An Architecture For Relational Learning Through Iterative Search Over Hypothesis Space

Osama F Rama
Department of Computing
Imperial College London
`ofr15@imperial.ac.uk`

Alessandra Russo
Department of Computing
Imperial College London
`a.russo@imperial.ac.uk`

Krysia Broda
Department of Computing
Imperial College London
`k.broda@imperial.ac.uk`

Abstract

In this paper we present Neural Concept Network (NCN), an architecture for relational learning through neural guided search. The architecture is composed of input nodes, which represent concepts given as facts in a knowledge-base, output nodes, which represent positive and negative ground instances of a concept to learn, and an intermediate layer, called *implication layer*, that captures the space of possible solutions. As this space is in practice very big, the implication layer is dynamically constructed during the training process as a neural guided search that learns the most relevant solutions. We evaluate the approach over two classes of problems, inductive learning and knowledge-base completion, and show that NCN achieves similar or better performance than existing methods for these tasks.

1 Introduction

Relational learning is essential for various cognitive tasks including reasoning, planning, problem solving, and language comprehension. Although deep learning approaches can easily deal with high-dimensional input and extract meaningful representations through supervised and unsupervised learning tasks (LeCun et al., 2015; Schmidhuber, 2015; Goodfellow et al., 2016), when it comes to relational learning, existing approaches struggle in terms of learning general-purpose abstractions (Puebla et al., 2020). The need to learn general definitions of high-level semantic concepts has recently been highlighted as critical for generalisation, transfer, continuous learning, and (hierarchical) composition of concepts (Bengio, 2020). Symbolic AI can provide powerful algorithms for learning first-order rule-based concept definitions from labelled ground instances (Muggleton et al., 2012). But, despite their advantages, existing approaches struggle with scalability and fall short in terms of learning relational concepts from unstructured data such as images.

Recent work has focused on how to combine the complementary benefits of Artificial Neural Networks (ANN) and symbolic representation, see (Garcez et al., 2019) for a survey. Most of existing approaches tackle knowledge-base reasoning through the learning of embedded representations of symbolic concepts (Bordes et al., 2013; Socher et al., 2013; Trouillon et al., 2016; Rocktäschel and Riedel, 2017). Their main drawback is the inability to generalise to unseen objects, limiting the transferability of their learned models. Neural-LP (Yang et al., 2017) addresses this by learning an entity independent model which is transferable across domains. δ-ILP (Evans and Grefenstette, 2018), on the other hand, provides differentiable inductive learning over an hypothesis space that is represented symbolically. Although capable of learning rule-based models from raw data, such approach is not scalable and limited to learning specific classes of predefined rule templates.

This paper presents an architecture, called *Neural Concept Network* (NCN), for structured rule learning that exploits a differentiable mechanism as a neural guided search over a set of possible solutions. This allows the architecture to be scalable whilst achieving similar or better performance than existing approaches. NCN learns concepts, expressed using first-order relations. Inference in NCN involves computing the gating tensor which is based on the network structure and which semantically filters the input and maps it to the grounded output. The network structure is based on an intermediate layer, called *implication layer*, which is composed of nodes that express Boolean combinations of relational concepts. This layer is dynamically constructed during the training process. Learning in NCN involves therefore learning the structure as well as parameters of the network. We evaluate the approach over two classes of problems, inductive learning and knowledge-base completion. We compare its performance with neuro-symbolic systems, such as δ-ILP (Evans and Grefenstette, 2018), NTP (Campero et al., 2018), Neural-LP (Yang et al., 2017), as well as a symbolic rule-mining system called AMIE+ (Galárraga et al., 2015). Our results show that NCN achieves similar or better performance as compared to these existing methods.

2 Architecture

The architecture of NCN is characterised by two types of nodes, *Implication Unit* (IU) and *Concept unit* (CU). A concept unit identifies a predicate and arity. Input and output nodes of the architecture are concepts units and capture, respectively, given facts in a knowledge-base (input nodes) and positive and negative ground instances of concepts to learn (output or target nodes). The IUs form the intermediate layer of the architecture. An IU represents a conjunction of concepts from the preceding layer (e.g., input nodes), and can be seen as a structured filter over such concepts. Specifically, an IU only produces an output ground concept (i.e. through its link to the target CUs) when there is a variable binding of its conjunction of concepts, that is satisfied by the facts represented by the input nodes. As different IU nodes may produce the same output ground instance concept, the target output node can be seen as a disjunction of alternative definitions of the target concept that needs to be learned. A single layer instantiation of the NCN architecture is given in Figure 1, where the Input Layer is formed by two input nodes representing binary predicates $in1$ and $in2$ respectively, whose instances are input facts from a given knowledge base. Layer 1 includes the Implication Layer (IL) and the output Concept layer (CL). The former is composed of a set of IUs that imposes a structured filter over the input facts, and the latter includes a single output node representing a binary $target$ predicate. In general, the output CL can be composed of as many CUs as the different predicates (i.e. concepts) that need to be learned. Note also that a NCN architecture may include more than one layer. In this case the CL of the hidden layers includes latent CUs needed to learn the final target concept. In what follow we refer to a general NCN with L layers.

NCN architecture is based on sparse connections. The edges from the CU nodes of layer $l-1$ to the IU of layer l are represented by the sparse weight matrix $W^{(1),l}$, where $l \in \{1, ..., L\}$ and L is the total number of layers[1]. Similarly, the edges from the IUs to CUs in layer l are represented by the sparse weight matrix $W^{(2),l}$. Inference in NCN is based on the gating tensor G^l – a sparse binary tensor – that captures the facts from the layer $l-1$ and the IUs used in layer l needed to generate the CUs facts in layer l. The shape of the gating tensor is (m, j, k, q, r), where m is the total number of literals in the IL and is equal to $\Sigma_i length(IU_i^l)$, where i is the number of IUs in layer l, j is the number of CUs in layer $l-1$, k and q are the number of unique argument tuples of facts of CUs in layer $l-1$ and l respectively, and r is the maximum number of proofs for each output fact of the IL and is defined by a hyperparameter. Computing the gating tensor involves grounding the rules represented by the IUs (see Appendix A for a detailed example).

$$\Theta_{mqr}^l = \sum_j \sum_k \left(A_{jk}^{l-1} \odot G_{mjkqr}^l \right) \quad (1)$$

$$\Phi_{iq}^l = f^1 \left(\sum_r \left(W_{im}^{(1),l} \cdot \Theta_{mqr}^l \right) \right) \quad (2)$$

$$A_{pq}^l = f^2 \left(W_{pi}^{(2),l} \cdot \Phi_{iq}^l \right) \quad (3)$$

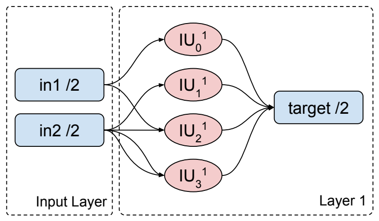


Figure 1: Example of NCN with one layer.

The forward propagation through a layer involves three tensor operations (Equations 1 - 3). Equation 1 filters and maps the input activation values onto the gating tensor and aggregates over the input facts.

[1]Although we present here a generic formulation for multiple layers, the evaluation in this paper is based on a single layer architecture as depicted in Figure 1, where $L = 1$.

Θ^l is the intermediate value of the IL of layer l. Φ^l is the output value of the IL. Intuitively, Equations 1 and 2 together are similar to a convolution operator except that, in this case, the filter (represented by the IU) is applied semantically to the input, i.e. the filter has a semantic prior as opposed to the structural prior based on locality in CNN filters (LeCun et al., 2015). The filter is dynamically imposed on the input, by means of the gating tensor G^l in Equation 1. Equation 3 aggregates the output of the IUs connected to the CUs, to obtain the final output A^l of shape (p, q), where p is the number of output CUs in layer l. Functions f^1 and f^2 are defined as $tanh$ and $max\left(0, tanh\right)$ respectively.

3 Learning

Learning in NCN involves learning the structure, characterised by the IUs of the intermediate layers, as well as the parameters of the network. The objective of the structure learning can be defined as obtaining the set of IUs, corresponding to rules that best approximate the definition of the target CU. The learning of the weights is performed using the learning objective function given in Equation 4. This is modelled as a Binary Cross-Entropy (BCE) problem where Y, the truth value of a target concept, is a binary label (1 and 0) for a positive and negative instance of the target concept, respectively. The regularisation loss $\mathcal{L}_{L1}$ is based on $L1$ norm of the parameters which characteristically induces sparsity. The loss $\mathcal{L}_{Len}$ is based on the scaled $L1$ norm of the CL parameters, where a IU to CU edge's norm is scaled by the number of conjunct concepts associated with the corresponding IU. This can be described as a term for structural regularisation, and represents a penalty term that helps prefer IUs with less concepts in conjunction over IUs with longer conjunction of concepts. This is in line with the Occam's razor principle. Appendix B.2 presents a brief ablation experiment over the loss terms with related results, in the case of the UMLS datset.

$$J\left(W\right) = \lambda_1 \cdot \mathcal{L}_{BCE}\left(A^L, Y\right) + \lambda_2 \cdot \mathcal{L}_{L1}\left(W\right) + \lambda_3 \cdot \mathcal{L}_{Len}\left(W^{(2)}\right) \tag{4}$$

The learning of the structure of an IL of the NCN architecture is performed through an iterative process (see algorithm in Appendix B.1), up to a given maximum number of IU length. Each iteration makes use of two functions *specialisation* and *pruning*. The former adds new IUs that have longer conjunction of concepts (i.e. seen as specialisations of existing (shorter) IUs), in order to reduce the inference of negative examples. The pruning function, instead, retains (from one iteration to the next), those IUs that are more likely to define the target concept. The pruning uses a magnitude-based approach, similar to (Han et al., 2015; Guo et al., 2016; Frankle and Carbin, 2019), to eliminate edges from the IL to the CL in a layer. The algorithm can, therefore, be seen as a beam search that iteratively explores the search space of possible rules (i.e. IUs) whilst containing its size through pruning at the end of each iteration. This allows scalable learning without using restrictive rule templates, as the algorithm only explores the part of the search space that may help in achieving best coverage of the target concept.

4 Experiments, Conclusion, and Future Work

We evaluate NCN on inductive learning problems. We consider the non-recursive tasks presented in (Evans and Grefenstette, 2018), as NCN does not yet support the learning of recursive rules. For comparison purposes we adopt the same evaluation metric used in (Evans and Grefenstette, 2018), i.e. percentage of runs that achieve less than 1e-4 mean squared test error. We evaluate NCN over 100 test runs for each task. Table 1 shows the results compared with δ-ILP (Evans and Grefenstette, 2018) and NTP (Campero et al., 2018). Further discussion is given in Appendix C.

Table 1: Results for inductive learning tasks including Undirected Edge (UE), Adjacent To Red (ATR), and Two Children (TC). The $-$ symbol means results not reported in the related paper.

	Task								
System	Predecessor	Son	Grandparent	Husband	Uncle	Father	UE	ATR	TC
δ-ILP	100	100	96.5	100	70	100	100	50.5	95
NTP	100	100	**100**	–	–	100	100	**100**	0
NCN	100	100	**100**	100	**100**	100	100	**100**	**100**

We have also considered knowledge-base Completion (KBC) tasks with large datasets to evaluate the scalability of NCN, where the objective is to learn a model for each relation in a knowledge-base that accurately predicts facts in the test set. Specifically, we have carried out experiments over the UMLS, Kinship, WN18, and FB15k-237 datasets, summarised in Table 2, using the same test set as in Neural-LP (Yang et al., 2017). The learning task is set up so that the training is carried out separately for each relational concept, in terms of other relations used as input concepts. The evaluation metrics used in this case are Mean Reciprocal Rank (MRR), average of the reciprocal rank of the test facts, and Hits@10, percentage of test facts ranked in top ten (Bordes et al., 2013). The ranking of each ground test fact is computed following the method described in (Bordes et al., 2013): two separate corruption sets, one for each fact's argument are generated, and the rank of a test fact is the average of its ranks w.r.t each set separately. Table 3 shows the results for the KBC tasks where UMLS (3) and Kinship (3) represent learning rules with up to three body literals, instead of two. Finally, we have evaluated NCN w.r.t. AMIE+ (Galárraga et al., 2015) using Inductive knowledge-base Completion (IKBC) metrics based on Sensitivity, Precision, and F1 score. To do so, the output of NCN was converted into Boolean values by applying a threshold on the activation values of the target CUs. Appendix D details the experiment setup and provides further discussion on the results.

Table 2: Main features of the knowledge-base completion datasets.

Dataset	Relations	Train (Positive)	Test	Total Facts	Entities
UMLS	46	5,896	633	6,529	135
Kinship	25	9,586	1,100	10,686	104
WN18	18	146,442	5,000	151,442	40,943
FB15k-237	237	289,650	20,466	310,116	14,541

Table 3: Results for knowledge-base completion tasks showing mean and standard deviation (shown in brackets) values over 5 test runs, except for FB15k-237 results which are based on single test runs.

Dataset	KBC Metric	Neural-LP	NCN	IKBC Metric	AMIE+	NCN
UMLS	MRR	**0.733** (0.008)	0.681 (0.020)	Sensitivity	**0.897**	0.805 (0.015)
	Hits@5	**0.877** (0.007)	0.870 (0.019)	Precision	0.022	**0.210** (0.015)
	Hits@10	0.921 (0.005)	**0.951** (0.004)	F1	0.043	**0.333** (0.018)
UMLS (3)	MRR	0.730 (0.009)	**0.860** (0.009)	Sensitivity	**0.946**	0.818 (0.012)
	Hits@5	0.886 (0.013)	**0.956** (0.007)	Precision	0.017	**0.283** (0.011)
	Hits@10	0.930 (0.007)	**0.987** (0.003)	F1	0.034	**0.420** (0.011)
Kinship	MRR	0.612 (0.004)	**0.620** (0.006)	Sensitivity	**0.999**	0.896 (0.004)
	Hits@5	0.789 (0.003)	**0.861** (0.006)	Precision	0.020	**0.197** (0.005)
	Hits@10	0.906 (0.002)	**0.957** (0.003)	F1	0.039	**0.323** (0.007)
Kinship (3)	MRR	0.602 (0.008)	**0.651** (0.006)	Sensitivity	**1.000**	0.880 (0.006)
	Hits@5	0.783 (0.010)	**0.871** (0.007)	Precision	0.015	**0.243** (0.006)
	Hits@10	0.905 (0.004)	**0.961** (0.003)	F1	0.029	**0.381** (0.007)
WN18	MRR	0.944 (0.000)	**0.951** (0.029)	Sensitivity	0.747	**0.749** (0.000)
	Hits@5	0.946 (0.000)	**0.985** (0.008)	Precision	0.427	**0.789** (0.088)
	Hits@10	0.954 (0.000)	**0.992** (0.006)	F1	0.543	**0.766** (0.041)
FB15k-237	MRR	**0.251**	0.246	Sensitivity	0.335	**0.429**
	Hits@5	**0.315**	0.303	Precision	**0.029**	0.003
	Hits@10	0.373	**0.389**	F1	**0.053**	0.006

We have presented a novel architecture for structured relational learning through neural guided search. Experiments using a single layer NCN demonstrates that the approach is scalable to large datasets, whilst preserving comparable if not better performance. As part of our future work, we aim to extend the approach in two ways: firstly, by integrating NCN with other differentiable architectures (e.g., CNN) to support end-to-end relational learning from unstructured data (e.g., images); secondly, by extending structure learning to multiple layers with the objective of learning latent and interdependent concepts. We envisage to use a learnable controller that will navigate the search space to build hierarchical dependencies of latent concepts with localised changes.

References

Y. Bengio. Deep learning for system 2 processing, 2020. URL `http://www.iro.umontreal.ca/~bengioy/AAAI-9feb2020.pdf`.

A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems 26*, pages 2787–2795. Curran Associates, Inc., 2013.

A. Campero, A. Pareja, T. Klinger, J. Tenenbaum, and S. Riedel. Logical rule induction and theory learning using neural theorem proving. *CoRR*, abs/1809.02193, 2018.

R. Evans and E. Grefenstette. Learning explanatory rules from noisy data. *J. Artif. Int. Res.*, 61(1): 1–64, jan 2018. ISSN 1076-9757.

J. Frankle and M. Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019.

L. Galárraga, C. Teflioudi, K. Hose, and F. M. Suchanek. Fast rule mining in ontological knowledge bases with amie+. *The VLDB Journal*, 24(6):707–730, 2015.

A. Garcez, M. Gori, L. C. Lamb, L. Serafini, M. Spranger, and S. N. Tran. Neural-symbolic computing: An effective methodology for principled integration of machine learning and reasoning. *Journal of Applied Logics*, 6(4):611–632, 2019.

I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

Y. Guo, A. Yao, and Y. Chen. Dynamic network surgery for efficient dnns. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pages 1387–1395, USA, 2016. Curran Associates Inc. ISBN 978-1-5108-3881-9.

S. Han, J. Pool, J. Tran, and W. J. Dally. Learning both weights and connections for efficient neural networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, pages 1135–1143, Cambridge, MA, USA, 2015. MIT Press.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL `http://arxiv.org/abs/1412.6980`.

Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

S. Muggleton, L. De Raedt, D. Poole, I. Bratko, P. Flach, K. Inoue, and A. Srinivasan. Ilp turns 20. *Machine Learning*, 86(1):3–23, Jan 2012. doi: 10.1007/s10994-011-5259-2.

G. Puebla, A. E. Martin, and L. A. Doumas. The relational processing limits of classic and contemporary neural network models of language processing. *Language, Cognition and Neuroscience*, pages 1–15, 2020.

T. Rocktäschel and S. Riedel. End-to-end differentiable proving. In *Advances in Neural Information Processing Systems 30*, pages 3788–3800. Curran Associates, Inc., 2017.

J. Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

R. Socher, D. Chen, C. D. Manning, and A. Ng. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems 26*, pages 926–934. Curran Associates, Inc., 2013.

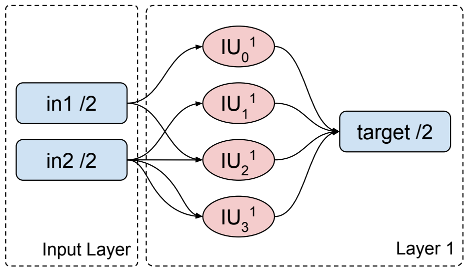
T. Trouillon, J. Welbl, S. Riedel, E. Gaussier, and G. Bouchard. Complex embeddings for simple link prediction. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 2071–2080. JMLR.org, 2016.

F. Yang, Z. Yang, and W. W. Cohen. Differentiable learning of logical rules for knowledge base reasoning. In *Advances in Neural Information Processing Systems 30*, pages 2319–2328. Curran Associates, Inc., 2017.

Appendices

A Architecture – Computing The Gating Tensor

Figure 2 shows an example instance of NCN with an input layer, containing two input CUs, an implication layer composed of four IUs, and an output concept layer containing a CU connected to the four IUs. Each IU captures a conjunction of input CUs as indicated in the table of Figure 2. The length of an IU is defined by the number of atoms it contains. The following is an example for computing the



IU	Description	IU Length
IU_0^1	$in1\,(V2, V1)$	1
IU_1^1	$in2\,(V2, V1)$	1
IU_2^1	$in1\,(V1, V3) \wedge in2\,(V3, V2)$	2
IU_3^1	$in2\,(V1, V3) \wedge in2\,(V3, V2)$	2

Figure 2: An example instance of NCN with IU description.

gating tensor G^1 of the network shown in Figure 2, given the input layer facts as shown in Table 4 and the IUs given in Figure 2. The first step involves grounding of the IUs. The input facts consist of seven unique argument tuples: $[(a, b), (c, b), (d, c), (d, f), (d, g), (e, c), (e, f)]$. The grounding is computed using the input layer facts and the IUs, and results in implication layer facts shown in Table 4, composed of nine unique argument tuples: $[(b, a), (b, c), (c, d), (c, e), (d, b), (e, b), (f, d), (f, d), (g, d)]$. For example, the fact $iu21(d, b)$ was computed using the facts $in1(d, c)$ and $in2(c, b)$ of the input layer and the IU_2^1. To capture this, two values of the gating tensor are set to 1 at the following indices: $G^1[2, 0, 2, 4, 0]$ and $G^1[3, 1, 1, 4, 0]$. To elaborate, the indices $[2, 0, 2, 4, 0]$ are obtained as follows:

- The first index represents the literal position in a concatenated list of all literals of IUs in layer 1: $[in1(V2, V1), in2(V2, V1), in1(V1, V3), in2(V3, V2), in2(V1, V3), in2(V3, V2)]$.
- The second index represents the corresponding CU in the input layer (which is $in1$ in this case).
- The third index represents the position of the tuple (d, c) in the aforementioned list of unique argument tuples in the input layer.
- The fourth index represents the position of (d, b) in the previously mentioned list of unique argument tuples in layer 1.
- The final index represents the proof in the total number of proofs for $lat2(d, b)$ from IU_2^1, which in this case is 0 because there's only a single proof.

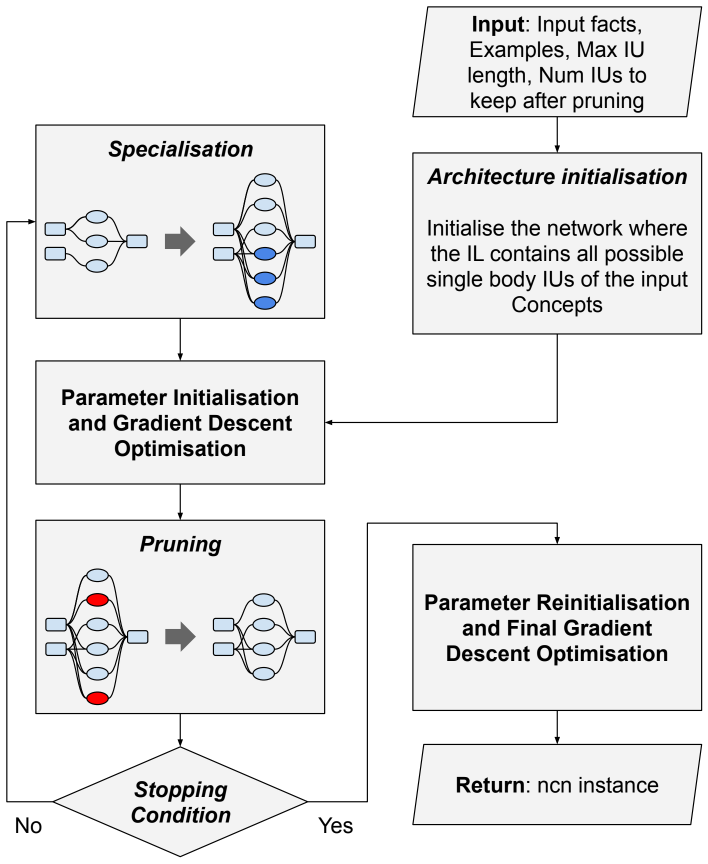
Table 4: Input and Implication layer facts.

Input Facts	Impplication Layer Facts	
$in1(a, b)$	$iu01(b, a)$	$iu21(d, b)$
$in1(d, c)$	$iu01(c, d)$	$iu31(e, b)$
$in1(d, f)$	$iu01(f, d)$	
$in1(d, g)$	$iu01(g, d)$	
$in2(c, b)$	$iu11(b, c)$	
$in2(e, f)$	$iu11(f, e)$	
$in2(e, c)$	$iu11(c, e)$	

B Neural Guided Structure Learning

B.1 Algorithm

Figure 3 shows the neural guided structure learning algorithm where the intermediate and final GD optimisation steps are based on 400 and 1000 epochs respectively for the evaluation. The input consists of input facts along with their activation values, target facts and their labels, and hyperparameters including: maximum IU length allowed, and the number of IUs to keep after pruning. Specialisation generates new IUs by adding an additional literal to the conjunction and considers all possible variable combinations with the restriction that at least one variable of the new literal must exist in the IU to be specialised. We use Adam optimisation (Kingma and Ba, 2015) for the intermediate and final gradient descent steps.



Input: Input facts, Examples, Max IU length, Num IUs to keep after pruning
Specialisation
Architecture initialisation
Initialise the network where the IL contains all possible single body IUs of the input Concepts
Parameter Initialisation and Gradient Descent Optimisation
Pruning
Parameter Reinitialisation and Final Gradient Descent Optimisation
Stopping Condition
No
Yes
Return: ncn instance

Figure 3: Neural guided structure learning algorithm for a single layer NCN.

B.2 Objective Function Ablation Experiments

The ablation experiments aim to investigate the effect of the regularisation loss terms $\mathcal{L}_{L1}$ and $\mathcal{L}_{Len}$. The results are based on four settings with different cost functions:

$$J1\left(W\right) = \lambda_1 \cdot \mathcal{L}_{BCE}\left(A^L, Y\right) \tag{5}$$

$$J2\left(W\right) = \lambda_1 \cdot \mathcal{L}_{BCE}\left(A^L, Y\right) + \lambda_2 \cdot \mathcal{L}_{L1}\left(W\right) \tag{6}$$

$$J3\left(W\right) = \lambda_1 \cdot \mathcal{L}_{BCE}\left(A^L, Y\right) + \lambda_3 \cdot \mathcal{L}_{Len}\left(W^{(2)}\right) \tag{7}$$

$$J4\left(W\right) = \lambda_1 \cdot \mathcal{L}_{BCE}\left(A^L, Y\right) + \lambda_2 \cdot \mathcal{L}_{L1}\left(W\right) + \lambda_3 \cdot \mathcal{L}_{Len}\left(W^{(2)}\right) \tag{8}$$

The experiments are based on the UMLS dataset under the same setting used for inductive knowledge-base completion tasks. The max IU length for the experiments is set to 3 – i.e. a maximum of three literals are allowed in the body of the target concept and hence the training involves 3 structure learning iterations. The number of IUs to keep after each pruning step is set to 10. The table 5 shows mean and standard deviation (of 5 test runs) of Sensitivity, Precision, and F1 scores on the test set for each setting. As compared to $J2$, the length loss in $J3$ results in a slight drop in precision but a significant increase in Sensitivity. In $J4$, the addition of $\mathcal{L}_{L1}$ loss improves the sensitivity and precision as compared to $J3$.

Table 5: Results for ablation experiments.

Cost Function	Sensitivity	Precision	F1
$J1$	0.311 (0.012)	0.192 (0.013)	0.237 (0.011)
$J2$	0.476 (0.014)	**0.199** (0.016)	**0.281** (0.018)
$J3$	0.661 (0.006)	0.169 (0.005)	0.269 (0.005)
$J4$	**0.677** (0.016)	0.171 (0.005)	0.273 (0.007)

C Inductive Learning Tasks

The NCN results, shown in table 1, are based on 100 test runs carried out for each task. The number of IUs kept after pruning was set to 32 and the maximum IU length was 3. The four tasks where NCN particularly outperforms are Grandparent, Uncle, Adjacent To Red, and Two Children. Convergence in δ-ILP (Evans and Grefenstette, 2018) highly depends on the weight initialisation and the system sometimes fails to converge to the optimal hypothesis. The following subsections detail each of the tasks along with the NCN model converted to rules, stating only the IUs with weight magnitude greater than 0.01 and omitting the literal weights ($W^{(1),1}$).

C.1 Predecessor

The task involves learning the $predecessor/2$ concept using the set of basic arithmetic facts of $zero/1$ and $succ/2$ relations. The number of positive and negative examples are 6 and 43 respectively. The top rule is as following:

```
target(V1,V2)::3.91 :- succ(V2,V1)
```

C.2 Son

The task involves learning the son/2 concept in terms of family relations $father/2$, $brother/2$, and $sister/2$. The number of positive and negative examples are 3 and 10 respectively. The top rule is as following:

```
son(V1,V2)::3.23 :- brother(V1,V3), father(V2,V3)
```

C.3 Grandparent

The task involves learning the $grandparent/2$ concept using facts of $mother/2$ and $father/2$ relations. The number of positive and negative examples are 7 and 74 respectively. The top 4 rules are as following:

```
grandparent(V1,V2)::3.05 :- father(V3,V2), mother(V1,V3)
grandparent(V1,V2)::3.05 :- father(V1,V3), father(V3,V2)
grandparent(V1,V2)::3.05 :- father(V1,V3), mother(V3,V2)
grandparent(V1,V2)::2.70 :- mother(V1,V3), mother(V3,V2)
```

C.4 Husband

The task involves learning the $husband/2$ concept using facts of $father/2$, $daughter/2$, and $brother/2$ relation. The number of positive and negative examples are 2 and 142 respectively. The top rule is as following:

```
target(V1,V2)::3.05 :- daughter(V3,V2), father(V1,V3)
```

C.5 Uncle

The task involves learning the $uncle/2$ concept using facts of family relations including $father/2$, $mother/2$, and $brother/2$. The number of positive and negative examples are 3 and 61 respectively. The top 2 rules are as following:

```
target(V1,V2)::3.05 :- brother(V1,V3), father(V3,V2)
target(V1,V2)::2.70 :- brother(V1,V3), mother(V3,V2)
```

C.6 Father

The task involves learning the $father/2$ concept using $husband/2$, $mother/2$, $brother/2$, and $aunt/2$ relations. The number of positive and negative examples are 2 and 223 respectively. The top rule is as following:

```
target(V1,V2)::3.05 :- husband(V1,V3), mother(V3,V2)
```

C.7 Undirected Edge

The task involves learning the $unconnected\_edge/2$ concept for a graph that is represented by facts of the $edge/2$ relation. The number of positive and negative examples are 5 and 11 respectively. The top 2 rules are as following:

```
target(V1,V2)::3.36 :- edge(V1,V2)
target(V1,V2)::3.36 :- edge(V2,V1)
```

C.8 Adjacent To Red

In this task, the nodes of a graph are coloured either green or red. The objective is to learn the concept $adjacent\_to\_a\_red\_node/1$. The background is composed of $edge/2$, $colour/2$, $green/1$, and $red/1$ relations. The number of positive and negative examples are 2 and 3 respectively. Note: the provided script for datasets by δ-ILP (Evans and Grefenstette, 2018) authors produces binary example facts for the target predicate, even though it mentioned that the target fact is unary. This is because representation in δ-ILP (Evans and Grefenstette, 2018) is strictly based on binary predicates. Since NCN supports unary predicates, we convert the binary examples to unary facts. The top rule is as following:

```
target(V1)::2.85 :- colour(V2,V3), edge(V1,V2), is_red(V3)
```

C.9 Two Children

The task involves learning the concept $has\_at\_least\_two\_children/1$. The background is defined in terms of $edge/2$ and not equals ($neq/2$) relations. The number of positive and negative examples are 2 and 3 respectively. Similar to the Adjacent To Red task, we convert the binary examples to unary facts. The top 2 rules are as following:

```
target(V1)::1.45 :- edge(V1,V2), edge(V1,V3), neq(V3,V2)
target(V1)::1.40 :- edge(V1,V2), edge(V1,V3), neq(V2,V3)
```

D knowledge-base Completion Tasks

The facts of the target concept present in the training set are used as positive examples for the KBC tasks. For each positive example, three corrupted facts are generated by randomly changing the first argument, the second argument, and lastly, both of the arguments. We make sure that none of the corrupted facts are present in either the training or the test set. The number of IUs kept after pruning was set to 16, except for FB15k-237 where it was set to 10. The maximum IU length was 2, except for UMLS (3) and Kinship (3), where the maximum IU length was 3.

Particularly of interest was the result for UMLS (3) and Kinship (3) which shows significant improvement as the maximum number of literals allowed in the body is increased, as opposed to Neural-LP (Yang et al., 2017) where the performance decreases when increasing the number of body literals. Table 6 shows the model learned for the *causes* concept in the UMLS dataset under two different settings, converted to rules, for the purpose of comparison. We also test NCN over FB15k-237 to evaluate the approach in terms of scalability. Due to time constraints only a single test was carried out for Neural-LP (Yang et al., 2017) and NCN.

Table 6: Example model learned for the causes concept under two different IU length settings, showing only the rules associated with the IUs of parameter magnitude greater than 0.01.

Max IU length = 3
causes(V1,V2)::2.05 :- result_of(V2,V3), uses(V3,V1)
causes(V1,V2)::1.01 :- analyzes(V3,V1), associated_with(V3,V2)
causes(V1,V2)::0.88 :- analyzes(V3,V1), diagnoses(V3,V2)
causes(V1,V2)::0.52 :- diagnoses(V3,V2), ingredient_of(V1,V4), uses(V3,V4)
causes(V1,V2)::0.42 :- ingredient_of(V1,V3), occurs_in(V2,V4), produces(V4,V3)
causes(V1,V2)::0.40 :- ingredient_of(V1,V3), occurs_in(V2,V4), uses(V4,V3)

Max IU length = 2
causes(V1,V2)::1.55 :- analyzes(V3,V1), associated_with(V3,V2)
causes(V1,V2)::1.40 :- analyzes(V3,V1), diagnoses(V3,V2)
causes(V1,V2)::1.18 :- associated_with(V3,V2), uses(V3,V1)
causes(V1,V2)::0.98 :- treats(V3,V2), uses(V3,V1)
causes(V1,V2)::0.22 :- associated_with(V3,V2), process_of(V3,V1)
causes(V1,V2)::0.19 :- isa(V3,V1), treats(V3,V2)

For IKBC tasks, we use the same NCN models learned for KBC tests. The evaluation is based on partial closed world assumption (Galárraga et al., 2015). It is assumed that the facts not present in the dataset are not true. This is also implicit in the aforementioned metrics (MRR and Hits@10). For each test fact, all possible corruptions are generated by corrupting the second argument of the fact. The confusion matrix is separately obtained for each concept in the test set, where true positives and false negatives are the test facts that the model outputs as positive and negative respectively. False positives are the corruptions of the test facts that the model outputs as positive.

For the purposes of evaluation, we make sure that the language bias is the same for both AMIE+ and NCN, and use the default settings for AMIE+ (Galárraga et al., 2015). The learned rules are non-recursive, safe, and can have a maximum of two body literals. Moreover, we only allow rules where each variable in a literal appears at least in one other literal. For the UMLS and Kinship datasets, while AMIE+ (Galárraga et al., 2015) achieves better sensitivity scores, it suffers significantly in terms of precision and F1. The higher true positive rate and lower precision is due to AMIE+ (Galárraga et al., 2015) learning more generic hypothesis. The number of rules learned by AMIE+ (Galárraga et al., 2015) was larger than NCN for most of the datasets. The table 7 compares the hypothesis size for each dataset.

Table 7: Hypothesis size comparision between NCN and AMIE+.

	Dataset					
System	UMLS	UMLS (3)	Kinship	Kinship (3)	WN18	FB15k-237
NCN	736	736	400	400	288	2,370
AMIE+	10,556	472,123	6,834	295,532	72	4,833

For the FB15k-237 dataset, NCN suffers significantly in terms of precision and F1. One possible explanation could be low number of epochs in each structure learning iteration. For initial evaluation, we used 100 and 400 epochs in the intermediate and final GD optimisation steps respectively. Looking at the result for each predicate in the dataset, it can be observed that just a small number of predicates are responsible for most of the false positive count. For example, 6 of the 237 relations together account for about 1.74 million false positives, more than half of the total of approximately 3 million. As part of the current work, in addition to increasing the number of epochs, incorporating additional negative examples in the training set will be explored.