

Contents

Acknowledgements	i
Abstract	v
List of Acronyms	xxi
1 Introduction	1
1.1 The Importance of Domain-Knowledge	4
1.2 Difficulties in Inclusion of Domain-Knowledge into Deep Neural Networks	5
1.3 Contributions of this Dissertation	6
1.4 Organisation of the Dissertation	7
2 Literature Review	9
2.1 Focus of this Review	10
2.2 Transforming the Input Data	11
2.2.1 Propositionalisation	11
2.2.2 Binary and n -ary Relations	15
2.3 Transforming the Loss Function	17
2.3.1 Syntactic Loss	17
2.3.2 Semantic Loss	18
2.4 Transforming the Model	21
2.4.1 Constraints on Parameters	21
2.4.2 Specialised Structures	23
2.5 Summary of the Review	26
3 Inclusion of Domain-Knowledge using Propositionalisation	29
3.1 Some Logic Programming Concepts	31
3.2 Relational Data and Relational Features	32
3.3 Propositionalisation	34
3.4 A Discrete Space of Relational Features	35
3.4.1 Bounding the Lattice of Relational Features	36
3.5 Utility-based Sampling of Relational Features	38
3.5.1 A Distributional Model of Discrete Search	40

3.6	Application to Deep Relational Machines (DRMs)	51
3.7	Empirical Evaluation	53
3.7.1	Aims	53
3.7.2	Materials	54
3.7.3	Method	58
3.7.4	Results	60
3.7.5	Limitations of DRMs	64
3.8	Summary	66
4	Simplified Inclusion of Relational Information using Vertex-Enrichment	69
4.1	Graph Neural Networks (GNNs)	70
4.1.1	General working principle of GNNs	72
4.1.2	Note on GNN variants	73
4.2	Inclusion of n -ary relations into GNNs by Enriching Vertex-Labels	74
4.2.1	Vertex-Enriched GNNs	78
4.3	Empirical Evaluation	80
4.3.1	Aims	80
4.3.2	Materials	80
4.3.3	Method	81
4.3.4	Results	83
4.3.5	Limitations of VEGNNs	85
4.4	Summary	87
5	Complete Inclusion of Relational Information using Inverse Entailment	89
5.1	Mode-Directed Inverse Entailment	90
5.1.1	Modes	93
5.1.2	Depth-Limited Bottom Clauses	95
5.2	BotGNNs	98
5.2.1	Notations and Assumptions	99
5.2.2	Construction of Bottom-Graphs	100
5.2.3	Some Properties of Clause-Graphs	104
5.2.4	Transformations for Graph Classification by a GNN	108
5.2.5	Note on Differences to Vertex-Enrichment	114
5.3	Empirical Evaluation	117
5.3.1	Aims	117
5.3.2	Materials	117
5.3.3	Method	118
5.3.4	Results	120
5.4	Summary	125

6	BotGNN as a System Component: An Application to Drug Design	127
6.1	The Problem	127
6.2	System Design and Implementation	130
6.2.1	Generating Acceptable Molecules	131
6.2.2	Obtaining Labels for Acceptable Molecules	132
6.2.3	Generating Active Molecules	132
6.3	System Testing	133
6.3.1	Materials	133
6.3.2	Method	134
6.3.3	Results	137
6.4	Summary	139
7	Conclusions and Future Work	141
7.1	Summary of the Dissertation	141
7.1.1	The Main Contributions	141
7.1.2	The Main Findings	142
7.2	Challenges and Future Work	143
7.3	Closing Remarks	144
A	Background	145
A.1	Deep Neural Networks	145
A.2	Inductive Logic Programming (ILP)	153
B	Additional Experimental Details	161
B.1	Details relevant to Chapter 3	161
B.2	Details relevant to Chapter 5	162
B.3	Details relevant to Chapter 6	162
	Bibliography	165
	List of Publications	187
	Biography of the Candidate	191
	Biography of the Supervisor	193
	Biography of the Co-supervisor	195

List of Figures

1.1	An example of using present day machine learning systems as assistance for scientific discoveries. The scientist-in-the-loop is a biologist. The biologist conducts experiments in a biological system, obtains experimental observations. The biologist then extracts data that can be used to construct machine learning model(s). Additionally, the machine learning system has access to domain-knowledge that can be obtained from the biologist. The machine learning system then conveys its predictions and explanations to the biologist.	2
2.1	Informal descriptions of (a) logical; and (b) numerical constraints.	10
2.2	Construction of a deep neural network model M from data (D) using a learner (\mathcal{L}). We use π to denote the structure (organisation of various layers, their interconnections, etc.) and θ to denote the parameters (synaptic weights) of the deep neural network. L denotes the loss function (for example, cross-entropy loss in case of classification).	10
2.3	Some implications of using domain-knowledge for commonly-used deep neural network architectures. Here MLP stands for Multilayer Perceptron, CNN stands for Convolutional Neural Network, RNN stands for Recurrenural Network and GNN stands for Graph Neural Network. MLPs, CNNs and RNNs are now commonplace architectures for deep neural networks and detailed descriptions can be found in any standard textbook (for example, [BGC17, ZLLS21]). GNNs are increasingly the DNN model of choice for dealing with graph-based data, and a good description can be found in [Ham20]. In this dissertation, we will be mainly concerned with MLPs and GNNs: the details required are in Appendix A	12
2.4	Introducing background knowledge into deep neural network by transforming data. \mathcal{T} is a transformation block that takes input data D , background knowledge (BK) and outputs transformed data D' that is then used to construct a deep model using a learner \mathcal{L}	13

2.5	Introducing background knowledge into deep neural network by transforming the loss function L . \mathcal{T} block takes an input loss L and outputs a new loss function L' by transforming (augmenting or modifying) L based on background knowledge (BK). The learner \mathcal{L} then constructs a deep model using the original data D and the new loss function L'	17
2.6	Introducing background knowledge into deep neural network by transforming the model (structure and parameter). In (a), the transformation block \mathcal{T} takes a input structure of a model π and outputs a transformed structure π' based on background knowledge (BK). In (b), the transformation block \mathcal{T} takes a set of parameters θ of a model and outputs a transformed set of parameters π' based on background knowledge (BK).	21
2.7	Some selected works, in no particular order, showing the principal approach of domain-knowledge inclusion into deep neural networks. DNN* refers to a DNN structure dependent on intended task. We use ‘MLP’ here to represent any neural network, that conforms to a layered-structure that may or may not be fully-connected. RNN also refers to sequence models constructed using Long Short-Term Memory (LSTM) or Gated Recurrent Unit (GRU) cells.	27
3.1	Michalski’s “trains” problem; adapted from [Mic80, MMPS94].	33
3.2	A fragment of the subsumption lattice of relational features for the trains problem.	36
3.3	The subsumption lattice of relational features for the trains problem. The space is bounded by $p(X) \leftarrow TRUE$ at the top and by the bottom-clause $(\perp_{B,M}(e))$ at the bottom. The size of the space is bounded by $\mathcal{O}(2^{ \perp_{B,M}(e) })$. The relational features are sampled from this space.	37
3.4	Redrawn and adapted from [JRS08]. Identifying the best subset of relational features for constructing a DRM. The X-axis enumerates the different subsets of relational features that can be constructed by an ILP engine (\mathcal{F} denotes the set of all possible relational features that can be constructed by the engine). The Y-axis shows the probability that a data instance drawn randomly using some pre-specified distribution will be correctly classified by the constructed DRM, given the corresponding feature-subset in X-axis. We wish to identify the subset that yields the highest probability, without actually constructing all the features in \mathcal{F}	39
3.5	The subsumption lattice of relational features for the trains problem. Each feature is associated with a utility score (shown in red colour). Our proposed utility-based sampling strategy selects features from this space. . .	40

3.6	Known Hider Distribution: (Left) Entropy of the hider distribution vs. Entropy of the seeker distribution, (Right) Entropy of the hider distribution vs. Expected number of misses by the seeker.	51
3.7	Unknown hider distribution, with more than 1 hider: (Left) $p = 0.1$, (Right) $p = 0.25$. That is, the proportion of boxes in the H 's partition of the step-approximation is known to be 10% and 25% of n . The number of balls is varied from 1% of n to 25% of n (X-axis). The expected number of misses is on the Y-axis.	52
3.8	Diagrammatic Representation of a Deep Relational Machine (DRM). The examples shown at the bottom are the predicates in data and background knowledge. The selection of relational features includes the feature construction and sampling steps.	52
3.9	Diagrammatic Representation of Constructing a DRM using relational features and propositionalisation. The inputs to an MLP represent a Boolean-valued feature vector obtained by propositionalisation of the relational features f_1, \dots, f_d . The parameters of the MLP are denoted as: $\mathbf{W}^{(\ell)}$, where ℓ denotes the layer index. In implementations, any $\mathbf{W}^{(\ell)}$ may contain an additional set of parameters called "bias weights" for which the inputs are always 1. The output of the MLP is an a class-label obtained from the class-probability vector of length k , where k is the number of classes. . .	53
3.10	A summary of the NCI-50 datasets (Total number of instances is approx. 220,000). Each instance in a dataset represents a chemical compound in atom-bond representation, along with its associated anti-cancer activity (positive or negative). Positive activity means the compound results in 50% growth inhibition of the tumor cells and negative activity means otherwise.	54
3.11	Levels of organisation of the background knowledge. Level 0 corresponds to the standard atom and bond information for the molecular compounds; Level 1 refers to the existence of various functional groups and ring structures; Level 2 knowledge is inferred further from Level 0 and 1.	55
3.12	Hierarchy of various functional groups in the background knowledge. . . .	56
3.13	Hierarchy of various ring structures in the background knowledge.	57

3.14	Improvements in predictive performance of DRMs, when provided with domain-knowledge through propositionalisation of relational features constructed using simple random sampling strategy by an ILP engine. The average number of relational features across the datasets is roughly 3800. Here X-axis represents the datasets (total 73 NCI datasets), and Y-axis shows the gain in predictive performance with respect to the baseline. Baselines (“1”) are the models without domain-knowledge. The corresponding quantitative comparison is shown in Figure 3.15	61
3.15	Comparison of predictive performance of DRM (Random Sampling) with and without domain-knowledge. The average number of relational features across the datasets is roughly 3800. The tabulations are the number of datasets on which DRM has higher, lower or equal predictive accuracy (obtained on a holdout set) than DRM without domain-knowledge. Statistical significance is computed by the Wilcoxon signed-rank test.	61
3.16	Qualitative comparison of predictive performance of DRMs (Hide-and-Seek: “HS” vs Random: “Rand”) with different number of relational features: {50, 100, 250, 500, 1000, 2500, 3800}; The number 3800 is to match the average number of features sampled using simple random sampling. Here X-axis represents the datasets (total 73 NCI datasets), and Y-axis shows the gain in predictive performance with respect to the baseline. Baseline here is the normalised performance of DRM-Rand: the “1” line. The corresponding quantitative comparison is shown in Figure 3.17	62
3.17	Comparison of predictive performance of DRM constructed with relational features sampled using hide-and-seek sampling strategy against DRM constructed using relational features sampled using simple random sampling. The last row contains 3800 features to match the average number of features sampled using simple random sampling. The tabulations are the number of datasets on which DRM(Hide-and-Seek) has higher, lower or equal predictive accuracy (obtained on a holdout set) than DRM(Rand). Statistical significance is computed by the Wilcoxon signed-rank test.	63
3.18	Comparison of predictive performance of DRM against LRNN [SAZ⁺18] and BCP+MLP [FZG14]. The DRM used here is the one constructed using 3800 relational features sampled using hide-and-seek sampling. The tabulations are the number of datasets on which DRM has higher, lower or equal predictive accuracy (obtained on a holdout set) than its counterparts. Statistical significance is computed by the Wilcoxon signed-rank test.	63
3.19	Degradation of DRM performance when expressivity of features is decreased from an unrestricted class to the class of relational features obtained using simple features as discussed in [MS98].	65

3.20	The minimum effort required to sample various number of relational features using the hide-and-seek sampling. The values tabulated are the number of relational features drawn from the large space features to obtain the number of features in the first column.	66
4.1	A diagrammatic representation of graph classification using a GNN. Graphs are of tuples of the form $(V, E, \sigma, \psi, \epsilon)$, where V is a set of vertices; E is a set of edges; σ is some neighbourhood function; ψ is a vertex-labelling; and ϵ is an edge-labelling. Often σ is left out, and derived from the edges in E	73
4.2	Components involved in implementing the workflow in section 4.1 for VEGNN models. The input is the vectorised representation of a vertex-enriched graph, denoted here as $VE\text{-}Graph(g)$ for an graph data-instance g . The blocks ‘Conv’ and ‘Pool’ refer to the graph-convolution and graph-pooling operations, respectively. The ‘Readout’ operation constructs a graph representation by accumulating information from all the vertex in the graph obtained after the pooling operation. The final graph representation is obtained in the READOUT block by an element-wise sum (shown as \oplus) of the individual graph-representations obtained after each AGGREGATE-COMBINE block. MLP stands for Multilayer Perceptron. .	82
4.3	Qualitative comparison of predictive performance of VEGNNs against Baseline (that is, GNN variants without access to domain-relations). Performance refers to estimates of predictive accuracy (obtained on a holdout set), and all performances are normalised against that of baseline performance (taken as 1). No significance should be attached to the line joining the data points: this is only for visual clarity.	84
4.4	Quantitative comparison of predictive performance of VEGNNs against GNNs. Here GNN refers to the graph-based neural network without domain-knowledge, and VEGNN refers to the network vertex-enriched with the generic domain-knowledge described in section 3.7.2 . The tabulations are the number of datasets on which VEGNN has higher, lower or equal predictive accuracy on a holdout-set. Statistical significance is assessed by the Wilcoxon signed-rank test.	85

4.5	Quantitative comparison of predictive performance of VEGNNs against DRMs. Here <i>VEGNN</i> denotes the vertex-enriched GNN with \mathcal{R} , and <i>DRM</i> denotes the Deep Relational Machine constructed using propositionalisation of relational features. The relational features for a DRM are sampled using the hide-and-seek sampling strategy proposed in Chapter 3. The set of the hide-and-seek features is denoted by \mathcal{R}' . The comparative performance of VEGNNs against DRMs starts worsening after $ \mathcal{R}' = 500$, which are not shown here. The tabulations are the number of datasets on which <i>VEGNN</i> has higher, lower or equal predictive accuracy on a holdout-set. Statistical significance is assessed by the Wilcoxon signed-rank test.	86
4.6	Quantitative comparison of predictive performance of <i>VEGNN</i> s against that of MLPs constructed using BCP features [FZG14]. The tabulations are the number of datasets on which <i>VEGNN</i> has higher, lower or equal predictive accuracy on a holdout-set. Statistical significance is assessed by the Wilcoxon signed-rank test.	86
4.7	Figure showing (a) a molecule with 2 fused benzene rings, (b) its corresponding molecular graph with vertices enriched with domain-relations. .	87
4.8	Figure highlighting a limitation of the vertex-enrichment technique for a molecular graph.	87
5.1	For the <i>gparent</i> example: (a) depth-limited bottom-clause $\perp_{B,M,2}(e)$; and (b) the corresponding clause-graph where the vertex-labels (λ, μ) s and (τ, γ) s are as provided in the preceding tables. The “dashed” square-box and the “dashed” arrow are shown to indicate the vertex specifying the head of the clause. The subscripts used in the labels correspond to the S.No. in the tables, for example, (λ_3, μ_3) refers to the third-row in the first table in this example; and, similarly, (τ_4, γ_4) refers to the fourth row in the second table.	99
5.2	Construction and use of bottom-graphs for use by GNNs in this chapter. We note that constituting the transformation of bottom-graphs are for the GNN implementations used in this chapter.	113
5.3	Dataset summary. Each bottom-graph can be represented using (G, \cdot) , where $G = (X, Y, E)$, where X represents the vertices corresponding to the relations, Y represents the vertices corresponding to ground terms in the bottom-clause constructed by MDIE, and E represents the edges between X and Y . The last 3 columns are the average number of X , Y and E in each bottom-graph in a dataset.	117

5.4	Components involved in implementing the workflow in section 4.1 for BotGNN models. ‘Conv’ and ‘Pool’ refer to the graph-convolution and graph-pooling operations, respectively. The ‘Readout’ operation constructs the representation of a graph by accumulating information from all the vertex in the graph obtained after the pooling operation. The final graph-representation is obtained in the READOUT block by an element-wise sum (shown as \oplus) of the individual graph representations obtained after each AGGREGATE-COMBINE block. MLP stands for Multilayer Perceptron.	119
5.5	Qualitative comparison of predictive performance of BotGNNs against Baseline (that is, GNN variants without access to domain-relations). Performance refers to estimates of predictive accuracy (obtained on a holdout set), and all performances are normalised against that of baseline performance (taken as 1). No significance should be attached to the line joining the data points: this is only for visual clarity.	121
5.6	Comparison of predictive performance of <i>BotGNNs</i> against <i>GNNs</i> . The tabulations are the number of datasets on which <i>BotGNN</i> has higher, lower or equal predictive accuracy (obtained on a holdout set) than <i>GNN</i> . Statistical significance is computed by the Wilcoxon signed-rank test.	122
5.7	Comparison of predictive performance of <i>BotGNNs</i> against <i>VEGNNs</i> . The tabulations are the number of datasets on which <i>BotGNN</i> has higher, lower or equal predictive accuracy (obtained on a holdout set) than a <i>VEGNN</i> . Statistical significance is computed by the Wilcoxon signed-rank test.	122
5.8	Quantitative comparison of predictive performance of <i>BotGNNs</i> against <i>DRMs</i> . <i>DRM</i> denotes the Deep Relational Machine constructed using propositionalisation of relational features. The relational features for a <i>DRM</i> are sampled using the hide-and-seek sampling strategy proposed in Chapter 3 . The comparative performance of BotGNNs against DRMs starts worsening after 1000 features, which are not shown here. The tabulations are the number of datasets on which <i>BotGNN</i> has higher, lower or equal predictive accuracy on a holdout-set. Statistical significance is assessed by the Wilcoxon signed-rank test.	123
5.9	Comparison of predictive performance of BotGNNs with an MLP constructed using BCP-based relational features. The tabulations are the number of datasets on which a <i>BotGNN</i> has higher, lower or equal predictive accuracy (obtained on a holdout set) than BCP+MLP.	124

5.10	Characterisation of vector-representation used for model-construction by BotGNNs, DRMs and BCP+MLP. Minimum/maximum values of the range are only shown to 3 meaningful digits (the actual values are not relevant here). The graph-representations (also, called graph-embeddings) for BotGNNs are constructed internally by the GNN. By “sparse” we mean that there are many 0-values, and by “very sparse”, we mean the values are mostly 0.	124
5.11	Comparison of predictive performance of BotGNNs with an ILP learner (Aleph system): (a) Without hyperparameter tuning in Aleph; (b) With hyperparameter tuning. In (a), the tabulations are the number of datasets on which <i>BotGNN</i> has higher, lower or equal predictive accuracy (obtained on a holdout set) than the ILP learner. In (b), each entry is the average of the accuracy obtained across 10-fold validation splits (as in [SKB03])	125
6.1	Early-stage drug-design (adapted from [WBS+15]).	128
6.2	An ideal conditional generator for instances of a random-variable denoting data (X) given a value for a random-variable denoting labels (Y) and domain-knowledge (B). Here, $Z \sim D$ denotes a random variable Z is distributed according to the distribution D . If the distributions shown are known, then a value for X is obtainable through the use of Bayes rule, either exactly or through some form approximate inference.	128
6.3	Training a conditional generator for generating “active” molecules. For the present, we assume the generator (G1) and discriminator (D) have already been trained (the G1 and D modules generate acceptable molecules and their labels respectively: the \hat{D} ’s are approximations to the corresponding true distribution). The Transducer converts the output of G1 into a form suitable for the discriminator. Actual implementations used in the chapter will be described below.	130
6.4	Training a generator for acceptable molecules. Training data consists of molecules, represented as SMILES strings, drawn from a database Δ . The VAE is a model constructed using the training data and generates molecules represented by SMILES strings. B_G denotes domain-knowledge consisting of constraints on acceptable molecules. The filter acts as a rejection-sampler: only molecules consistent with B_G pass through. . . .	131
6.5	Architecture of the VAE in Figure 6.4. m_1, m_2, n, k denote the number of blocks. The decoder along with the μ and σ constitute the generator that generates molecules in SMILES representation.	132

6.6	Discriminator based on BotGNN. “Logical” molecules refers to a logic-based representation of molecules. Bottom-graphs are a graph-based representation of most-specific (“bottom”) clauses constructed for the molecules by an ILP implementation based on mode-directed inverse entailment. . .	132
6.7	Summary of system performance. $B_D = B_1$ denotes that the discriminator has access to both propositional and relational domain-knowledge; $B_D = B_0$ denotes that the discriminator has access to propositional domain-knowledge only. <i>Random</i> denotes a random draw of molecules from the unconditional molecule generator G1. M denotes the set of molecules drawn (from the conditional generator, or from the unconditional generator for <i>Random</i>). The results are compared against the performance of a methodology purely based on Deep Reinforcement Learning [KBRR21]. M' denotes the set of acceptable molecules generated in the sample of M molecules (acceptable molecules satisfy molecular constraints defined on molecular properties). <i>Act</i> denotes the proportion of M' that are predicted active (the proxy model predicts an $\text{pIC}_{50} \geq 6.0$); <i>Sim</i> denotes the proportion of M' that are similar to active target inhibitors (Tanimoto similarity to active JAK2 inhibitors > 0.75). The numbers in parentheses denote the standard deviation in the corresponding estimate.	138
6.8	A chemical assessment of possible new JAK2 inhibitors. The molecules are from the sample of molecules from the conditional generator, that are predicted to have high JAK2 activity, and are significantly dissimilar to known inhibitors. The assessment is done by a computational chemist [†] . The assessment uses structural features and functional groups identified for the JAK2 site in the literature [KBRR21, DS13, DYCFY14].	140
A.1	Construction of a DNN model from data (Based on Figure 2.2; reproduced here for readability and completeness).	146
A.2	Representing MLP with layers as boxes. No importance to be given to the width of the boxes. The <i>depth</i> of the MLP is L . \mathbf{h} denotes a vector of hidden layer activations (also called hidden representation) and $\hat{\mathbf{y}}$ denotes the outputs. Superscript (ℓ) represents the layer index. The arrows show propagation of information (activations) from one layer to another. $\mathbf{W}^{(\ell)}$ denotes the parameters (a matrix of synaptic weights) at layer ℓ	147
A.3	Michalski’s trains problem; adapted from [Mic80, MMPS94].	153
A.4	Bounded search space in Progol.	158
A.5	A fragment of the hypothesis space in Aleph for the grandparent example, bounded by the most general hypothesis (at the top) and the most specific hypothesis (at the bottom).	159

List of Acronyms

Adam	Adaptive Moment Estimation (an optimisation algorithm)
AI	Artificial Intelligence
Aleph	A Learning Engine for Proposing Hypotheses (an ILP system)
BotGNN	Bottom-Graph Neural Network
BK	Background Knowledge
CNN	Convolutional Neural Network
CONV	Convolution (used for a block or a layer)
DL	Deep Learning
DNN	Deep Neural Network
DRM	Deep Relational Machine
GNN	Graph Neural Network
ILP	Inductive Logic Programming
MDIE	Mode-Directed Inverse Entailment
ML	Machine Learning
MLP	Multilayer Perceptron
NCI	National Cancer Institute
NN	Neural Network
POOL	Pooling (used for a block or a layer)
RNN	Recurrent Neural Network
SGD	Stochastic Gradient Descent
SMILES	Simplified Molecular-Input Line-Entry System
VAE	Variational Autoencoder
VEGNN	Vertex-Enriched Graph Neural Network
XAI	Explainable Artificial Intelligence

