, e.g. [214, 215, 216, 217] for radical right hate speech and [218] for jihadi radicalisation. Some of the proposed strategies may also allow the integration of other (non-CL) domain knowledge. It may be generally helpful to estimate auxiliary quantities that are known to be relevant to the task and may usefully constrain the DNN's attention and learnt features. The two other methods for focusing the DNN's attention (i.e. stimulated attention and stimulated LSTM input gates) may be generally used to weight more critical elements of the training data.

Perspectives for OG prevention – The proposed OG classification method has been designed based on requirements from specialised law enforcement to assist in the investigation of large quantities of chat logs. Its intended usage is to facilitate triage by law enforcement of digital materials seized from suspected offenders after enough evidence allowed launching the procedure. Flagged conversations are to be investigated more thoroughly by a trained human operator following law enforcement's strict robustness and security protocols to ensure integrity. Within this usage scenario, there is, therefore, no risk of innocents being automatically prosecuted. However, this may happen should the tool be used in an unintended usage scenario, as its detection results might bias the decision of a human operator. This work aims not to address the possible biases in human decisions, which are addressed by law enforcement's protocols. Accordingly, mitigation measures should be put in place to avoid intentional and non-intentional misuse of the system, but these are outside the scope of this work. Nevertheless, a statement may accompany the deployment of this system that includes accuracy rates to inform users of risks of false detection and alert them that the system's flagging of a conversation should not be taken as evidence. However, it is worth noting that the system may allow a more thorough and fairer investigation of the flagged conversations through the proposed visualisation that helps to focus on crucial aspects of the conversation, together with the reduced workload and associated lowered time pressure.

Table 9.6: *Impact of each domain knowledge integration strategy on MPNN (top) and SchNet (bottom) for Augm. QM9. Results are in the format: mean (std). The specialised interaction methods that optimise at best energy and geometry are highlighted in bold for each GNN.*

Strategy		AE	RE	DSG
MPNN base model with no BT information		0.242 (1.318)	0.0029 (0.015)	0.034 (0.074)
	MPNN-BTF	0.091 (0.476)	0.0012 (0.005)	0.039 (0.056)
Specialised proc. channels	(Eq. 6.7) ($\alpha = 0.624$)	0.272 (1.293)	0.0034 (0.014)	0.051 (0.080)
	(Eq. 6.9) scalar λ_r ($\alpha = 0.628$)	0.122 (0.431)	0.0016 (0.005)	0.032 (0.057)
	(Eq. 6.9) vector λ_r ($\alpha = 0.627$)	0.105 (0.502)	0.0013 (0.005)	0.050 (0.049)
	(Eq. 6.10) impl. 1) ($\alpha = 0.615$)	**0.106 (0.253)**	**0.0014 (0.003)**	**0.023 (0.048)**
	(Eq. 6.10) impl. 2) ($\alpha = 0.635$)	**0.073 (0.170)**	**0.0010 (0.002)**	**0.030 (0.048)**
	(Eq. 6.10) impl. 3) ($\alpha = 0.618$)	0.131 (0.436)	0.0017 (0.005)	0.025 (0.055)
Indirect feature spec.	# atoms of each type	0.171 (0.986)	0.0021 (0.011)	0.031 (0.046)
	# orbitals	0.195 (0.545)	0.0025 (0.006)	0.033 (0.046)
	distance to stable geom.	0.119 (0.698)	0.0015 (0.008)	0.033 (0.046)
	SchNet base model	0.038 (0.037)	0.0005 (0.0005)	0.020 (0.031)
Specialised proc. channels	(Eq. 6.8) ($\alpha = 0.734$)	0.036 (0.035)	0.0005 (0.0005)	0.022 (0.032)
	(Eq. 6.9) scalar λ_r ($\alpha = 0.529$)	**0.020 (0.018)**	**0.0003 (0.0002)**	**0.025 (0.032)**
	(Eq. 6.9) vector λ_r ($\alpha = 0.570$)	0.024 (0.028)	0.0003 (0.0003)	0.034 (0.034)
	(Eq. 6.11) ($\alpha = 0.727$)	0.031 (0.032)	0.0004 (0.0004)	0.015 (0.028)
Indirect feature spec.	# atoms of each type	0.038 (0.036)	0.0005 (0.0005)	0.032 (0.033)
	# orbitals	0.036 (0.035)	0.0005 (0.0005)	0.022 (0.032)
	distance to stable geom.	0.049 (0.044)	0.0007 (0.0006)	0.037 (0.033)

9.2 Chemical Energy Estimation with Deep Learning

A highlight of the individual prior knowledge augmentations applied to the base models is shown in Table 9.6. From these results, it was observed that both the specialised information processing and the indirection specialisation through auxiliary tasks can be beneficial for the production of the augmented models.

All the physics integration methods were combined into final Augmented-MPNN and Augmented-SchNet DNNs. For Augmented-MPNN Equation 6.10 implementation 2 was used, and Equation 6.9 scalar was used for Augmented-SchNet. These types of specialised information processing methods were chosen due to their best performance in both AE and DSG metrics. For both augmented models, all indirect feature specialisations were selected. All models (both augmented and non-augmented) were then tested on the different scenarios of the three datasets[3].

The results of combining all augmentations for all chemical datasets are shown in Table 9.7. Firstly, when compared to the individual integrations, we observed that it was beneficial to combine all auxiliary estimations and BT information.

When using the Augmented-QM9 dataset, SchNet out-performed MPNN, possibly due

[3]For the Growing Crystal dataset, we consider system sizes from 15 to 75 atoms instead of the maximum 114 due to memory restrictions in MPNN.

Table 9.7: *Evaluation of the base (left) and augmented (right) models. For each DNN, best results between base and augmented models are highlighted in bold.*

		Base		Augmented	
Model	Dataset	AE	DSG	AE	DSG
MPNN	Augm. QM9	0.24 (1.32)	0.034 (0.074)	**0.07 (0.20)**	**0.033 (0.047)**
	Inf. Crystal	0.04 (0.03)	0.201 (0.211)	**0.03 (0.04)**	**0.048 (0.035)**
	SCG	2.80 (3.80)	–	**0.51 (0.53)**	–
	UCG	2.89 (3.82)	0.532 (0.595)	**2.39 (3.56)**	**0.162 (0.2367)**
SchNet	Augm. QM9	0.04 (0.04)	**0.020 (0.031)**	**0.02 (0.02)**	0.021 (0.031)
	Inf. Crystal	**13.44 (15.98)**	**0.214 (0.357)**	13.68 (17.54)	0.311 (0.180)
	SCG	2.02 (1.83)	–	**0.3 (0.3)**	–
	UCG	5.86 (4.38)	0.569 (0.604)	**1.1 (1.0)**	**0.528 (0.623)**

to the better accounting of long-range atomic interactions and more efficient operations. The same observation applied to the SCG subset, while SchNet performed worse than base MPNN on UCG. Furthermore, SchNet performed particularly poorly on Infinite Crystals. The reason why the CNN cannot handle this dataset is to be investigated in future work, but it may be related to the fact that SchNet struggles to handle small periodic chemical systems. Both base (non-augmented) MPNN and SchNet produced worse results on the Growing Crystal datasets than Augmented-QM9, which is likely due to the wider variety of geometries and number of atoms in the dataset. For example, while Augmented-QM9 contains chemicals of up to 29 atoms (only one chemical at this size), the Growing Crystal datasets each span up to 75 atoms with random placements of atoms. These differences increase the chemical complexities with which the models are expected to learn from. Base MPNN did slightly better than base SchNet on USG, but SchNet benefits more from our integration of physics knowledge and Augmented SchNet obtained the best results on this subset.

Both MPNN and SchNet benefited from the prior integration strategies. Overall, the augmented DNNs performed as well (2 cases) or better (all other cases) than their base models at finding stable geometries. The improvement was particularly strong on the Growing Crystal datasets, hinting that the augmented models were able to generalise better to new geometries of this dataset. Consequently, on the Infinite Crystal dataset, which contains only two mono-atomic infinite crystals, we expect to see further improvements in future works by considering structures outside the fcc lattice and a wider variety of atom types. On the UCG subset, where base SchNet under-performed at energy estimation, the CNN benefited more from the integration of physics knowledge than MPNN, and Augmented-SchNet obtained the best energy estimations on this subset. On the other hand, still on the same subset, MPNN's optimisation of stable geometries benefited more from the augmentation and produces the best DSG results. These successful augmentations reinforced the two DNNs' respective advantages at estimating accurate energies and stable geometries.

MPNN suffered from a high std, especially for DSG. SchNet also had high std on Augmented QM9 and Crystal Growth. Upon examination, this might be explained by some

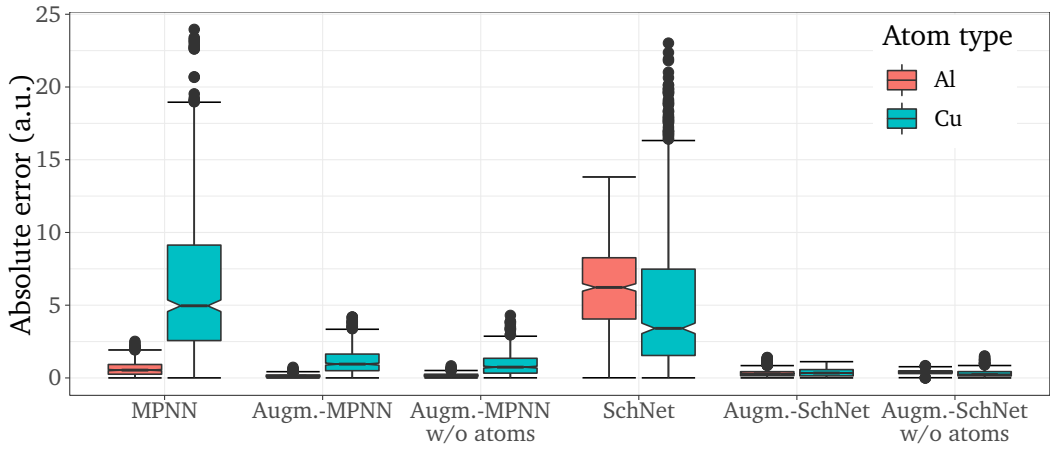


Figure 9.1: *Performance of base and augmented models on the UCG dataset. For the augmented models, we test with all augmentations and without the auxiliary atom type prediction.*

difficult cases[4] where estimations were close to the true energy values but do not contain a minimum. On the other hand, while the energy estimates of the augmented models were slightly worsened by an offset on these cases, they still account for the repulsive effect of atoms getting close to each other to predict an energy minimum at a reasonable geometry. The energy offset may be addressed in future works, e.g. by a different loss term or new auxiliary estimates.

9.2.1 Ablation Study

In testing the effectiveness of combining specialised information processing and indirect feature specialisation, an ablation study was performed using the UCG dataset. As the indirect specialisations aimed at estimating the atom-level features (number of atoms of each type and number of interacting electrons), the auxiliary predictions of the number of atoms were removed. Firstly, in Figure 9.1, it is shown the augmentations have an observable effect on the performance of the GNNs. For each of the GNNs, there is a consistent pattern where estimations are more accurate for Al crystals than for Cu, possibly due to the larger energy values seen in Cu. When the auxiliary prediction for atom types were removed from the augmented models, however, the absolute errors for Cu were slightly improved. The reason for this effect is unknown, but perhaps suggests the weighting of the auxiliary tasks can be optimised as the higher errors of Cu may be overpowering the auxiliary tasks in the loss during training.

To understand if this effect only occurs for the atom type auxiliary task, a complete ablation study with all augmentations was performed, and the results are shown in Table 9.8. For Al crystals, the auxiliary tasks were helping produce a more accurate energy estimate (as the results are worsened with the task removed), while for Cu crystals, the methods make the estimate slightly worse. As this pattern occurs for every auxiliary task, it further suggests we may optimise the weighting of tasks during training. Despite the slightly worsened energy estimates, we observed that the DSG measure

[4]These more difficult cases differ between models, and their cause will be investigated in future work.

Table 9.8: *Ablation study and effect on handling different atom types in crystals of UCG*

	Al		Cu	
Method	AE	DSG	AE	DSG
MPNN	0.643 (0.495)	0.428 (0.480)	6.250 (4.772)	0.566 (0.514)
Augm.-MPNN	0.150 (0.120)	0.111 (0.135)	1.160 (0.878)	0.361 (0.410)
Augm.-MPNN w/o spec. interaction	0.180 (0.144)	0.178 (0.130)	0.638 (0.582)	0.237 (0.236)
Augm.-MPNN w/o aux. # atoms	0.179 (0.129)	0.040 (0.098)	0.928 (0.760)	0.328 (0.419)
Augm.-MPNN w/o aux. # orbitals	0.190 (0.141)	0.071 (0.121)	0.906 (0.724)	0.267 (0.245)
Augm.-MPNN w/o aux. DSG	0.177 (0.151)	0.124 (0.136)	0.834 (0.680)	0.276 (0.340)
SchNet	6.357 (3.159)	0.803 (0.592)	5.355 (5.278)	0.336 (0.525)
Augm.-SchNet	0.333 (0.247)	0.246 (0.254)	0.368 (0.259)	0.006 (0.025)
Augm.-SchNet w/o spec. interaction	0.444 (0.358)	0.287 (0.257)	1.424 (1.490)	0.113 (0.200)
Augm.-SchNet w/o aux. # atoms	0.395 (0.169)	0.365 (0.085)	0.304 (0.238)	0.004 (0.029)
Augm.-SchNet w/o aux. # orbitals	0.241 (0.146)	0.022 (0.073)	0.286 (0.176)	0.179 (0.102)
Augm.-SchNet w/o aux. DSG	0.296 (0.203)	0.125 (0.174)	0.328 (0.220)	0.000 (0.000)

would sometimes improve as a result of using the auxiliary atom type predictions. For example, we see the Augmented SchNet DSG result increased from 0.246 to 0.365 when the auxiliary atom types were removed.

9.2.2 Effectiveness for Estimating Stable Configurations

The task for the GNNs was to estimate stable configurations, i.e. configurations in which the energy of the system is at its minimum potential state. While for the results, we have used DSG as a metric to give a quantitative measure of the capability of each network in carrying out this task, the predicted energy curves were visualised to evaluate each model's effectiveness for this task quantitatively.

To quantitatively evaluate the performance of the DNNs, we selected examples of the lowest and highest energy errors for different type of molecules. In Figure 9.2 (top), all GNNs (augmented and non-augmented) were able to closely match the simulated energy curves for this molecule from the Augmented-QM9 dataset. However, for other molecules (Figure 9.2 bottom), the non-augmented versions of the GNNs failed to create plausible energy curves as it doesn't have a correctly placed minimum. In such cases, the prior knowledge augmentations can rectify the deficiencies of their non-augmented counterparts.

9.2.3 Generalisation to Larger Systems

The models' ability to scale to larger systems was evaluated using the UCG subset. The increasing energy errors for larger system sizes (Figure 9.4 left) suggest that the models struggle with handling too many and long-range interactions (although the CNN performs better than the graph-DNN overall). When examining examples of such

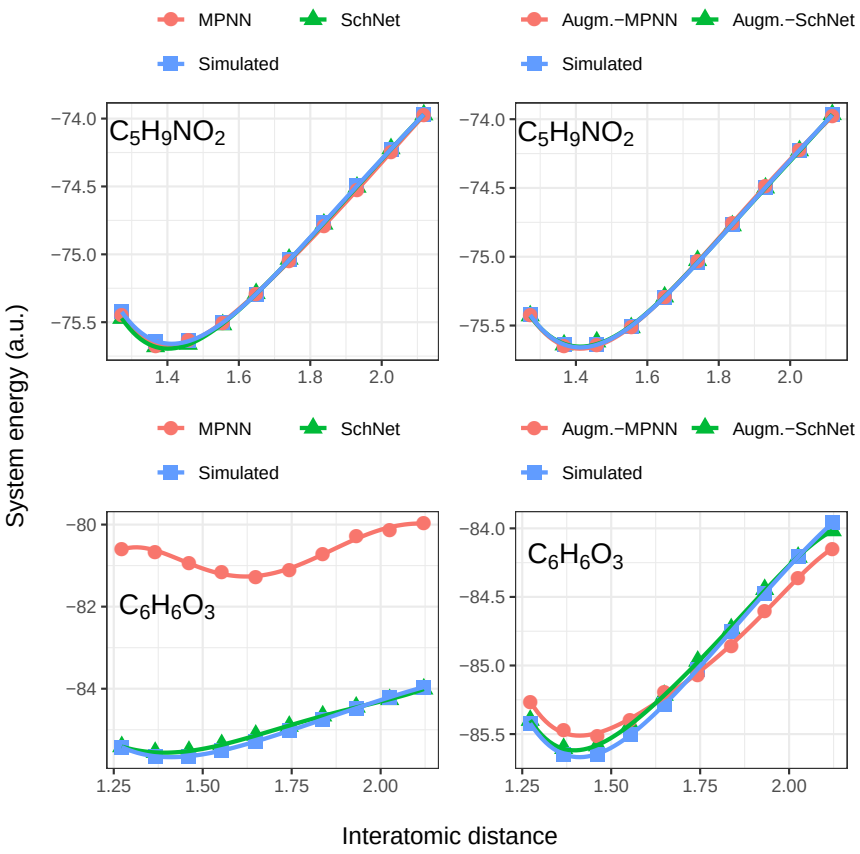


Figure 9.2: *Energy estimations at different scalings of picked molecules for base (left) and augmented (right) GNNs. Top:* $C_5H_9NO_2$*, bottom:* $C_6H_6O_3$*.*

systems (Figure 9.3), we observe energy minima at correct stable geometries for small systems (e.g. 15 atoms) even when the energy estimations have an offset. However, in extreme cases (e.g. 75 atoms), the predicted energies no longer have a correctly located minimum. This failure to correctly learn energy curves at a higher number of atoms highlights our models' current limitation and indicates that further work is needed towards better accounting for atomic interactions in large structures.

Figure 9.4 presents energy AE on UCG as a function of system size for base and augmented GNNs when trained on all system sizes (up to 75 atoms) or on small systems only (up to 25 atoms), keeping the testing set equal. For Augmented-SchNet, the better accuracy from the augmentations maintains the AE in the best achieved range when training on small systems only. For Augmented-MPNN, when training on small systems only, the range of AE was also improved by a factor ∼100 compared to base MPNN. Furthermore, accuracy correlates with system size only marginally stronger than when training on all sizes (Pearson coef. 0.200 against 0.193). This was an improvement from base MPNN, where Pearson correlation increased from 0.171 to 0.364. Therefore, the augmentations has allowed MPNN for learning of some basic principles about the atomic interactions that are applicable to larger, unfamiliar systems. When using the specialised interaction method of Equation 6.9 with scalar λ_r, Pearson correlation was similar at 0.224, and the range of AE was improved by a factor ∼400. We explain

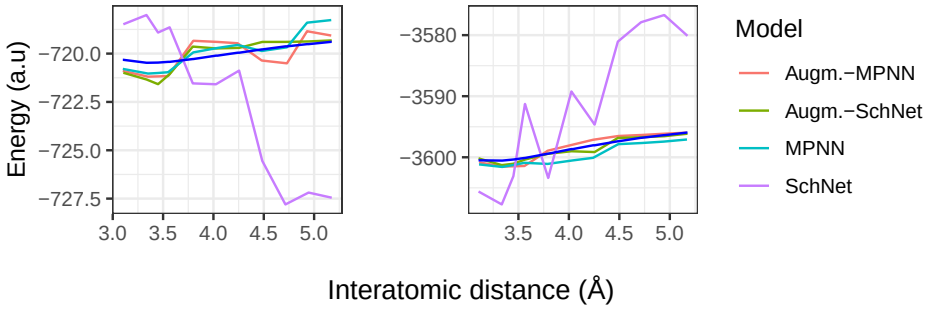


Figure 9.3: *Energy estimations (red for Augm.-MPNN, green for MPNN) at different scalings for a small (15 atoms, left) and large (75 atoms, right) crystalline system. Blue is ground truth.*

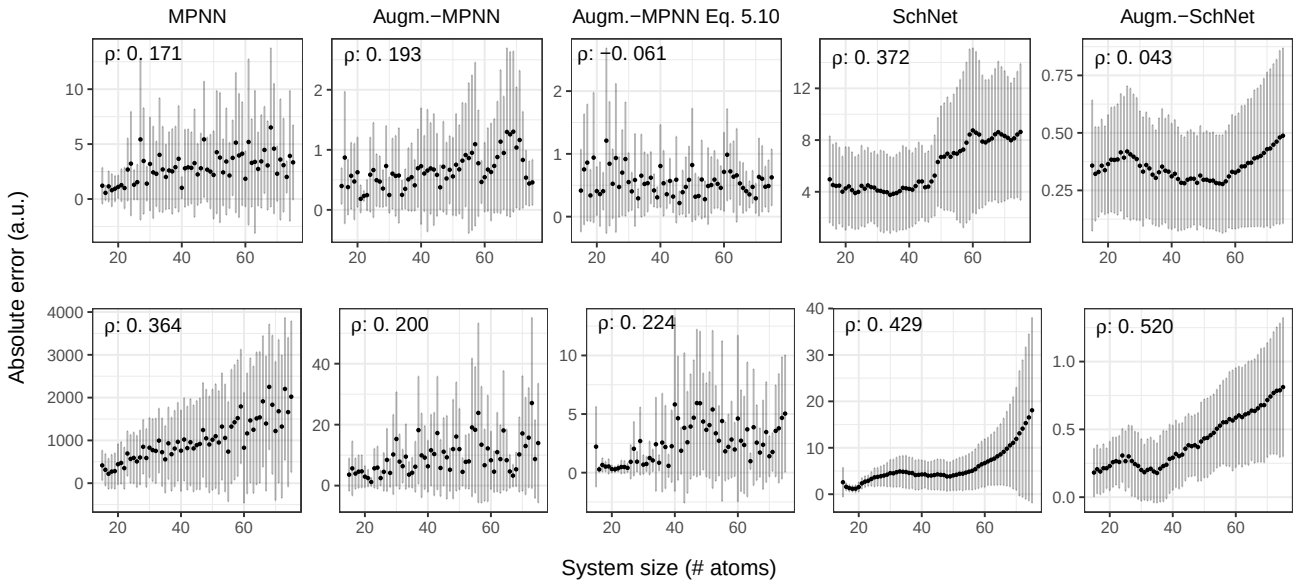


Figure 9.4: *Impact of system size on accuracy when trained on systems of up to 75 (top) and 25 (bottom) atoms from UCG.*

this better AE by the different complexities of Equations 6.10 and 6.9, which made the model more or less robust to smaller training sets.

9.2.4 Interpretation of the Atoms' Hidden States

Since the DNNs do not precisely simulate the physics of atomic interaction, investigating their learnt representations may help understanding how to interpret and trust their estimates and ability to generalise to new systems. We may visualise the hidden states of the atoms during the readout phase of the graph-DNN and after sum pooling for the CNN. These steps both conveniently reduce the atoms' vectors to scalar values before producing the energy estimate. After normalising across all atoms, these reduced values may be considered the *atoms' contributions* to the energy estimate.

After training Augmented-SchNet on UCG, we considered an atom newly added to a (stable) crystal seed of 14 atoms. When scaling the distance of this (single) atom to

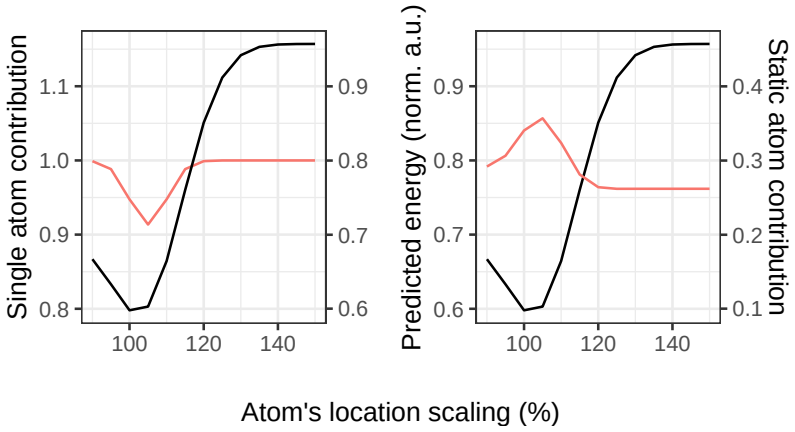


Figure 9.5: *Contribution of a moving atom (left, red curve) and mean contribution of static atoms (right, red curve) toward the energy estimate of a crystal (black curves) by Augmented-SchNet.*

the rest of the crystal, we examined its contribution and that of the rest of the (static) atoms (Figure 9.5)[5]. At its stable location, the atom's contribution was marginal while static atoms contribute maximally, in line with the crystal being at its minimum energy driven by its regular structure. As the atom was moved closer or pulled away, its contribution increases simultaneously with energy. At the same time, the relative contributions of static atoms slightly decreased to give way to the perturbation of the moving atom. This strongly suggested that the model learnt to pay attention to the location of individual atoms, although it was trained on systems that are isometrically scaled. This confirmed the previous experiment's conclusion that Augmented-SchNet learnt transferable principles of atomic interaction.

This visualisation method could further inform improvements in the model, such as the issues mentioned earlier of large range atomic interactions. It may also be helpful in the simulation of crystal growth where a lattice is not imposed by pinpointing stable atomic positions that entail equal contributions of all atoms to the system's energy.

9.2.5 Conclusion of Combining Chemical Physics Knowledge and Deep Learning

Physics knowledge has been integrated into DNNs to improve its accuracy and generalisation to estimate potential energies and stable geometries of both molecules and crystals. Two strategies were proposed for the physics knowledge integration, namely (1) specialised information passing within a DNN to better account for interaction types within the system, and (2) further relating the learnt representations to the underlying physics of the studied phenomenon using indirect feature specialisation. These methods were incorporated into two architecture types, graph-based and CNN, to form the new state-of-the-art. These models have different advantages and drawbacks, with the augmented CNN favouring accuracy of energy estimation and the augmented graph-DNN obtaining more stable geometries and better generalisation and

[5]This scenario differs from the previous whole system scaling, but our models produce plausible energy estimates.

scalability. These models proved to be able to learn the principles of atomic interactions, including capturing the contributions of single atoms to the total energy. These principles have allowed handling new geometries, including new chemical systems of varying sizes and new perturbations of their atoms' locations. When working with crystals, the models can also train on smaller datasets, while generalising better to larger systems to their non-augmented counterparts. More work on long-range atomic interactions may further help with handling larger chemical systems.

Perspective for chemistry research and material engineering – DNNs may transform the current practice in chemistry research and material engineering by providing considerably faster estimations than traditional numerical simulations. Thus, the models may be used for a fast scan of an extensive range of conditions and atomic configurations. In addition, they may be used straightforwardly for estimating a large range of physical properties of chemical systems and discovering new materials. Finally, their improved capturing of principles of atomic interaction may also allow the simulation of crystal growth in future works to further help with material discovery. However, DNNs cannot always guarantee sufficiently chemically accurate results for unseen molecules and suffer from well known and documented problems such as adversarial examples. Therefore, although our methods improve their robustness, numerical simulations remain necessary to verify a DNN's output.

9.3 Solar Burst Detection with Deep Learning

In previous chapters, methods were designed to provide auxiliary predictions and perform adversarial training with adaptive neighbourhoods. Though the auxiliary predictions have not had an impact on the detection results (which may be addressed in future work by performing the auxiliary predictions from the proposals instead of intermediate feature network), we used the adaptive neighbourhoods as a form of defence, as well as the data-specific augmentations to enhance the WAVES dataset for training.

9.3.1 Comparison with Baselines

With the best performing data augmentations, and the prior integration strategies presented in the preceding chapters, the final Augmented Faster R-CNN detector was created. This detector used the following strategies:

- Data augmentation using best performing augmentations: **D2**, **D3** and **D4**.
- Adversarial training using FGSM+AN to modulate the intensity of attack.

To create our final detector, we add each strategy onto the base Faster R-CNN in series, thereby allowing us to evaluate the contributive effect of each augmentation to the final performance metrics. This final augmented detector is compared against the baseline detector with no augmentations, and the current state-of-the-art model for

Table 9.9: *Performance on WAVES dataset for different models. With exception of [2], whose model type is determinisic, metrics are reported over 10 trials in the format: mean (standard deviation).*

Model	Precision	Recall	F_1 score	IoU	IoU (Type II only)
[2]	0.247 (0.000)	0.550 (0.000)	0.341 (0.000)	**0.724 (0.000)**	0.273 (0.000)
Base Faster R-CNN	0.335 (0.090)	0.350 (0.043)	0.468 (0.044)	0.667 (0.003)	0.195 (0.000)
+ Data augmentation	0.514 (0.124)	**0.715 (0.200)**	0.562 (0.097)	0.628 (0.064)	**0.281 (0.057)**
+ Adv. training	**0.540 (0.034)**	0.626 (0.051)	**0.577 (0.008)**	0.671 (0.018)	0.274 (0.014)

Type II burst detection using the WAVES dataset from Jenkins et al. [2]. The non-DL model of [2] used a curved region of interest (ROI) to integrate the knowledge of physics of Type II drift rates. These curved ROI followed the drift pattern of Type II bursts and flattens the representation, thereby enabling the use a rectangular sliding window detection. At each step of the sliding window, the Histogram of oriented Gradients (HOG) were computed, and used as inputs to a logistic regressor to create the detection of bursts.

The performance of each detector model is shown in Table 9.9. Firstly, we observed the detector of Jenkins et al. [2], to although provide a high IoU score relative to the DNN-based models, created more false-positives resulting in a lower precision and overall F_1 score. The base Faster R-CNN detector produced a higher F_1 score than [2], but perhaps due to the small amount of data needed to train a DNN model, the localisation as shown by the IoU was not as effective. When we used data augmentations with Faster R-CNN, we found the F_1 score to increase further to 0.562, an improvement over the base Faster R-CNN with 0.468. While the IoU score did decrease as a result of the data augmentations, we found the IoU scores for the Type II classes to improve from 0.195 to 0.281. Finally, by adding the adversarial training regime with FGSM+AN, the Faster R-CNN produced the best F_1 score thus far with 0.577 as well as matching the IoU Type II score with 0.274.

From these results, we observed how the augmentations have enabled the Faster R-CNN to produce the state-of-the-art F_1 scores, while also matching (or in the case of Type II only) IoU scores with the previous state-of-the art. To better improve this detector, further integration of domain knowledge may be used through specialised information processing or attention on data.

9.3.2 Conclusion on Combining Solar Physics Knowledge with Deep Learning

Through the combination of pre-trained weights, data augmentation, and our prior knowledge integration strategies, a state-of-the-art DNN has been created to detect Type II solar radio bursts. When compared with the previous state-of-the-art model, this new object detector is much more precise with its detection, while also spotting more Type II bursts than previous methods. This improvement is the result of augmenting the WAVES dataset through a series of data augmentations, thereby helping

the DNN train from a limited set of data. These results are further improved by the addition of adversarial training. In this method of adversarial training, adaptive neighbourhoods were created for each sample of data to allow the adversarial generation method to self-modulate its strength of perturbation with respect to the sample. This allows the adversarial algorithm to effectively improve its success of generating an adversarial and thus improve the robustness of the object detector.

Though indirect specialisation has had a small impact on the performance of this object detector, future work should consider more direct methods of integrating the prior knowledge of the morphology of Type II solar bursts. These direct methods, when integrated into the design of the DNN, may have substantial improvements over these previous methods by helping the DNN to avoid misleading features present in the WAVES dataset, such as the salient Type III bursts and background noise that fool the detector into detecting the presence of Type II bursts. Our augmentations of additional Type III bursts, and adversarial training has helped in this regard, though more methods are available to use from this thesis.

9.4 Effect of Prior Knowledge on the Amount of Required Data

One of the critical barriers to entry for practitioners to use DL is the availability of ready to use annotated datasets. Indeed, most of the research within the DL community itself revolve around existing or toy datasets to demonstrate the advances their contributions provide. However, when the research objective is not to further DL milestones but other pursuits entirely, making use of DNN becomes more troublesome.

The use of prior knowledge has been suggested as one method for *reducing* the amount of training data needed to achieve a competitive computational model [182]. Furthermore, using prior knowledge and incorporating it into the DNNs may help the learning process, and thus may also reduce the necessity of large amounts of training data to extrapolate key insights, but rather, partly rely on the insights provided by the expert in the respective field. [219] describe the general laws of ML where a model aims to learn a function $\hat{f}$ from a set of functions $\mathcal{F}$. If a model picks a function f based upon the samples from the training set, then the generalisation error depends on the number of examples as $\sqrt{\frac{VC(\mathcal{F})}{l}}$, where $VC(\mathcal{F})$ is the VC (Vapnik-Chervonekis) dimension [220] that measures the complexity and expressiveness of functions in $\mathcal{F}$, and l is the number of examples. Concerning classification algorithms, VC describes how complex the resulting model is. As the number of examples is proportional to the hypothesis class complexity, integrating prior knowledge to help the learning process may allow us to use fewer examples without much impact on performance.

9.4.1 Data-Sampling Process

To show the effect of limited data on the performance of DNNs, the training set size (while leaving the validation and testing size fixed) was reduced through a random

sampling process.

Molecular & Crystalline Datasets

The training set was reduced in 10% increments from 100% of the original size to 10%. So, for example, a single model was trained on 50% of the possible training data and then tested on 100% of the available testing data. This test was performed for the Augmented-QM9 and SCG datasets as these have thus far provided the most interesting insights into the comparative measures between the augmented and non-augmented models.

In the Augmented-QM9 dataset, there are 10k molecules of various sizes, where each molecule was compressed/expanded from 90% to 150% of the stable inter-atomic distance. To randomly sample from this dataset, it was filtered by molecular compounded. For example, when testing the performance on 50% of training, 50% of the molecules were selected from the training set. Each of these molecules includes all variations of inter-atomic distance from 90% to 150%. This sampling method contrasts SCG where the data contains only crystals at their most stable inter-atomic distance. In this dataset, all system scales were randomly sampled.

An alternate method of sampling was performed for the Augmented-QM9. Instead of sampling molecules, we sampled from the different system scalings. Therefore, all potential molecules were included in the training data, but with some scalings filtered out of the dataset.

Online Grooming Chat logs

The annotations provided by the expert linguists are available for a small portion of the positive data. Therefore, in order best find a comparative difference for using the prior knowledge integrations, we will want to ensure these annotations are available in the training data irregardless of the training set size. This dataset we have labelled set **A**.

However, this form of sampling from the training set does not vary the number of annotations. Thus in this case, OG theme detection was not tested for robustness by the smaller amounts of data. We view this as the optimal case for OG theme detection: having the largest number of annotations possible. However, we would also want to find out how OG theme detection performs when there are a limited number of annotations. For this, we create another dataset that also reduces the number of annotations. This dataset we have labelled set **B**.

In summary, from the sampling, there are two resulting datasets at different sizes from 10% to 90%:

1. **A** – Stratified sampling of grooming/non-grooming labels, but the number of grooming annotations remains at its maximum.

2. **B** – Stratified sampling of both grooming/non-grooming labels, but also grooming annotations. Thus the number of annotations is small at 10% and increases with respect to the sample size.

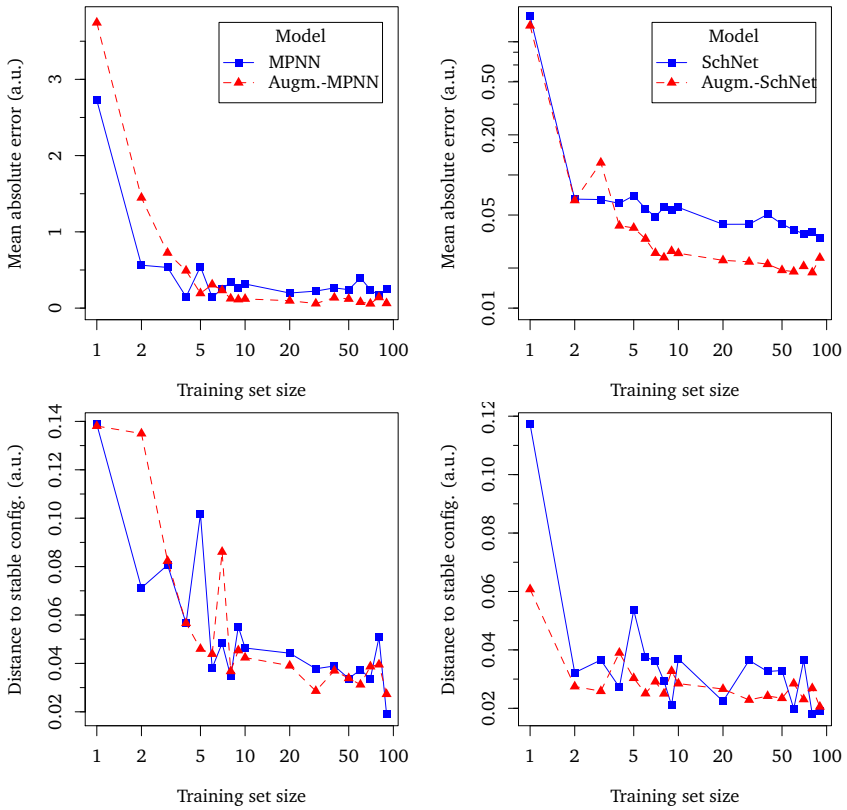


Figure 9.6: *Robustness of base and augmented models to smaller training sets for Augmented QM9, when sampling on molecule types. Top: effect on energy estimation accuracy (AE), bottom: effect on finding stable configuration (DSG). Left: MPNN, right: SchNet. Blue: base model, red: augmented model.*

Our sampling process consists of the following steps: (1) Stratified samples were created of grooming/non-grooming data at 10% intervals between 10% and 90% of the original training data. (2) At each interval, 10% of all the available training data was added to the training set, ensuring the grooming samples included all possible annotated examples to fulfil the criteria of set **A**. In cases where the number of grooming samples in the stratified sample was less than the number of annotated examples, annotated examples were randomly sampled to fit the stratified sample. (3) Again, at each interval, a stratified sample was taken of the annotations and (non)-grooming conversations, to creating set **B**.

9.4.2 Results of Energy Estimation with Limited Data

On both datasets, as expected, all models improved their energy estimates (mean AE) and their ability to find stable configurations (DSG) when training on a wider variety of molecules or crystals. However, on Augmented QM9 (Figure 9.6), both the augmented variants of MPNN and SchNet, given enough data, produced generally better energy estimations and were better at producing a plausible energy curve as shown by the DSG metric. However, these augmented models required a small amount of data (5%) in order to surpass the base models. On the SCG subset (Figure 9.7), both augmented

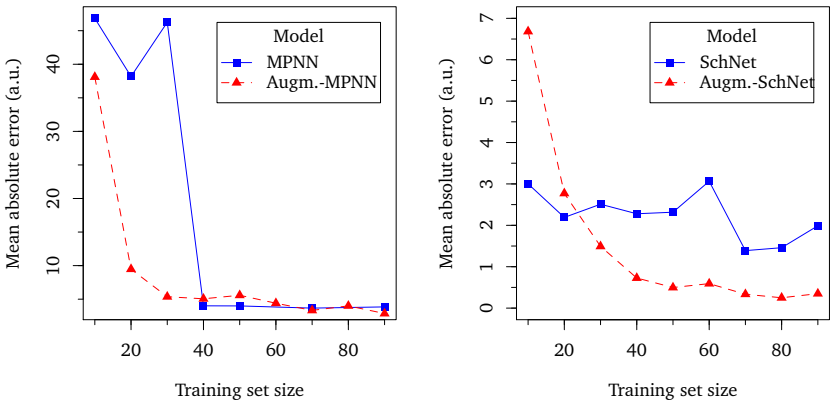


Figure 9.7: *Robustness of base and augmented models to smaller training sets for the SCG subset. Left: MPNN, right: SchNet. Blue: base model, red: augmented model.*

models did see a sharper and earlier drop in mean AE when increasing the dataset size. This indicates that, in the case of crystals, a smaller training set is required by the augmented models to reach a good performance, this is perhaps due to the augmentations providing insights to the models that did not need to be learnt from data, hence reducing the need for annotated training samples.

We also note, in two cases out of four, the non-augmented models obtained worse results than base models for extremely small training sets. This may be explained by the increased model complexity (due to additional learnt parameters and auxiliary tasks) requiring a minimal amount of data to train effectively. However, these additional complexities does not prevent the augmented models reaching adequate performance as soon as or earlier than the base models, and to outperform them with larger dataset sizes.

We further tested the robustness to smaller training sets brought by our augmentations, by training on random subsets of the compressions and dilations of all the molecules of Augmented QM9 (Figure 9.8). We find that MPNN and SchNet did not improve from training on additional scaling samples, while its augmented version is able to slightly improve its performance using the additional examples.

9.4.3 Results of Limited Online Grooming Detection Chat Logs

Both OGD-R and OGD-T were trained using the two different datasets of varying sizes. In Figure 9.9, we observed the F_1 -score improve for all models with the increasing dataset sizes. This shows that both augmented and non-augmented models are able to take advantage of the extra variety and examples that larger training set sizes affords.

However, the augmented version of OGD-R (top row) improved more quickly, and when only 30% of the available data is used, augmented OGD-R achieved its highest score yet. This quick improvement is perhaps due to the prior integration strategies helping the augmented model learn from a small amount of data. A similar effect can be seen in Figure 9.9b when using dataset **B**. However, the increases in performance for dataset **B** were less drastic as the number of annotations were smaller than in dataset

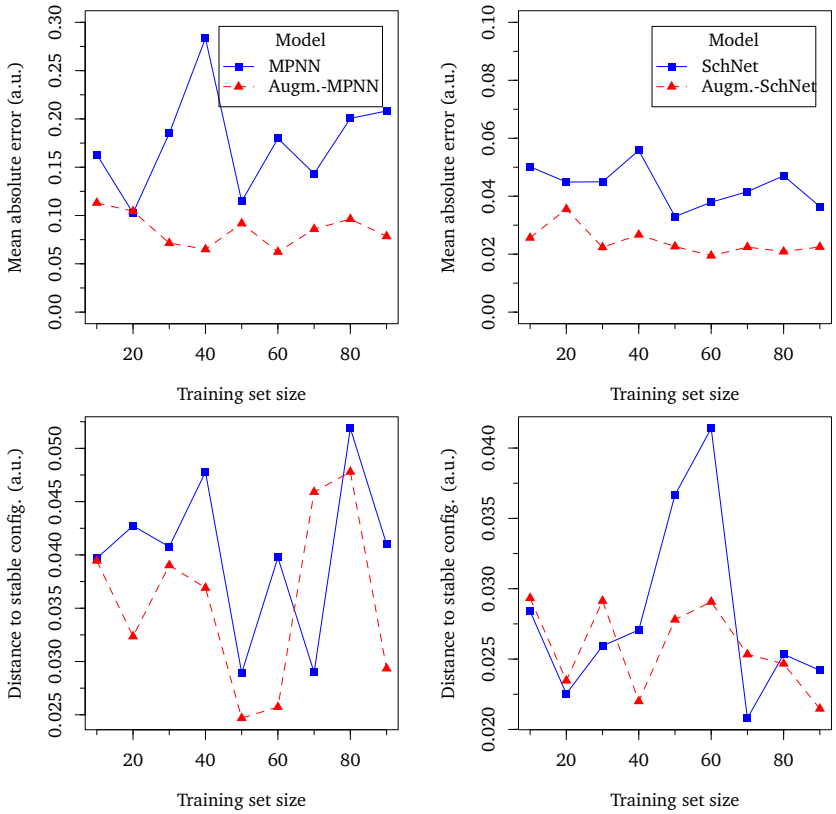
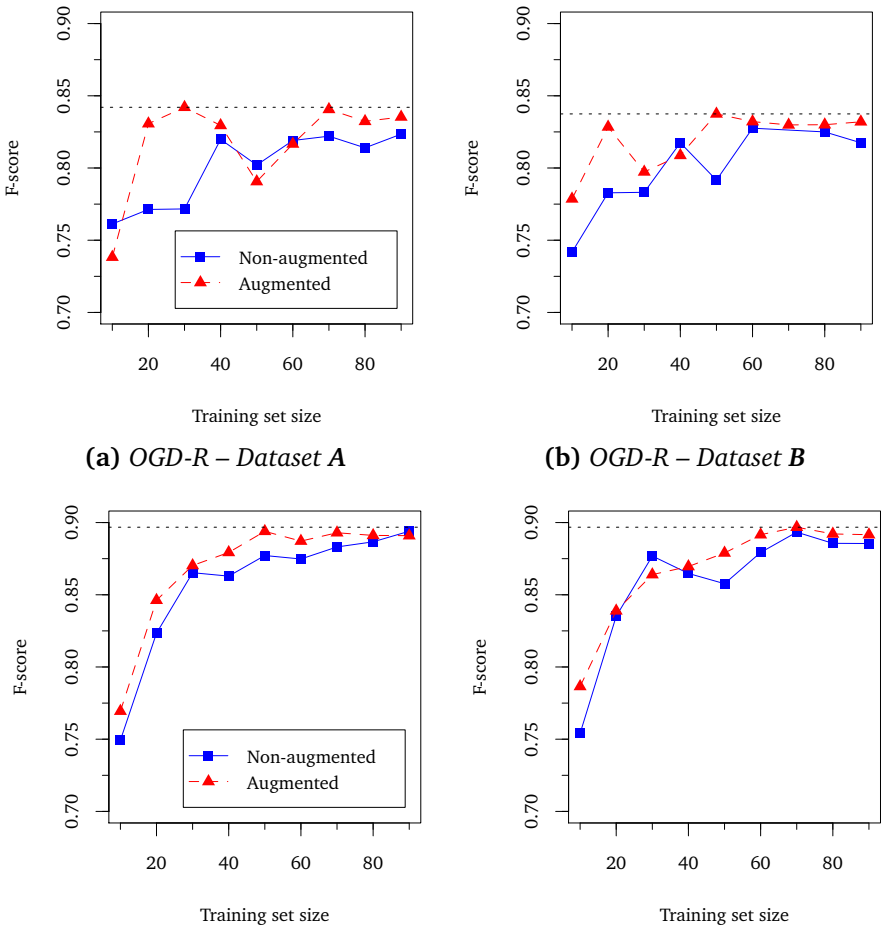


Figure 9.8: *Robustness of base and augmented models to smaller training sets for Augmented QM9, when sampling on molecule's scaling. Top: effect on energy estimation accuracy (AE), bottom: effect on finding stable configuration (DSC). Left: MPNN, right: SchNet. Blue: base model, red: augmented model.*

A and, therefore, the prior integration strategies have less annotations with which to learn from. Nevertheless, with the exception of training set size (40%), the augmented OGD-R outperformed base OGD-R highlighting the improvement that prior integration strategies affords.

Like OGD-R, the augmentations helped OGD-T (Figure 9.9 bottom row) take advantage of the annotations, thereby leading to better performance in all but one interval (90%). When all of the annotations were used (as is the case for dataset **A**, Figure 9.9c), the augmented OGD-T performed consistently better than base OGD-T. When the annotations were sampled randomly (dataset **B**, Figure 9.9d), there was less of a difference in performance between models, and in some intervals such as 30%, the non-augmented model performed better. These results demonstrate the methods reliance on the expert annotations during training.

In our experiments, we have seen the prior integration strategies help with limited sizes of data. Specifically for OGD-R, the best-achieved score occurs at 30% of the available data. As for OGD-T, while the prior integrations certainly helped improve the accuracy of the classifier, they did not necessarily benefit from small amounts of data, with both augmented and non-augmented models improving performance at the same rate. Despite this, prior integration strategies have shown to be an effective



(a) *OGD-R – Dataset **A***

(b) *OGD-R – Dataset **B***

(c) *OGD-T – Dataset **A***

(d) *OGD-T – Dataset **B***

Figure 9.9: *Performance of augmented and non-augmented OGD-R (top) and OGD-T (bottom) on varying sizes of training datasets. Dotted grey lines indicate the highest performance.*

method for situations where the amount of data is limited and consistently improve with more data.

9.5 Chapter Summary

In previous chapters, the methods by which prior knowledge can be integrated into DNNs were introduced. However, these methods were tested in isolation. This chapter has improved on the analysis combining these strategies to test their effect. The final augmented classifier or regressors were tested and compared against the current state-of-the-art in their respective domains

To further evaluate the interaction between prior integration strategies, ablation studies were performed. Finally, we evaluated the performance of the augmented and

non-augmented models with varying training set sizes, finding that prior knowledge augmentations can help boost performance when training set sizes are relatively small.

The key points from this chapter are:

- It is possible to use a combination of prior integration strategies to further enhance the performance of DNNs. Even if the strategies are built from the same priors, the DNN still increase accuracy with each additional strategy. For example, we observed an improvement when combining indirect feature specialisation with supervised/excited attention mechanisms.
- Using prior knowledge helps in situations where there is a small number of training samples. It is possible to improve the performance of DNNs by providing an inductive bias using prior knowledge.
- Through the combination of CL prior knowledge a state-of-the-art OG detector was created. This detector, while providing the state-of-the-art classification performance of OG in text-based chat logs, also includes additional output visualisations of predicted grooming communication processes that may be helpful for human decision making.
- The methods for integrating physics knowledge into GNNs can improve the models' accuracy, while also improving its effectiveness for handling OoE chemical systems, and even generalising better to systems larger than the model has been trained on. These two generalisation properties may be instrumental for generating novel chemical compounds that will be explored in future work.
- The methods for augmenting the training data enabled the production of the state-of-the-art Type II detector. The augmentations enhanced and enriched the dataset used to train the DNN.

The contributions of this chapter are:

- A demonstration of how prior knowledge integration strategies can be combined to improve upon their individual performance. We show that, even if the strategies use the same prior, DNNs can benefit for the combination of methods.
- State-of-the-art classifier for OG detection, energy estimation of OoE chemical systems, and the detection of Type II bursts.
- An evaluation of how prior knowledge can help DNNs learn with smaller datasets.

Chapter 10

Conclusion

DNNs have quickly become one of the most prolific types of model in ML. This is due, in part, to their ease of construction (through the facilitation of high-level DL frameworks), the reduction in the need to perform feature extraction or feature engineering, and their state-of-the-art performance in many tasks from many different domains, including those that could be considered safety-critical. Despite these advancements, practitioners can gain further improvements to the performance of DNNs by using their existing domain knowledge and take advantage of the strategies presented in this thesis for encoding this knowledge into the architectural design of DNNs.

The current research has aimed to identify these strategies that enable the use of prior domain knowledge for DNNs. These strategies, whether they are added to the DNN singularly, or combined, aid the DNN in producing more accurate and trustworthy predictions, and in some cases, also provide additional representations for visualisation, both essential characteristics for informed decision making.

This work has demonstrated four methods to integrate prior knowledge into the design and training of DNNs. Our proposed prior knowledge integrations are:

- **Feature specialisation** where the prior knowledge can be used to inform and shape existing feature representations of the DNN.
- **Specialised information processing** where processing channels are used to replicate concepts within the domain to learn specialised parameters that help the performance in the downstream tasks.
- **Attention on data** by attending to critical features of a section of the input.
- **Augmenting training data** where, using the prior knowledge of the data, augmentations are applied in a domain-meaningful way to enrich the existing dataset.

This manual process of augmentation is complemented by an automated method of adversarial training with adaptive neighbourhoods.

Firstly, the method of feature specialisation uses indirect or direct specialisation that modifies the learnt features to better reflect the domain knowledge. The results show that indirect feature specialisation can provide a small performance improvement, while also creating an auxiliary output that is useful for visualisation. Direct specialisation, while not providing the same useful output as indirect specialisation, benefits more (in terms of task performance) than indirect specialisation as the inductive bias is stronger. We have proposed three novel methods of applying direct specialisation to the semantic relationships of word-embeddings. However, these methods are general enough as to extend to tasks where the distance between points in a coordinate system need to be reduced or expanded to accommodate prior knowledge.

Secondly, specialised information processing provides the DNN with specialised information processing channels for each concept from the domain. Like indirect feature specialisation, these representations can be useful for visualisation and understanding what the DNN has learnt. Moreover, we proposed the combined use of MTL with specialised information processing to better represent the domain concept. In these cases auxiliary tasks can be used to provide hints to the DNN as to how to represent the domain concept.

Thirdly, attention on data helps the DNN to shift focus onto important features. By doing so, this work has shown how the attention on data method can help the DNN avoid misleading features that result in false-positives. For this method of increasing attention onto important features using prior knowledge as a guide, we proposed methods for stimulating the activations of RNNs, self-attention, and unsupervised attention mechanisms.

Finally, data augmentations can be used to enhance the training of DNNs. We demonstrated two methods (one automatic and one manual) for performing augmentations. The first method of data-specific augmentations used prior knowledge to determine domain-meaningful augmentations to enrich the variety of data the DNN can use to train. The second automatic method of adversarial training can help the DNN be more robust to small changes in the input. By combining existing adversarial generation algorithms with our proposed adaptive neighbourhoods method, the attack success of these algorithms increases due to the adaptive neighbourhoods increasing/decreasing the perturbation strength with respect to the input, and thereby helps the DNN to become more robust to perturbation than the original adversarial generation algorithm.

To compare and test these prior integration strategies, three case studies were used. The different case studies have enabled the development of implementations for a wide range of evaluation of the strategies in different use cases, and to also experiment with different DNN architectures to explore the method's generalisability to different frameworks. To facilitate the training of these DNNs for the case studies (the detection of online grooming and the energy estimation of molecular/crystalline systems in particular), novel datasets have been created, in addition to providing additional expert annotations that enable the use of the prior integration strategies. Through these case studies, we have shown our proposed methods to be helpful in improving

the performance of DNNs, while also providing means of visualisation of the indicative functions of decision process as well as the latent representations, thereby improving the trustworthiness of the DNNs.

In many of the case studies presented in this thesis, it was shown how auxiliary annotations of the positive cases (e.g. examples of OG processes and auxiliary predictions of Type II bursts) have resulted in a more accurate classifier that also reduces the number of false-positives. Based on the conclusions of our research, when using these methods, consideration should also be given to the prior knowledge of the background/negative examples, where there is further room for exploration and potential for an increase in classification accuracy.

Future work should consider expanding and adapting these methods for domains other than those demonstrated here. For instance, logic is one form of prior knowledge that has not been explored in this work. Therefore, one may investigate the use of prior domain knowledge for logic-based models or evaluate mechanisms to include first-order logic knowledge into DNNs using the paradigm presented here. However, to do so, one must consider how to efficiently store the knowledge of experts through knowledge representation.

Other work may wish to consider how these strategies incorporate negative enforcement of prior knowledge that extends further than the absence of positive annotation. For example, consider the case study of online grooming detection, where the prior knowledge is encoded as annotations of grooming processes. In this example, negative examples use only the absence of positive annotations to distinguish themselves from the positive.

Finally, our methods have shown the advantage of incorporating prior knowledge into the design of DNNs. This work demonstrates the various potential methods that researchers in scientific domains can make use of when considering how to improve the robustness, accuracy, and trustworthiness when designing and implementing DNNs. Our recommendation to these pursuits is to carefully analyse one form of prior, and explore the various means of including this prior into the model. As demonstrated in this work, one can use a combination of strategies to take full advantage of a single prior.

Bibliography

[1] M. T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015, pp. 1412–1421.

[2] J. Jenkins, A. Paiement, J. Aboudarham, and X. Bonnin, "Physics-informed detection and segmentation of type II solar radio bursts," in *Proceedings of the 31st British Machine Vision Conference*, 2020.

[3] I. J. Goodfellow, J. Shlens, and S. Christian, "Explaining and harnessing adversarial examples," in *Proceedings of the 3rd International Conference of Learning Representations*, 2015.

[4] S. Huang, N. Papernot, I. Goodfellow, Y. Duan, and P. Abbeel, "Adversarial attacks on neural network policies," *arXiv preprint arXiv:1702.02284*, 2017.

[5] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 2030–2096, 2016.

[6] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[7] L. Schulze, S. Behling, and S. Buhrs, "Automated guided vehicle systems: a driver for increased business performance," in *Proceedings of the International MultiConference of Engineers and Computer Scientists*, vol. 2, 2008.

[8] J. Lu, H. Sibai, and E. Fabry, "Adversarial examples that fool detectors," *arXiv preprint arXiv:1712.02494*, 2017.

[9] D. Song, K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, F. Tramer, A. Prakash, and T. Kohno, "Physical adversarial examples for object detectors," in *12th {USENIX} Workshop on Offensive Technologies ({WOOT} 18)*, 2018.

[10] T. B. Kristiansen, "Erroneous data and drug industry bias can impair machine learning algorithms," *The British Medical Journal*, vol. 367, 2019.

[11] D. Borchmann, "Learning terminological knowledge with high confidence from erroneous data," Ph.D. dissertation, Saechsische Landesbibliothek-Staats-und Universitaetsbibliothek Dresden, 2014.

[12] C.-M. Teng, "Correcting noisy data," in *Proceedings of the International Conference on Machine Learning*, 1999, pp. 239–248.

[13] C. Olah, A. Mordvintsev, and L. Schubert, "Feature visualization," *Distill*, vol. 2, no. 11, p. e7, 2017.

[14] J. Baxter, "A model of inductive bias learning," *Journal of Artificial Intelligence Research*, vol. 12, pp. 149–198, 2000.

[15] C. Sun, A. Shrivastava, S. Singh, and A. Gupta, "Revisiting unreasonable effectiveness of data in deep learning era," in *Proceedings of the International Conference on Computer Vision*, 2017, pp. 843–852.

[16] K. Schütt, F. Arbabzadah, S. Chmiela, K. Müller, and A. Tkatchenko, "Quantum-chemical insights from deep tensor neural networks," *Nature Communications*, vol. 8, pp. 6–13, 2017.

[17] O. Bilek and L. Skála, "From finite to infinite crystals: Analytic solution of simple tight binding model of finite sc, fcc and bcc crystals of arbitrary size," *Czechoslovak Journal of Physics*, vol. 28, no. 9, pp. 1003–1019, 1978.

[18] H. Jiang, H. Baranger, and W. Yang, "Density-functional theory simulation of large quantum dots," *Physical Review B - Condensed Matter and Materials Physics*, vol. 68, no. 16, pp. 1–9, 2003.

[19] A. Erba, J. Baima, I. Bush, R. Orlando, R. Dovesi, and A. Erba, "Large-scale condensed matter DFT simulations: Performance and capabilities of the CRYSTAL code," *Journal of Chemical Theory and Computation*, vol. 13, no. 10, pp. 5019–5027, 2017.

[20] A. Cohen, P. Mori-Sánchez, and W. Yang, "Challenges for density functional theory," *Chemical Reviews*, vol. 112, no. 1, pp. 289–320, 2012.

[21] B. Lanyon, J. Whitfield, G. Gillett, M. Goggin, M. Almeida, I. Kassal, J. D. Biamonte, M. Mohseni, B. J. Powell, M. Barbieri, A. Aspuru-Guzik, and A. G. White, "Towards quantum chemistry on a quantum computer," *Nature Chemistry*, vol. 2, no. 2, pp. 106–111, 2010.

[22] A. Gomes, C. Jacob, and L. Visscher, "Calculation of local excitations in large systems by embedding wave-function theory in density-functional theory," *Physical Chemistry Chemical Physics*, vol. 10, no. 35, pp. 5353–5362, 2008.

[23] M. Glavatskikh, J. Leguy, G. Hunault, T. Cauchy, and B. Da Mota, "Dataset's chemical diversity limits the generalizability of machine learning predictions," *Journal of Cheminformatics*, vol. 11, 2019.

[24] S. Chmiela, A. Tkatchenko, H. E. Sauceda, I. Poltavsky, K. T. Schütt, and K.-R. R. Müller, "Machine learning of accurate energy-conserving molecular force fields," *Science Advances*, vol. 3, no. 5, p. e1603015, 2017.

[25] L. C. Blum and J.-L. Reymond, "970 million druglike small molecules for virtual screening in the chemical universe database gdb-13," *Journal of the American Chemical Society*, vol. 131, no. 25, pp. 8732–8733, 2009.

[26] G. Montavon, M. Rupp, V. Gobre, A. Vazquez-Mayagoitia, K. Hansen, A. Tkatchenko, K. R. Müller, and O. Anatole Von Lilienfeld, "Machine learning of molecular electronic properties in chemical compound space," *New Journal of Physics*, vol. 15, p. 095003, 2013.

[27] T. McEnery and A. Wilson, *Corpus linguistics*. Edinburgh University Press, 2001.

[28] J. Sinclair, *Corpus, concordance, collocation*. Oxford University Press, 1991.

[29] N. Lorenzo-Dus, C. Izura, and R. Pérez-Tattam, "Understanding grooming discourse in computer-mediated environments," *Discourse, Context and Media*, vol. 12, pp. 40–50, 2016.

[30] E. Chiang and T. Grant, "Deceptive identity performance: Offender moves and multiple identities in online child abuse conversations," *Applied Linguistics*, vol. 40, no. 4, pp. 675–698, 2019.

[31] S. M. White, "Solar radio bursts and space weather," *Asian Journal of Physics*, vol. 16, pp. 189–207, 2007.

[32] H. Salmane, R. Weber, K. Abed-Meraim, K.-L. Klein, and X. Bonnin, "A method for the automated detection of solar radio bursts in dynamic spectra," *Journal of Space Weather and Space Climate*, vol. 8, p. A43, 2018.

[33] R. E. Overill, "Cosmic rays: a neglected potential threat to evidential integrity in digital forensic investigations?" in *Proceedings of the 15th International Conference on Availability, Reliability and Security*, 2020, pp. 1–6.

[34] E. Carley, "Using supervised machine learning to automatically detect type II and III solar radio bursts," in *EGU General Assembly Conference Abstracts*, 2020, p. 5109.

[35] V. V. Lobzin, I. H. Cairns, P. A. Robinson, G. Steward, and G. Patterson, "Automatic recognition of coronal type II radio bursts: the automated radio burst identification system method and first observations," *The Astrophysical Journal Letters*, vol. 710, no. 1, p. L58, 2010.

[36] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[37] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *arXiv preprint arXiv:1706.03762*, 2017.

[38] H. Schubert, *Topology*. Macdonald & Company, 1968.

[39] Ç. Gülçehre and Y. Bengio, "Knowledge matters: Importance of prior information for optimization," *Journal of Machine Learning Research*, vol. 17, pp. 1–32, 2016.

[40] S.-T. Chen, C. Cornelius, J. Martin, and D. H. Chau, "Shapeshifter: Robust physical adversarial attack on Faster R-CNN object detector," in *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2019, p. 52–68.

[41] M. Casadio, M. Daggitt, E. Komendantskaya, W. Kokke, D. Kienitz, and R. Stewart, "Property-driven training: All you (n)ever wanted to know about," *arXiv preprint arXiv:2104.01396*, 2021.

[42] K. Sokol and P. Flach, "Explainability fact sheets: a framework for systematic assessment of explainable approaches," in *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, 2020, pp. 56–67.

[43] W. Samek, T. Wiegand, and K.-R. Müller, "Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models," *arXiv preprint arXiv:1708.08296*, 2017.

[44] M. T. Ribeiro, S. Singh, and C. Guestrin, "'Why should I trust you?' Explaining the predictions of any classifier," in *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 1135–1144.

[45] X. Huang, M. Kwiatkowska, S. Wang, and M. Wu, "Safety Verification of Deep Neural Networks," in *Proceedings of the International Conference on Computer Aided Verification*, 2017, pp. 3–29.

[46] W. Xiang, P. Musau, A. A. Wild, D. Manzanas, L. Nathaniel, H. X. Yang, J. Rosenfeld, and T. T. Johnson, "Verification for machine learning, autonomy, and neural networks survey," *arXiv preprint arXiv:1810.01989*, pp. 1–51, 2018.

[47] S. Khalid, T. Khalil, and S. Nasreen, "A survey of feature selection and feature extraction techniques in machine learning," in *Proceedings of the Science and Information Conference*, 2014, pp. 372–378.

[48] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and intelligent laboratory systems*, vol. 2, no. 1-3, pp. 37–52, 1987.

[49] L. Cayton, "Algorithms for manifold learning," *Univ. of California at San Diego Tech. Rep*, vol. 12, no. 1-17, p. 1, 2005.

[50] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, no. 86, pp. 2579–2605, 2008.

[51] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural computation*, vol. 15, no. 6, pp. 1373–1396, 2003.

[52] D. L. Donoho and C. Grimes, "Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data," *National Academy of Sciences*, vol. 100, no. 10, pp. 5591–5596, 2003.

[53] L. K. Saul, K. Q. Weinberger, F. Sha, J. Ham, and D. D. Lee, "Spectral methods for dimensionality reduction." *Semi-supervised learning*, vol. 3, 2006.

[54] J. M. G. Hidalgo and A. A. Díaz, Caurcel, "Combining predation heuristics and chat-like features in sexual predator identification," in *Proceedings of the Conference and Labs of the Evaluation Forum*, 2012.

[55] K. S. Jones, "A statistical interpretation of term specificity and its application in retrieval," *Journal of documentation*, vol. 28, no. 1, pp. 11–21, 1972.

[56] J. W. Pennebaker, M. E. Francis, and R. J. Booth, "Linguistic inquiry and word count: Liwc 2001," *Mahway: Lawrence Erlbaum Associates*, vol. 71, 2001.

[57] O. W. Ahmed, R. Qahwaji, T. Colak, P. A. Higgins, P. T. Gallagher, and D. S. Bloomfield, "Solar flare prediction using advanced feature extraction, machine learning, and feature selection," *Solar Physics*, vol. 283, no. 1, pp. 157–175, 2013.

[58] P. Best, M. Ferrari, M. Poupard, S. Paris, R. Marxer, H. Symonds, P. Spong, and H. Glotin, "Deep learning and domain transfer for orca vocalization detection," in *Proceedings of the International Joint Conference on Neural Networks*, 2020.

[59] S. Thulasidasan, G. Chennupati, J. Bilmes, T. Bhattacharya, and S. Michalak, "On mixup training: Improved calibration and predictive uncertainty for deep neural networks," *arXiv preprint arXiv:1905.11001*, 2019.

[60] K. Zhang, J. Xu, M. R. Min, G. Jiang, K. Pelechrinis, and H. Zhang, "Automated IT system failure prediction: A deep learning approach," in *Proceedings of the International Conference on Big Data (Big Data)*, 2016, pp. 1291–1300.

[61] A. van den Oord, S. Dieleman, and B. Schrauwen, "Deep content-based music recommendation," in *Proceedings of the 26th Advances in Neural Information Processing Systems*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds., vol. 26, 2013, pp. 2643–2651.

[62] R. Caruana, "Multitask learning," *Machine learning*, vol. 28, no. 1, pp. 41–75, 1997.

[63] S. Ruder, "An overview of multi-task learning in deep neural networks," *arXiv preprint arXiv:1706.05098*, 2017.

[64] O. Sener and V. Koltun, "Multi-task learning as multi-objective optimization," *arXiv preprint arXiv:1810.04650*, 2018.

[65] Z. Yang, R. Salakhutdinov, and W. Cohen, "Multi-task cross-lingual sequence tagging from scratch," *arXiv preprint arXiv:1603.06270*, 2016.

[66] L. Duong, T. Cohn, S. Bird, and P. Cook, "Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser," in *Proceedings of the International Joint Conference on Natural Language Processing*, vol. 2, 2015, pp. 845–850.

[67] Y. Yang and T. M. Hospedales, "Trace norm regularised deep multi-task learning," *arXiv preprint arXiv:1606.04038*, 2016.

[68] X. Sun, R. Panda, R. Feris, and K. Saenko, "Adashare: Learning what to share for efficient deep multi-task learning," *arXiv preprint arXiv:1911.12423*, 2019.

[69] T. Standley, A. Zamir, D. Chen, L. Guibas, J. Malik, and S. Savarese, "Which tasks should be learned together in multi-task learning?" in *Proceedings of the International Conference on Machine Learning*, 2020, pp. 9120–9132.

[70] M. Raissi, A. Yazdani, and G. Karniadakis, "Hidden Fluid Mechanics: A Navier-Stokes Informed Deep Learning Framework for Assimilating Flow Visualization Data," *arXiv preprint arXiv:1808.04327*, 2018.

[71] K. Schütt, P. Kindermans, H. E. S. Felix, S. Chmiela, A. Tkatchenko, and K. Müller, "Schnet: A continuous-filter convolutional neural network for modeling quantum interactions," in *Proceedings of the 30th Advances in Neural Information Processing Systems*, 2017, pp. 991–1001.

[72] L. Yang, D. Zhang, and G. E. Karniadakis, "Physics-informed generative adversarial networks for stochastic differential equations," in *arXiv preprint arXiv:1811.02033*, 2018.

[73] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proceedings of the International Conference on Machine Learning*, 2009, pp. 41–48.

[74] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[75] J. Weston, F. Ratle, H. Mobahi, and R. Collobert, "Deep learning via semi-supervised embedding," in *Neural networks: Tricks of the trade*. Sringer, 2012, pp. 639–655.

[76] N. Muralidhar, M. R. Islam, and M. Marwah, "Incorporating prior domain knowledge into deep neural networks," in *Proceedings of the 2018 IEEE International Conference on Big Data*. IEEE, 2018, pp. 36–45.

[77] Z. Hu, Z. Yang, R. Salakhutdinov, X. Liang, L. Qin, H. Dong, and E. Xing, "Deep generative models with learnable knowledge constraints," in *Proceedings of the 31st Advances in Neural Information Processing Systems*, 2018, pp. 10 501–10 512.

[78] K. Ganchev, J. Gillenwater, and B. Taskar, "Posterior regularization for structured latent variable models," *Journal of Machine Learning Research*, vol. 11, no. Jul, pp. 2001–2049, 2010.

[79] Z. Hu, X. Ma, Z. Liu, E. Hovy, and E. Xing, "Harnessing deep neural networks with logic rules," *arXiv preprint arXiv:1603.06318*, 2016.

[80] A. Bakarov, "A survey of word embeddings evaluation methods," *arXiv preprint arXiv:1801.09536*, 2018.

[81] M. Kaneko and D. Bollegala, "Gender-preserving debiasing for pre-trained word embeddings," *arXiv preprint arXiv:1906.00742*, 2019.

[82] T. Bolukbasi, K.-W. Chang, J. Zou, V. Saligrama, and A. Kalai, "Man is to computer programmer as woman is to homemaker? Debiasing word embeddings," in *Proceedings of the 29th Advances in Neural Information Processing Systems*, 2016, pp. 4349–4357.

[83] J. Zhao, T. Wang, M. Yatskar, V. Ordonez, and K.-W. Chang, "Gender bias in coreference resolution: Evaluation and debiasing methods," *arXiv preprint arXiv:1804.06876*, 2018.

[84] J. Zhao, T. Wang, M. Yatskar, R. Cotterell, V. Ordonez, and K.-W. Chang, "Gender bias in contextualized word embeddings," *arXiv preprint arXiv:1904.03310*, 2019.

[85] T. Wang, X. V. Lin, N. F. Rajani, B. McCann, V. Ordonez, and C. Xiong, "Double-hard debias: Tailoring word embeddings for gender bias mitigation," *arXiv preprint arXiv:2005.00965*, 2020.

[86] P. Lu, T. Bai, and P. Langlais, "SC-LSTM: Learning task-specific representations in multi-task learning for sequence labeling," in *Association for Computational Linguistics: Human Language Technologies*, vol. 1, 2019, pp. 2396–2406.

[87] Y.-B. Kim, K. Stratos, and R. Sarikaya, "Frustratingly easy neural domain adaptation," in *Proceedings of the International Conference on Computational Linguistics*, 2016, pp. 387–396.

[88] L. Liu, X. Ren, J. Shang, J. Peng, and J. Han, "Efficient contextualized representation: Language model pruning for sequence labeling," *arXiv preprint arXiv:1804.07827*, 2018.

[89] A. Søgaard and Y. Goldberg, "Deep multi-task learning with low level tasks supervised at lower layers," in *Association for Computational Linguistics*, vol. 2, 2016, pp. 231–235.

[90] H. M. Alonso and B. Plank, "When is multitask learning effective? semantic sequence prediction under varying data conditions," *arXiv preprint arXiv:1612.02251*, 2016.

[91] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *Proceedings of the European Semantic Web Conference*, 2018, pp. 593–607.

[92] R. Ying, J. You, C. Morris, X. Ren, W. L. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling," *arXiv preprint arXiv:1806.08804*, 2018.

[93] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P. S. Yu, "Heterogeneous graph attention network," in *Proceedings of the World Wide Web Conference*, 2019, pp. 2022–2032.

[94] D. Hershcovich, O. Abend, and A. Rappoport, "Multitask parsing across semantic representations," *arXiv preprint arXiv:1805.00287*, 2018.

[95] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, and Y. Bengio, "End-to-end attention-based large vocabulary speech recognition," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, 2016, pp. 4945–4949.

[96] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[97] M. Nguyen and T. H. Nguyen, "Who is killed by police: Introducing supervised attention for hierarchical LSTMs," in *Proceedings of the International Conference on Computational Linguistics*, 2018, pp. 2277–2287.

[98] V. Mnih, N. Heess, A. Graves, and K. Kavukcuoglu, "Recurrent models of visual attention," in *Proceedings of the 27th Advances in Neural Information Processing Systems*, 2014, pp. 2204–2212.

[99] M. Cornia, L. Baraldi, G. Serra, and R. Cucchiara, "Predicting human eye fixations via an lstm-based saliency attentive model," in *Proceedings of the IEEE Transactions on Image Processing*, vol. 27, no. 10, 2018, pp. 5142–5154.

[100] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le, "XLNet: Generalized autoregressive pretraining for language understanding," in *Proceedings of the 32nd Advances in Neural Information Processing Systems*, 2019, pp. 5754–5764.

[101] X. Zhang, T. Wang, J. Qi, H. Lu, and G. Wang, "Progressive attention guided recurrent network for salient object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 714–722.

[102] J. Kawahara, C. J. Brown, S. P. Miller, B. G. Booth, V. Chau, R. E. Grunau, J. G. Zwicker, and G. Hamarneh, "BrainNetCNN: Convolutional neural networks for brain networks; Towards predicting neurodevelopment," *NeuroImage*, vol. 146, pp. 1038–1049, 2017.

[103] A. W. Yu, D. Dohan, M.-T. Luong, R. Zhao, K. Chen, M. Norouzi, and Q. V. Le, "Qanet: Combining local convolution with global self-attention for reading comprehension," *arXiv preprint arXiv:1804.09541*, 2018.

[104] K. M. Hermann, T. Kočiskỳ, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom, "Teaching machines to read and comprehend," *arXiv preprint arXiv:1506.03340*, 2015.

[105] W. Li, K. Liu, L. Zhang, and F. Cheng, "Object detection based on an adaptive attention mechanism," *Scientific Reports*, vol. 10, no. 1, pp. 1–13, 2020.

[106] Y. Alami Mejjati, C. Richardt, J. Tompkin, D. Cosker, and K. I. Kim, "Unsupervised attention-guided image-to-image translation," in *Proceedings of the 31st Advances in Neural Information Processing Systems*, vol. 31, 2018, pp. 3693–3703.

[107] J. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," *arXiv preprint arXiv:1506.07503*, 2015.

[108] A. Manchin, E. Abbasnejad, and A. van den Hengel, "Reinforcement learning with attention that works: A self-supervised approach," in *Proceedings of the International Conference on Neural Information Processing*, 2019, pp. 223–230.

[109] A. Mott, D. Zoran, M. Chrzanowski, D. Wierstra, and D. J. Rezende, "Towards interpretable reinforcement learning using attention augmented agents," *arXiv preprint arXiv:1906.02500*, 2019.

[110] F. Perazzi, P. Krähenbühl, Y. Pritch, and A. Hornung, "Saliency filters: Contrast based filtering for salient region detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 733–740.

[111] L. Wang, H. Lu, X. Ruan, and M.-H. Yang, "Deep networks for saliency detection via local estimation and global search," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3183–3192.

[112] M. M. Derakhshani, S. Masoudnia, A. H. Shaker, and O. Mersa, "Assisted excitation of activations: A learning technique to improve object detectors," in

Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 9201–9210.

[113] H. Mi, Z. Wang, and A. Ittycheriah, "Supervised attentions for neural machine translation," *arXiv preprint arXiv:1608.00112*, 2016.

[114] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *Proceedings of the International Conference on Machine Learning*, 2015, pp. 2048–2057.

[115] N. Papernot, P. Mcdaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *Proceedings of the IEEE European Symposium on Security and Privacy*, 2016.

[116] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Proceedings of the IEEE Symposium on Security and Privacy*, 2017, pp. 39–57.

[117] S.-M. M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2016, pp. 2574–2582.

[118] X. Huang, M. Kwiatkowska, S. Wang, and M. Wu, "Safety verification of deep neural networks," in *Proceedings of the International Conference on Computer Aided Verification*, 2017, pp. 3–29.

[119] W. Ruan, X. Huang, and M. Kwiatkowska, "Reachability analysis of deep neural networks with provable guarantees," *arXiv preprint arXiv:1805.02242*, pp. 1–24, 2018.

[120] S. Zheng, Y. Song, T. Leung, and I. Goodfellow, "Improving the robustness of deep neural networks via stability training," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4480–4488.

[121] C. Guo, J. R. Gardner, Y. You, A. G. Wilson, and K. Q. Weinberger, "Simple black-box adversarial attacks," *arXiv preprint arXiv:1905.07121*, 2019.

[122] M. Wicker, X. Huang, and M. Kwiatkowska, "Feature-guided black-box safety testing of deep neural networks," in *Proceedings of the International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, 2018, pp. 408–426.

[123] S. Bach, A. Binder, G. Montavon, F. Klauschen, K. R. Müller, and W. Samek, "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation," *Public Library of Sciences ONE*, vol. 10, no. 7, pp. 1–46, 2015.

[124] K. Duncan, E. Komendantskaya, R. Stewart, and M. Lones, "Relative robustness of quantized neural networks against adversarial attacks," in *Proceedings of the 2020 International Joint Conference on Neural Networks*, 2020, pp. 1–8.

[125] D. Das, N. Mellempudi, D. Mudigere, D. Kalamkar, S. Avancha, K. Banerjee, S. Sridharan, K. Vaidyanathan, B. Kaul, E. Georganas *et al.*, "Mixed precision training of convolutional neural networks using integer operations," *arXiv preprint arXiv:1802.00930*, 2018.

[126] M. Gorsline, J. Smith, and C. Merkel, "On the adversarial robustness of quantized neural networks," *arXiv preprint arXiv:2105.00227*, 2021.

[127] Y. Zhao, H. Zhu, R. Liang, Q. Shen, S. Zhang, and K. Chen, "Seeing isn't believing: Practical adversarial attack against object detectors," *arXiv preprint arXiv:1812.10217*, 2018.

[128] Y. Huang, A. W.-K. Kong, and K.-Y. Lam, "Adversarial signboard against object detector." in *Proceedings of the British Machine Vision Conference*, 2019, p. 231.
[129] M. Lee and Z. Kolter, "On physical adversarial patches for object detection," *arXiv preprint arXiv:1906.11897*, 2019.
[130] Y. Wang, K. Wang, Z. Zhu, and F. Wang, "Adversarial attacks on faster R-CNN object detector," *Neurocomputing*, vol. 382, pp. 87–95, 2020.
[131] C. Xie, J. Wang, Z. Zhang, Y. Zhou, L. Xie, and A. Yuille, "Adversarial examples for semantic segmentation and object detection," in *Proceedings of the International Conference on Computer Vision*, 2017, pp. 1378–1387.
[132] J. Lu, T. Issaranon, and D. Forsyth, "Safetynet: Detecting and rejecting adversarial examples robustly," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017.
[133] X. Wei, S. Liang, N. Chen, and X. Cao, "Transferable adversarial attacks for image and video object detection," *arXiv preprint arXiv:1811.12641*, 2018.
[134] Q. Liao, X. Wang, B. Kong, S. Lyu, Y. Yin, Q. Song, and X. Wu, "Category-wise attack: Transferable adversarial examples for anchor free object detection," *arXiv preprint arXiv:2003.04367*, 2020.
[135] W.-Z. Dai, Q. Xu, Y. Yu, and Z.-H. Zhou, "Bridging machine learning and logical reasoning by abductive learning," in *Proceedings of the 32nd Conference on Advances in Neural Information Processing Systems*, 2019, pp. 2185–2826.
[136] A. K. Debnath, R. L. Lopez de Compadre, G. Debnath, A. J. Shusterman, and C. Hansch, "Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity," *Journal of medicinal chemistry*, vol. 34, no. 2, pp. 786–797, 1991.
[137] M. Alshahrani, M. A. Khan, O. Maddouri, A. R. Kinjo, N. Queralt-Rosinach, and R. Hoehndorf, "Neuro-symbolic representation learning on biological knowledge graphs," *Bioinformatics*, vol. 33, no. 17, pp. 2723–2730, 2017.
[138] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, 2014, pp. 701–710.
[139] M. Kulmanov, M. A. Khan, and R. Hoehndorf, "DeepGO: Predicting protein functions from sequence and interactions using a deep ontology-aware classifier," *Bioinformatics*, vol. 34, no. 4, pp. 660–668, 2018.
[140] M. Kulmanov, W. Liu-Wei, Y. Yan, and R. Hoehndorf, "EL embeddings: Geometric construction of models for the description logic EL++," in *Proceedings of the International Joint Conferences on Artificial Intelligence*, 2019, pp. 6103–6109.
[141] M. Diligenti, S. Roychowdhury, and M. Gori, "Integrating prior knowledge into deep learning," in *Proceedings of the 16th IEEE International Conference on Machine Learning and Applications*, 2017, pp. 920–923.
[142] L. Serafini and A. d. Garcez, "Logic tensor networks: Deep learning and logical reasoning from data and knowledge," *arXiv preprint arXiv:1606.04422*, 2016.
[143] F. Bianchi and P. Hitzler, "On the capabilities of logic tensor networks for deductive reasoning." in *Proceedings of the AAAI Spring Symposium: Combining Machine Learning with Knowledge Engineering*, 2019.
[144] F. Krüger, M. Nyolt, K. Yordanova, A. Hein, and T. Kirste, "Computational state space models for activity and intention recognition. A feasibility study," *Public Library of Sciences ONE*, vol. 9, no. 11, 2014.

[145] M. Fischer, M. Balunovic, D. Drachsler-Cohen, T. Gehr, C. Zhang, and M. Vechev, "Dl2: training and querying neural networks with logic," in *Proceedings of the International Conference on Machine Learning*, 2019, pp. 1931–1941.

[146] T. Khot, N. Balasubramanian, E. Gribkoff, A. Sabharwal, P. Clark, and O. Etzioni, "Markov logic networks for natural language question answering," *arXiv preprint arXiv:1507.03045*, 2015.

[147] S. Kok and P. Domingos, "Learning markov logic networks using structural motifs," in *Proceedings of the International Conference on Machine Learning*, 2010, pp. 551–558.

[148] L. Pulina and A. Tacchella, "An abstraction-refinement approach to verification of artificial neural networks," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6174 LNCS, pp. 243–257, 2010.

[149] R. Ehlers, "Formal verification of piece-wise linear feed-forward neural networks," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10482 LNCS, pp. 269–286, 2017.

[150] W. Kokke, E. Komendantskaya, D. Kienitz, R. Atkey, and D. Aspinall, "Neural networks, secure by construction," in *Proceedings of the Asian Symposium on Programming Languages and Systems*, 2020, pp. 67–85.

[151] G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer, "Reluplex: An efficient smt solver for verifying deep neural networks," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10426 LNCS, pp. 97–117, 2017.

[152] S. Dutta, S. Jha, S. Sankaranarayanan, and A. Tiwari, "Output range analysis for deep feedforward neural networks," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10811 LNCS, pp. 121–138, 2018.

[153] S. Muggleton, "Inductive logic programming," *New Generation Computing*, vol. 8, no. 4, pp. 295–318, 1991.

[154] S. Muggleton, L. De Raedt, D. Poole, I. Bratko, P. Flach, K. Inoue, and A. Srinivasan, "ILP turns 20," *Machine learning*, vol. 86, no. 1, pp. 3–23, 2012.

[155] P. Flach and A. Kakas, "Abductive and inductive reasoning: background and issues," in *Abduction and induction*. Springer, 2000, pp. 1–27.

[156] J. Denzinger, M. Fuchs, C. Goller, and S. Schulz, "Learning from previous proof experience: A survey," 1999.

[157] G. Grov, E. Komendantskaya, and A. Bundy, "A statistical relational learning challenge–extracting proof strategies from exemplar proofs," in *Proceedings of the International Conference on Machine Learning Workshop on Statistical Relational Learning*, 2012.

[158] T. Gauthier, C. Kaliszyk, and J. Urban, "Learning to reason with HOL4 tactics," *arXiv preprint arXiv:1804.00595*, 2018.

[159] T. Gransden, N. Walkinshaw, and R. Raman, "SEPIA: search for proofs using inferred automata," in *Proceedings of the International Conference on Automated Deduction*, 2015, pp. 246–255.

[160] J. Heras, E. Komendantskaya, M. Johansson, and E. Maclean, "Proof-pattern recognition and lemma discovery in acl2," in *Proceedings of the International*

Conference on Logic for Programming Artificial Intelligence and Reasoning, 2013, pp. 389–406.

[161] E. Komendantskaya, J. Heras, and G. Grov, "Machine learning in proof general: Interfacing interfaces," *arXiv preprint arXiv:1212.3618*, 2012.

[162] G. Holmes, A. Donkin, and I. H. Witten, "Weka: A machine learning workbench," in *Proceedings of ANZIIS'94-Australian New Zealand Intelligent Information Systems Conference*, 1994, pp. 357–361.

[163] D. Aspinall, "Proof General: A generic tool for proof development," in *Proceedings of the International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, 2000, pp. 38–43.

[164] J. Heras and E. Komendantskaya, "Recycling proof patterns in coq: Case studies," *Mathematics in Computer Science*, vol. 8, no. 1, pp. 99–116, 2014.

[165] S. Chmiela, H. E. Sauceda, K. R. Müller, and A. Tkatchenko, "Towards exact molecular dynamics simulations with machine-learned force fields," *Nature Communications*, vol. 9, no. 1, pp. 1–12, 2018.

[166] R. Ramakrishnan, P. O. Dral, M. Rupp, and O. A. Von Lilienfeld, "Quantum chemistry structures and properties of 134 kilo molecules," *Scientific Data*, vol. 1, pp. 1–7, 2014.

[167] J. Gilmer, S. Schoenholz, P. Riley, O. Vinyals, and G. Dahl, "Neural message passing for quantum chemistry," in *Proceedings of the International Conference on Machine Learning*, 2017, pp. 1263–1272.

[168] G. Inches and F. Crestani, "Overview of the International Sexual Predator Identification Competition at PAN-2012," in *Proceedings of the Conference and Labs of the Evaluation Forum*, vol. 30, 2012.

[169] Y. G. Cheong, A. K. Jensen, E. R. Gudnadottir, B. C. Bae, and J. Togelius, "Detecting predatory behavior in game chats," *Proceedings of the IEEE Transactions on Computational Intelligence and AI in Games*, vol. 7, no. 3, pp. 220–232, 2015.

[170] J. Parapar, D. E. Losada, and A. Barreiro, "A learning-based approach for the identification of sexual predators in chat log," in *Proceedings of the Conference and Labs of the Evaluation Forum*, 2012.

[171] A. E. Cano, M. Fernandez, and H. Alani, "Detecting Child Grooming Behaviour Patterns on Social Media," in *Proceedings of the International conference on social informatics*, 2014, pp. 412–427.

[172] C. H. Ngejane, G. Mabuza-Hocquet, J. H. Eloff, and S. Lefophane, "Mitigating online sexual grooming cybercrime on social media using machine learning: A desktop survey," in *Proceedings of the Advances in Big Data, Computing and Data Communication Systems*, 2018, pp. 1–6.

[173] M. Meyer, "Machine learning to detect online grooming," *MSc Dissertation, Uppsala University*, 2015.

[174] D. Schneevogt, E. Chiang, and T. Grant, "Do Perverted Justice chat logs contain examples of overt persuasion and sexual extortion? A research note responding to Chiang and Grant (2017, 2018)," *Language and Law= Linguagem e Direito*, vol. 5, no. 1, pp. 97–102, 2018.

[175] D. Liu, C. Y. Suen, and O. Ormandjieva, "A novel way of identifying cyber predators," in *arXiv preprint arXiv:1712.03903*, 2017.

[176] J. Pennington, R. Socher, and C. Manning, "GloVe: Global vectors for word representation," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2014.

[177] C. Mansfield, M. Sun, Y. Liu, A. Gandhe, and B. Hoffmeister, "Neural text normalization with subword units," in *North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2019.

[178] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proceedings of the 28th Advances in Neural Information Processing Systems*, vol. 28, 2015, pp. 91–99.

[179] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2961–2969.

[180] Y. Zhang and Q. Yang, "An overview of multi-task learning," *National Science Review*, vol. 5, no. 1, pp. 30–43, 2017.

[181] A. Kendall, Y. Gal, and R. Cipolla, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7482–7491.

[182] P. Niyogi, F. Girosi, and T. Poggio, "Incorporating prior information in machine learning by creating virtual examples," in *Proceedings of the IEEE*, vol. 86, no. 11, 1998, pp. 2196–2208.

[183] J. Timoshenko, C. J. Wrasman, M. Luneau, T. Shirman, M. Cargnello, S. R. Bare, J. Aizenberg, C. M. Friend, and A. I. Frenkel, "Probing atomic distributions in mono-and bimetallic nanoparticles by supervised machine learning," *Nano letters*, vol. 19, no. 1, pp. 520–529, 2018.

[184] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[185] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *arXiv preprint arXiv:1310.4546*, 2013.

[186] O. Levy, Y. Goldberg, and I. Dagan, "Improving distributional similarity with lessons learned from word embeddings," *Transactions of the Association for Computational Linguistics*, vol. 3, pp. 211–225, 2015.

[187] A. Paiement, L. Tao, M. Camplani, S. Hannuna, D. Damen, and M. Mirmehdi, "Online quality assessment of human movement from skeleton data," in *British Machine Vision Conference*, 2014, pp. 153–166.

[188] Z. Yin and Y. Shen, "On the dimensionality of word embedding," in *Proceedings of the 31st Advances in Neural Information Processing Systems*, 2018, pp. 894–905.

[189] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[190] J. Gilmer, L. Metz, F. Faghri, S. S. Schoenholz, M. Raghu, M. Wattenberg, and I. Goodfellow, "Adversarial spheres," *arXiv preprint arXiv:1801.02774*, 2018.

[191] M. Schlichtkrull, T. Kipf, P. Bloem, R. van den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *Proceedings of the European Semantic Web Conference*, 2018, pp. 593–607.

[192] J. Wang, W. Wang, P. A. Kollman, and D. A. Case, "Antechamber: an accessory software package for molecular mechanical calculations," *Journal of the American Chemical Society*, vol. 222, p. U403, 2001.

[193] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir *et al.*, "Wide & deep learning for rec-

ommender systems," in *Workshop on Deep Learning for Recommender Systems*, 2016, pp. 7–10.

[194] J. Ba, V. Mnih, and K. Kavukcuoglu, "Multiple object recognition with visual attention," *arXiv preprint arXiv:1412.7755*, 2014.

[195] B. Alexe, N. Heess, Y. Teh, and V. Ferrari, "Searching for objects driven by context," in *Proceedings of the 25th Advances in Neural Information Processing Systems*, vol. 25. Citeseer, 2012, pp. 881–889.

[196] H. Larochelle and G. E. Hinton, "Learning to combine foveal glimpses with a third-order boltzmann machine," in *Proceedings of the 23rd Advances in neural information processing systems*, vol. 23, 2010, pp. 1243–1251.

[197] M. Denil, L. Bazzani, H. Larochelle, and N. de Freitas, "Learning where to attend with deep architectures for image tracking," *Neural computation*, vol. 24, no. 8, pp. 2151–2184, 2012.

[198] R. Perko and A. Leonardis, "Context driven focus of attention for object detection," in *International Workshop on Attention in Cognitive Systems*, 2007, pp. 216–233.

[199] R. Ghaeini, X. Z. Fern, and P. Tadepalli, "Interpreting recurrent and attention-based neural models: a case study on natural language inference," *arXiv preprint arXiv:1808.03894*, 2018.

[200] U. Bhatt, A. Xiang, S. Sharma, A. Weller, A. Taly, Y. Jia, J. Ghosh, R. Puri, J. M. Moura, and P. Eckersley, "Explainable machine learning in deployment," in *Proceedings of the Conference on Fairness, Accountability, and Transparency*, 2020, pp. 648–657.

[201] E. Sood, S. Tannert, D. Frassinelli, A. Bulling, and N. T. Vu, "Interpreting attention models with human visual attention in machine reading comprehension," in *Proceedings of the ACL SIGNLL Conference on Computational Natural Language Learning*, 2020, pp. 12–25.

[202] G. Weir and A. Duta, "Strategies for neutralising sexually explicit language," in *Cybercrime and Trustworthy Computing Workshop*, 2012, pp. 66–74.

[203] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *Proceedings of the IEEE European symposium on security and privacy*, 2016, pp. 372–387.

[204] S. Gu and L. Rigazio, "Towards deep neural network architectures robust to adversarial examples," *arXiv preprint arXiv:1412.5068*, 2014.

[205] O. Lindenbaum, J. S. Stanley, G. Wolf, and S. Krishnaswamy, "Geometry based data generation," in *Proceedings of the 31st Advances in Neural Information Processing Systems*, 2018, pp. 1400–1411.

[206] H. Narayanan and S. Mitter, "Sample complexity of testing the manifold hypothesis," in *Proceedings of the 23rd Advances in neural information processing systems*, 2010, pp. 1786–1794.

[207] P. P. Brahma, D. Wu, and Y. She, "Why deep learning works: A manifold disentanglement perspective," in *Proceedings of the IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, 2016, pp. 1997–2008.

[208] ——, "Why deep learning works: A manifold disentanglement perspective," *IEEE transactions on neural networks and learning systems*, vol. 27, no. 10, pp. 1997–2008, 2015.

[209] T. K. Ho and M. Basu, "Complexity measures of supervised classification problems," in *Proceedings of the IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 3, 2002, pp. 289–300.
[210] L. Frank and E. Hubert, "Pretopological approach for supervised learning," in *Proceedings of 13th International Conference on Pattern Recognition*, 1996, pp. 256–260.
[211] A. C. Lorena, L. P. Garcia, J. Lehmann, M. C. Souto, and T. K. Ho, "How complex is your classification problem? A survey on measuring classification complexity," *ACM Computing Surveys*, vol. 52, no. 5, pp. 1–34, 2019.
[212] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.
[213] P. Flach and M. Kull, "Precision-Recall-Gain curves: PR analysis done right," in *Proceedings of the 13th International Conference on Neural Information Processing Systems*, 2015, pp. 838–846.
[214] A. Brindle, *The language of hate: A corpus lingusitic analysis of white supremacist language*. Routledge, 2016.
[215] L. Nouri and N. Lorenzo-Dus, "Investigating Reclaim Australia and Britain First's use of social media: Developing a new model of imagined political communities online," *Journal for Deradicalization*, vol. 18, pp. 1–37, 2019.
[216] N. Lorenzo-Dus and L. Nouri, "The discourse of the US alt-right online – A case study of the Traditionalist Worker Party blog," *Critical Discourse Studies*, vol. 18, no. 4, pp. 410–428, 2021.
[217] I. Saridakis and E. Mouka, "A corpus study of outgrouping in Greek radical right computer-mediated discourses," *Journal of Language Aggression and Conflict*, vol. 8, pp. 188–231, 2020.
[218] P. Baker, R. Vessey, and T. McEnery, *The language of violent jihad*. Cambridge University Press, 2021.
[219] V. Vapnik, *Estimation of dependences based on empirical data*. Springer Science & Business Media, 2006.
[220] V. N. Vapnik and A. Y. Chervonenkis, "On the uniform convergence of relative frequencies of events to their probabilities," in *Measures of complexity*. Springer, 2015, pp. 11–30.

Appendix A - CP2K Input

Example CP2K input file for a infinite FCC crystal composed of Al atoms.

```
&GLOBAL
  PROJECT Al-Crystal-01
  RUN_TYPE ENERGY
  !RUN_TYPE GEO_OPT
  PRINT_LEVEL LOW
&END GLOBAL

&FORCE_EVAL
  METHOD Quickstep
  &DFT
    BASIS_SET_FILE_NAME  BASIS_MOLOPT
    POTENTIAL_FILE_NAME  POTENTIAL

    &POISSON
      PERIODIC XYZ
    &END POISSON
    &QS
      EXTRAPOLATION USE_GUESS ! required for K-Point sampling
    &END QS
    &SCF
      !SCF_GUESS ATOMIC
      SCF_GUESS RESTART
      EPS_SCF 1.0E-10
      MAX_SCF 1000

      ADDED_MOS 4
      &SMEAR ON
        METHOD FERMI_DIRAC
        ELECTRONIC_TEMPERATURE [K] 300
      &END SMEAR
```

```
      &DIAGONALIZATION
        ALGORITHM STANDARD
        EPS_ADAPT 0.01
      &END DIAGONALIZATION
      &MIXING
        METHOD BROYDEN_MIXING
        ALPHA 0.2
        BETA 1.5
        NBROYDEN 8
      &END MIXING
    &END SCF
    &XC
      &XC_FUNCTIONAL PADE
      &END XC_FUNCTIONAL
    &END XC
    &KPOINTS
      SCHEME MONKHORST-PACK 2 2 2
      SYMMETRY OFF
      WAVEFUNCTIONS REAL
      FULL_GRID .TRUE.
      PARALLEL_GROUP_SIZE  0
!       KPOINT    0.0    0.0    0.0    1.0
!       KPOINT    1./2.  0.0    0.0    1.0
!       KPOINT    0.0    0.0    0.0    1.0
      &BAND_STRUCTURE
        ADDED_MOS 4
        FILE_NAME Al-bands.bs
        &KPOINT_SET
          UNITS CART_BOHR ! work around a bug in CP2K, should be B_VECTOR
          SPECIAL_POINT 0.0    0.0    0.0
          SPECIAL_POINT 1./2.  0.0    0.0
          SPECIAL_POINT 1./2.  1./2.    0.0
          SPECIAL_POINT 1./2.    1./2.    1./2.
          NPOINTS 10
        &END
      &END BAND_STRUCTURE
    &END KPOINTS
    &PRINT
                          # &EFIELD_CUBE
                          # &END EFIELD_CUBE
                          # &ELF_CUBE
                          # &END ELF_CUBE
                          # &E_DENSITY_CUBE
                          # &END E_DENSITY_CUBE
                          # &TOT_DENSITY_CUBE
                          # &END TOT_DENSITY_CUBE
                          # &V_HARTREE_CUBE
                          # &END V_HARTREE_CUBE
```

```
                        # &MO_CUBES
                        #    WRITE_CUBE T
                        #    NHOMO 1
                        #    NLUMO 1
                        # &END MO_CUBES
                    &END PRINT
  &END DFT

  &SUBSYS
    &CELL
      # Experimental values for a face-centered cubic Aluminum crystal
      ABC [angstrom] 4.04958*1.0 4.04958*1.0 4.04958*1.0
      ALPHA_BETA_GAMMA 90. 90. 90.
      SYMMETRY CUBIC
      PERIODIC XYZ
      MULTIPLE_UNIT_CELL 1 1 1
    &END CELL
    &COORD
      SCALED
      Al   0.0    0.0    0.0
      Al   1./2.   1./2.   0.0
      Al   1./2.   0.0   1./2.
      Al   0.0   1./2.   1./2.
    &END
          &KIND Al
                  BASIS_SET DZVP-MOLOPT-SR-GTH
                  POTENTIAL GTH-PADE-q3
    &END KIND
  &END SUBSYS
&END FORCE_EVAL
```

Appendix B - Hyper-parameters for energy predictions

The following hyperparameters were used for all experiments:

- Batch size: 20 for the Augmented QM9 and Infinite Crystal datasets, and 5 for the Crystal Growth dataset due to memory constraints.
- Learning rate: 1e-4
- Learning rate decay: 0.6
- Early stopping: 50 stable epochs
- Auxiliary alpha: 0.3
- Size of BT-specialised message $m_v|_{BT}$ (MPNN): 73
- Number of bond types: 4
- Number of message passing iterations (MPNN): 3

When training with the different implementations of the BT integration, though the size of the BT-specialised message $m_v|_{BT}$ remains consistent at 73 dimensions in MPNN, in case b) the concatenation of bond type messages entails that the nodes' hidden state size is 292. For clarity, we detail the size of MPNN's nodes' hidden state here for each of the methods:

- a: 73
- b: 292
- c: 73

Appendix C - Hyper-parameters for online grooming detection

- Optimiser: RMSprop (using the default learning rate)
- Scheduler: Cyclic Learning Rate (RMSprop base, 5e-3 max)
- Early stopping (tracking validation loss metric)
  - OGD-R: 50 epochs
  - OGD-T: 100 epochs
- Batch-size:
  - ODG-E: 128
  - OGD-T: 8 (with gradient accumulation over 16 batches)
- Gradient clipping: ± 0.5
- WSR Dimensionality:
  - OGD-R: 300
  - OGD-T: 768
- Pre-trained Glove embedding: 840B-300D crawl.
- Pre-trained XLNet: xlnet-base-cased (`https://github.com/zihangdai/xlnet/` & `https://github.com/huggingface/transformers`).
- Out-of-vocabulary (OOV) default embedding vector: random coordinates following a normal distribution of mean 0 and std 1 (i.e. close to the centre of the embedding manifold).
- LSTM hidden size (both base models): 256
- LSTM # layers (both base models): 2

- Classification layer (both base models): 1 fully connected layer
- Dropout rate between LSTM layers and classification layer: 20%
- Training/validation split: 70/30 (stratified)
- Training/testing split: 30/70 (stratified)
- Maximum sequence length: 2,000 - sorted & bucketed batches
- λ (weights of the additional losses): 1, 1, 1/3 for Stimulation of LSTM, Stimulation of attention, and Aux. OG process estimations, respectively
- Random weight initialisation seed: 42

Tokenisation details

As a standard step in NLP, we tokenise named entities prior to OG classification. The criteria for tokenisation and word replacement are as follows:

- All Spacey entities (see `https://spacy.io/api/annotation#named-entities`) are encoded to their respective categories, in addition to *LONGWORD* for words with more than 35 characters, and *URL* for URLs.
- Stemming using SnowballStemmer (NLTK).
- Tokenised using Spacey 'en' (English) model (`https://spacy.io/models`).
- Tokens with less than 5 occurrences in the corpus are replaced by OOV.

Appendix D - Example data from the Quantum Chemistry case study

For each of the three datasets used in the Quantum Chemistry case study, the data is stored in the same format. Therefore, as an example, we use an example from Augmented-QM9 to demonstrate the storage of molecular and crystalline data.


```
{"category":"dsgdb9nsd_050902.xyz",
 "cell_lengths":[25.5229999320,19.3680963191,18.9482609838,90.0,90.0,90.0],
 "center_mass":[12.443108345300001,9.157760554199999,9.4207166133],
 "chemical_formula":"C5H6N2O2",
 "chemical_symbols":["O","C","N","C","O","N","C","C","C","H","H","H","H","H","H"],
 "distances":[[0.0,1.7971037271, 3.4781958382, 4.4556783376, ...],
              [1.7971037271, 0.0, 2.0850125271, 3.7638974253, ...],
     [4.4556783376, 3.7638974253, 2.1192687483, ...]
     ...],
 "energy":"output\/calc-dsgdb9nsd_050902.xyz-9.inp",
 "index":99979,
 "input_a":1.5,
 "input_b":1.5,
 "input_c":1.5,
 "pdf":0.136503849499564,
 "positions":[[7.9179381322,7.499313307,8.2436204744],
              [8.2030742278,9.0144709281,9.1669712181],
      [10.0189409416,9.7012658599,9.927377713],
      ...],
 "simulated_energy":-82.1097602424,
 "smiles":"O=CNC(=O)NCC#C",
 "volume":9366.730252523799209,
 "uid":199999}
```


For details on the information stored in this data, please refer to Table 4.4.

Appendix E - Example data from OG dataset

Here we show some examples of the data (and the format in which it is stored) for the OG dataset.


```
{"id":"0002de15312dc33d78b6e9e4b5f61f1f",
 "input":"hi is there anybody there hy asl",
 "target":false,
 "tokenized":["hi","is","there","anybodi","there","hy","asl"],
 "themes":[[0.0,0.0,0.0,0.0,0.0,0.0,0.0],
           [0.0,0.0,0.0,0.0,0.0,0.0,0.0],
           [0.0,0.0,0.0,0.0,0.0,0.0,0.0],
           [0.0,0.0,0.0,0.0,0.0,0.0,0.0],
           [0.0,0.0,0.0,0.0,0.0,0.0,0.0],
           [0.0,0.0,0.0,0.0,0.0,0.0,0.0],
           [0.0,0.0,0.0,0.0,0.0,0.0,0.0]]}
```


This example contains the an `id` key as a unique identifier. It also contains the original `input`, as well as the `tokenized` input by the Spacey tokenizer. In addition, `themes` indicates the presence of the seven grooming processes per word. As this is a negative example, there are no grooming processes present.

This data, though currently formatted into multiple lines for readability, is stored in a JSON-lines (.jsonl) format. This allows the streaming of conversations instead of holding the entire corpus in memory.

Appendix F - Application of Methods

These strategies presented in this thesis, though general in their design, have been tested with concrete implementations for our three case studies. It is possible to use varying implementation styles for each of the different case studies. In Table 10.1, we show the basic implementation detail to use prior knowledge.

Table 10.1: *High level implementation style for the different case studies presented in this thesis.*

Section	Case Study Description	Implementation Style
Feature Specialisation		
5.2.1	Aux. OG process	Auxiliary loss
5.2.2	Aux. Atom Properties	Auxiliary loss
5.2.3	Aux. Type II properties	Auxiliary loss
5.3.3	Supervised WSR modification	Auxiliary loss
5.3.4	Elastic pull	Algorithm external to the DNN
5.3.5	Manifold learning	Algorithm external to the DNN
Specialised Information Processing		
6.2	Specialised OG channels	Additional information channel + auxiliary loss
6.3	Specialised bond type channels	Additional information channel
Attention on Data		
7.2	Stimulating attention on OG processes	Algorithm internal to the DNN
Data Meaningful Transformations		
8.1	Meaningful transformations of Type II bursts	Algorithm external to the DNN
8.2	Adversarial training for Type II bursts	Algorithm external to the DNN