

Imperial College
London

End-to-end Neuro-symbolic Rule Learning

Nuri Cingillioglu

Motivation

The human brain has developed to process symbolic information in a connectionist architecture.

Yet this level harmony between neural and symbolic research in machine learning remains a mystery.



Goal

- Learn dense image representations
- Learn to identify objects
- Continuous dense representations

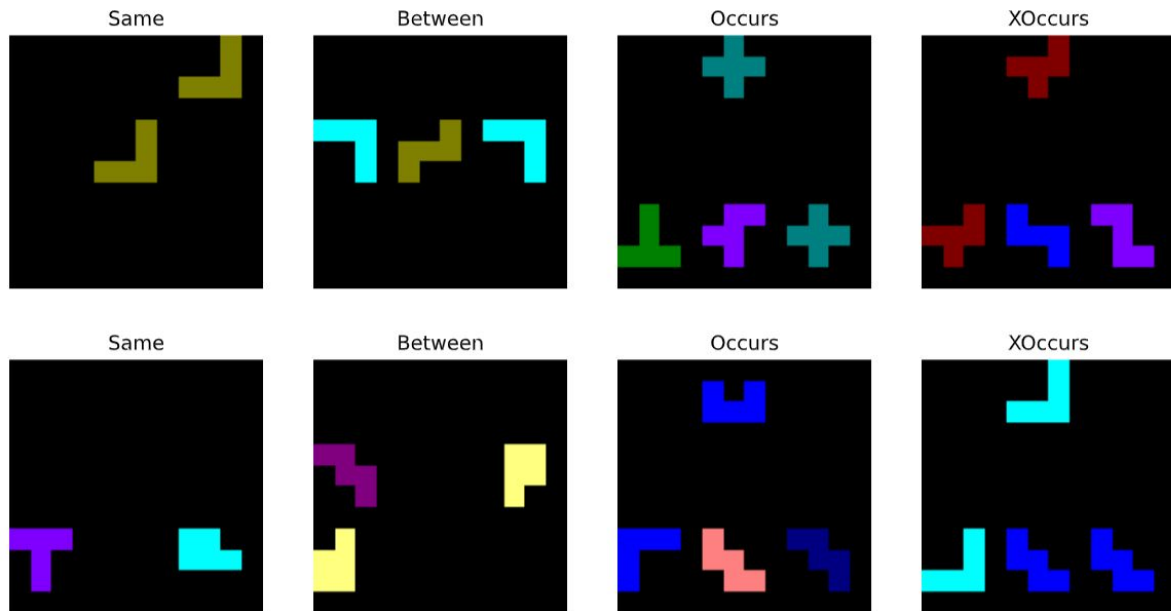
- Learn relations between objects
- Learn first-order rules to reason with
- Fuzzy logic representations

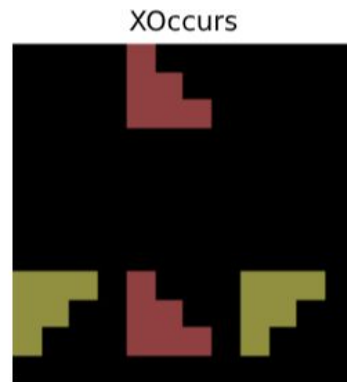
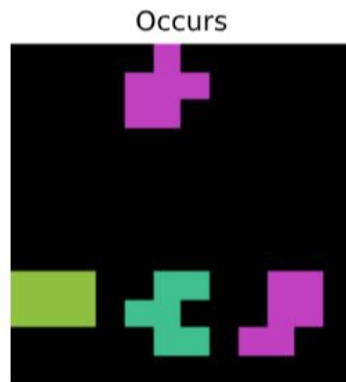
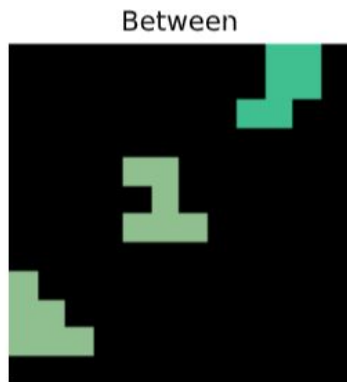
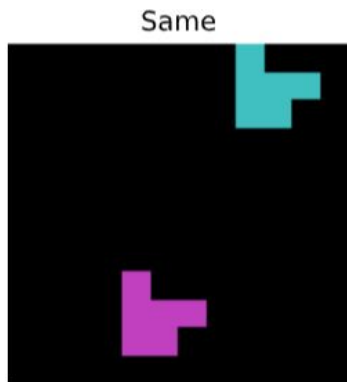
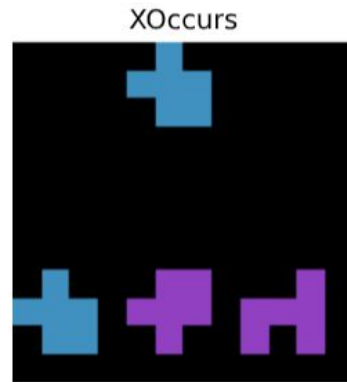
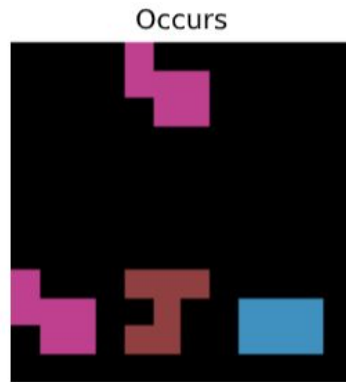
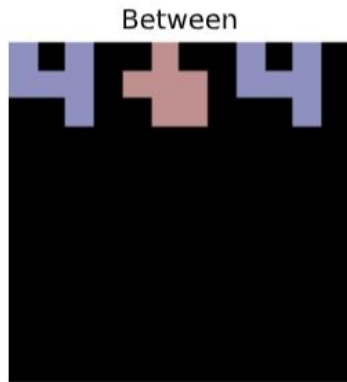
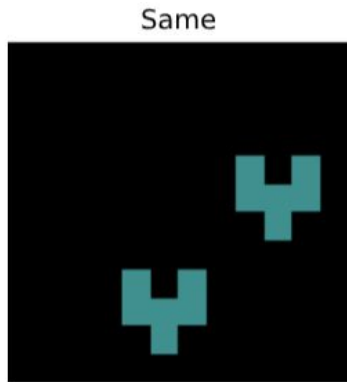
All in an end-to-end fashion, solely from examples

Relations Game

Binary image classification task with different shapes and colours that exhibit compound relations. 4 tasks in total with 3 sets, pentominoes, hexominoes and striped shapes.

Train on pentominoes, shown right, and them evaluate on unseen hexomomies and striped shapes.



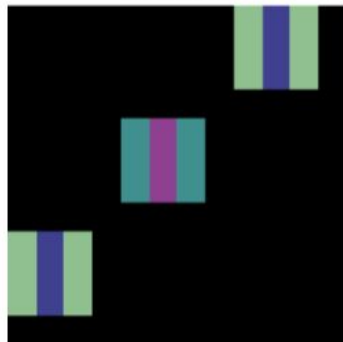


Example images from the hexominoes set

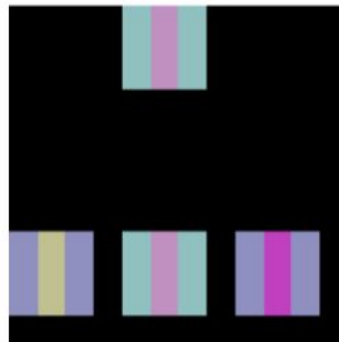
Same



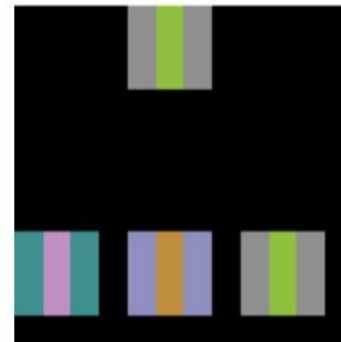
Between



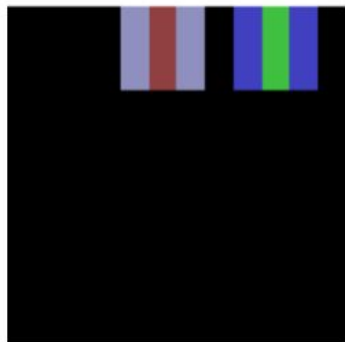
Occurs



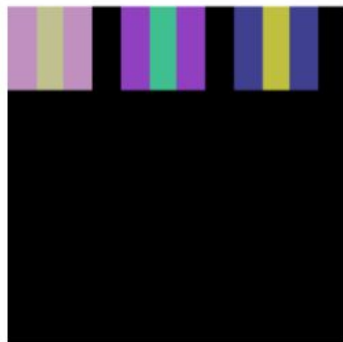
XOccurs



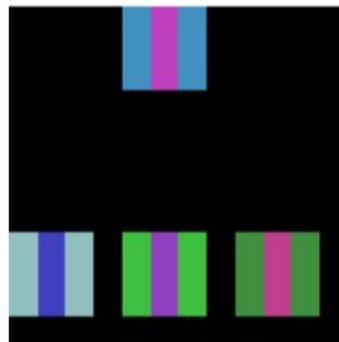
Same



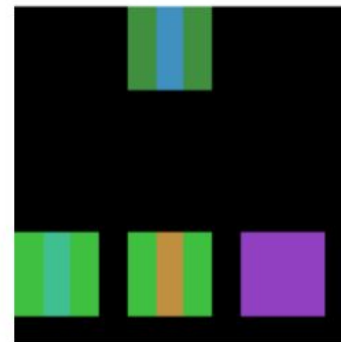
Between



Occurs



XOccurs



Example images from the stripes set

Between Rule

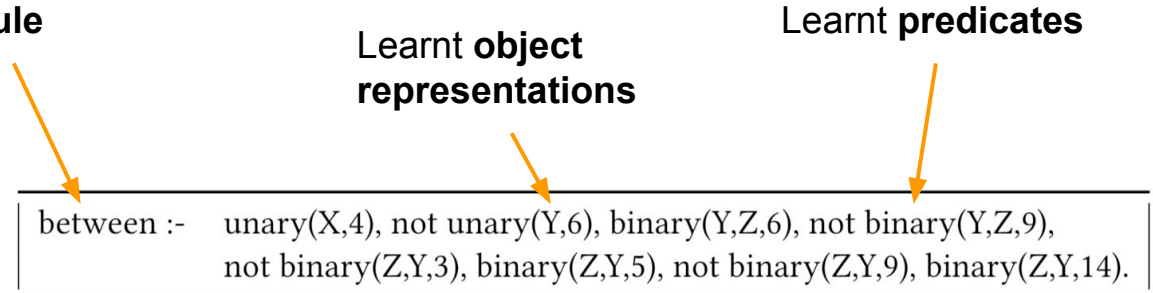
Trained on between task with 1k training examples, the following **symbolic** rule can be used to solve the task. The last argument is the predicate ID.

It achieves 97% accuracy for the hexominoes test set. Over sample test batch of 64 examples, clingo using the output of the neural networks achieves 90%. This is the first result that can learn dense object representations, relations and rules in one unifying architecture.

Learnt rule


Learnt object representations

Learnt predicates



```
between :- unary(X,4), not unary(Y,6), binary(Y,Z,6), not binary(Y,Z,9),
not binary(Z,Y,3), binary(Z,Y,5), not binary(Z,Y,9), binary(Z,Y,14).
```

Threshold relations and pass to **clingo**



```
# For an absolute sanity check, let's run the output of the neural network through clingo for this batch only
threshold_interpretation = {k: (v > 0.0)*2-1 for k, v in interpretation.items()}
result = utils.clingo.clingo_rule_check(threshold_interpretation, learnt_rules)
batch_acc = (result == report['out_label']).mean()
batch_acc
```

```
100%|██████████| 64/64 [00:01<00:00, 34.56it/s]
```

```
0.90625
```

All Tasks Rules

The following program is learnt when all the 4 tasks are mixed together. Here, we demonstrate predicate invention. Nullary(0..3) correspond to task ids, Same nullary(0), Between nullary(1) and so on. The obj(..) is added to make the rules ASP safe and the uniqueness of variables are added to let clingo prune unnecessary bindings.

The following program achieves >90% accuracy on all tasks combined.

Learnt / invented
hidden predicates

```
unary(V0,10) :- not c3unary(V0,10), obj(V0).
c3unary(V0,10) :- not nullary(3), unary(V0,3), binary(V0,V1,6), not binary(V1,V0,1),
                 binary(V1,V0,3), binary(V1,V0,10), binary(V1,V0,14),
                 obj(V1), V1 != V0, obj(V0).

unary(V0,11) :- not c1unary(V0,11), obj(V0).
c1unary(V0,11) :- not nullary(3), not binary(V0,V1,1), not binary(V0,V1,9),
                 not binary(V0,V1,11), binary(V1,V0,10), not binary(V1,V0,11),
                 binary(V1,V0,13), binary(V1,V0,14), obj(V1), V1 != V0, obj(V0).

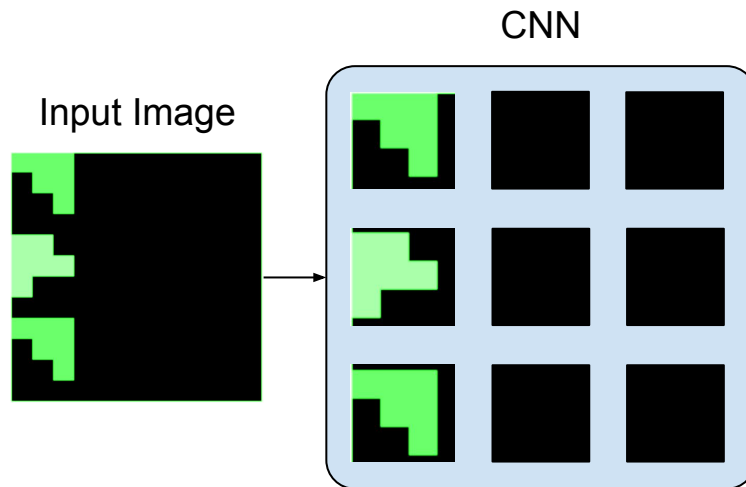
binary(V0,V1,23) :- not binary(V1,V0,13), obj(V1), V1 != V0, obj(V0).
binary(V0,V1,23) :- not c3binary(V0,V1,23), obj(V0), obj(V1), V0 != V1.
c3binary(V0,V1,23) :- not nullary(0), not binary(V0,V1,1), binary(V0,V1,3), binary(V0,V1,15),
                    not binary(V1,V0,1), binary(V1,V0,10), not binary(V1,V0,11),
                    binary(V1,V0,13), binary(V1,V0,14), obj(V1), V1 != V0, obj(V0).

t :- not c4t.
c4t :- unary(V2,10), unary(V2,11), obj(V2).
t :- binary(V0,V1,23), not binary(V3,V2,23), obj(V2), V2 != V1, V2 != V3,
      V2 != V0, obj(V1), V1 != V3, V1 != V0, obj(V3), V3 != V0, obj(V0).
```

Prediction
(target) label

Step 1: CNN

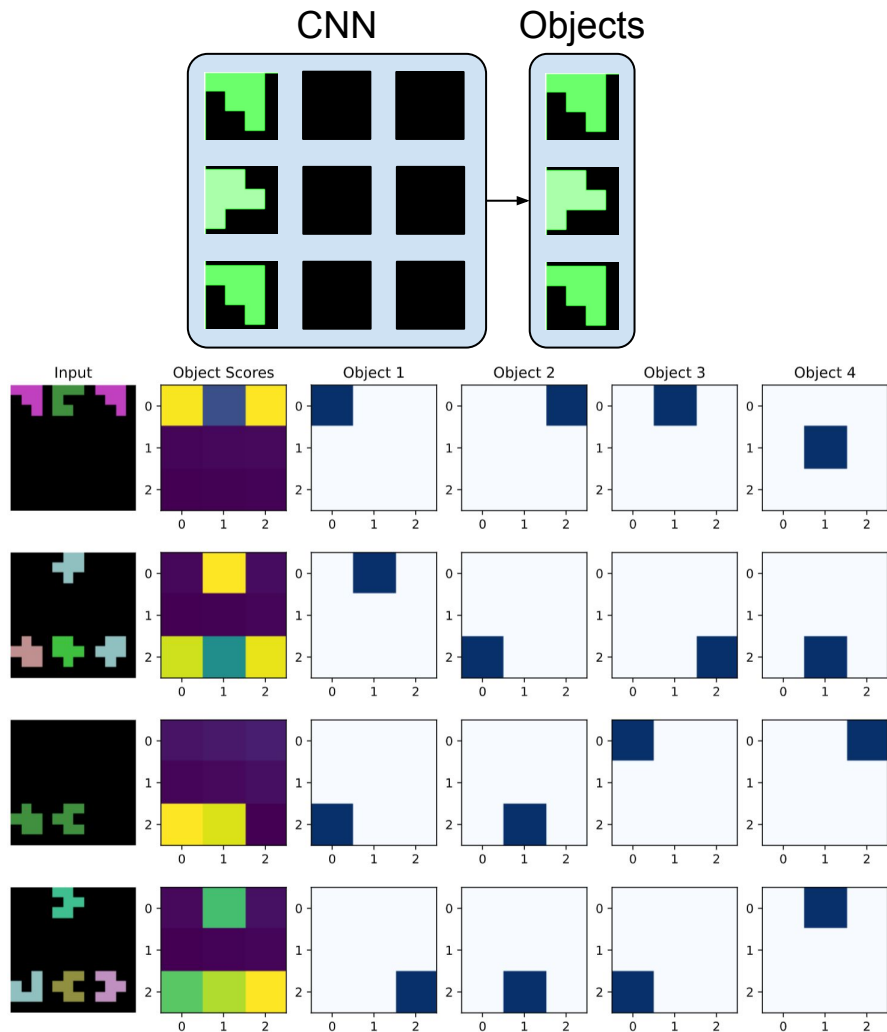
We process the image using a CNN that computes a dense representation with location information added.



0.	3.356	0.728	0.	0.871	1.170	1.726	1.528	1.174	0.117	1.586	0.478
2.527	0.286	0.724	1.723	0.580	0.	3.770	0.943	0.517	3.114	3.456	0.
0.710	0.975	2.030	1.570	1.251	1.971	1.830	2.688	0.5	0.5	0.5	0.5
2.624	0.304	0.244	0.	0.	2.959	0.589	3.234	2.745	2.195	0.	2.967
0.510	2.248	2.414	1.953	2.047	1.409	1.645	2.542	0.	2.825	1.280	0.
2.709	2.844	0.293	0.349	3.549	0.	0.	0.	1.	1.	0.	0.
2.624	0.304	0.244	0.	0.	2.959	0.589	3.234	2.745	2.195	0.	2.967
0.510	2.248	2.414	1.953	2.047	1.409	1.645	2.542	0.	2.825	1.280	0.
2.709	2.844	0.293	0.349	3.549	0.	0.	0.	0.	0.	1.	1.

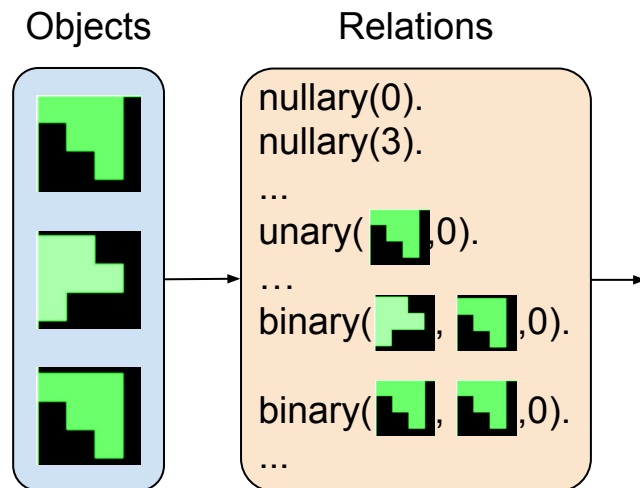
Step 2: Object Selection

Using an iterative method with Gumbel-Softmax, the model learns to recognise a subset of those as relevant objects. A single feed-forward layer computes the logits for the Gumbel-Softmax distribution, and we sample one object at a time. Since the sampling is random, the order of the objects can vary.



Step 3: Relations

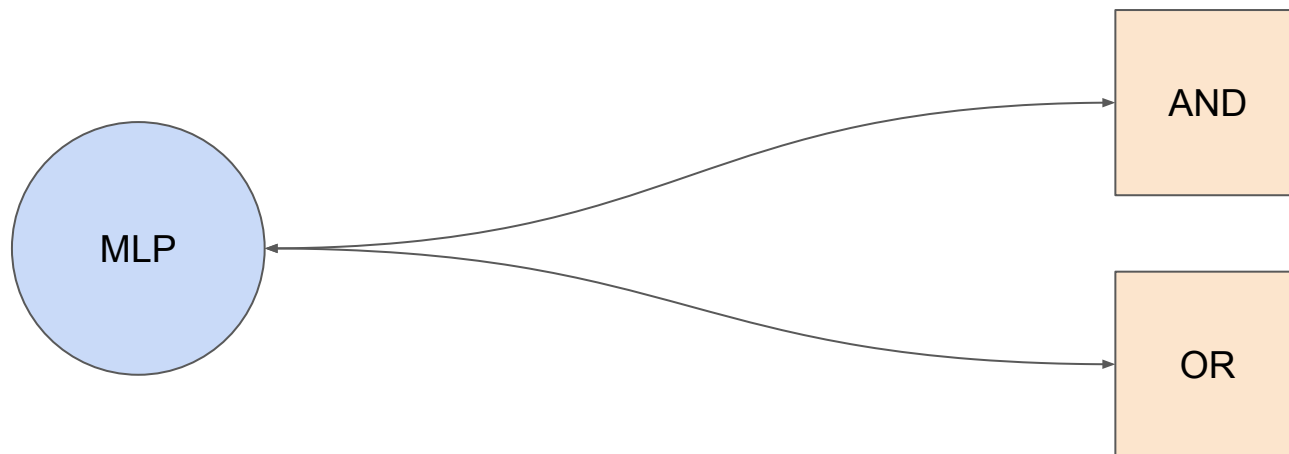
A single feed-forward layer with tanh activation learns unary and binary relations between objects. The nullary predicates are added in based on the task id and are not learnt.



$$\text{unary}(X, i) = \tanh(\mathbf{W}_i X + b)$$

$$\text{binary}(X, Y, j) = \tanh(\mathbf{W}_j [X, Y, X - Y] + b)$$

Step 4: Differentiable Rule Learning



End-to-end differentiable

Semi-symbolic Layer

If we utilise t-norms to implement fuzzy logic, as the number of inputs increases, the operation suffers from vanishing gradients and becomes unviable for downstream layers such as CNNs. This phenomenon occurs due to the 1 out of n failure or success characteristic of conjunction and disjunction respectively. Hence, we are interested in an operation that does not starve gradients and **eventually** converges to the desired semantics.

$$y = f\left(\sum_i w_i x_i + \beta\right) \quad (1)$$

$$\beta = \delta\left(\max_i |w_i| - \sum_i |w_i|\right) \quad (2)$$

Delta is the semantic gate selector ranging from 0 to 1 for conjunction and -1 for disjunction. Negation naturally is implemented as multiplicative inverse. We select f to be \tanh .

```
import tensorflow as tf
```

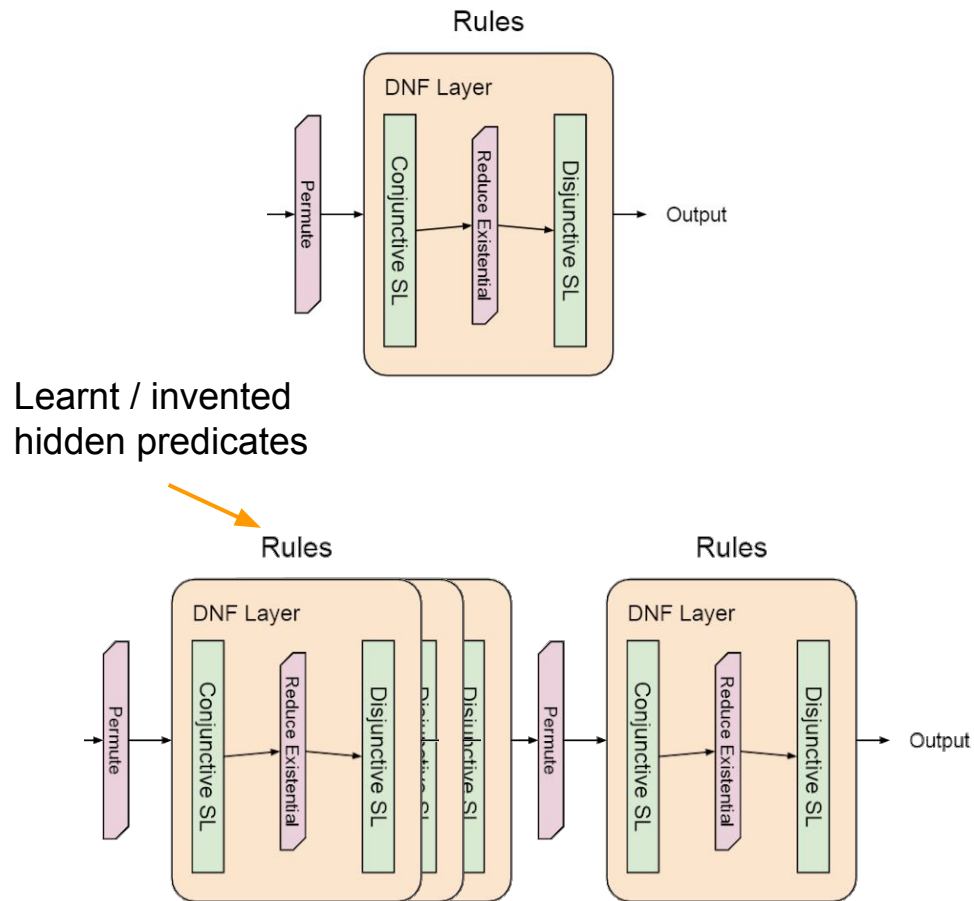
```
def semisymbolic_layer(in_tensor: tf.Tensor, kernel: tf.Tensor, delta: float):  
    """Compute semi-symbolic layer outputs of a given input tensor."""  
    # in_tensor (... , H), kernel (H, ), delta [1, -1]  
    abs_kernel = tf.math.abs(kernel) # (H, )  
    and_bias = tf.reduce_max(abs_kernel) - tf.reduce_sum(abs_kernel) # ()  
    conjuncts = tf.reduce_sum(in_tensor * kernel, -1) + delta*and_bias  
    return tf.nn.tanh(conjuncts)
```

Example implementation of SL in TensorFlow

Step 4: Differentiable Rule Learning

We construct a disjunctive normal form layer (DNF) by stacking two semi-symbolic layers, one conjunctive and one disjunctive. To learn first order rules, we curate all permutations of object to variable binding and use the *max* operator to reduce existential variables.

We construct 2 additional variants: **DNF-h** has an extra hidden DNF layer which invents 14 new predicates and **DNF-r** iterates the DNF layer twice learning recursive rules with 7 new predicates.



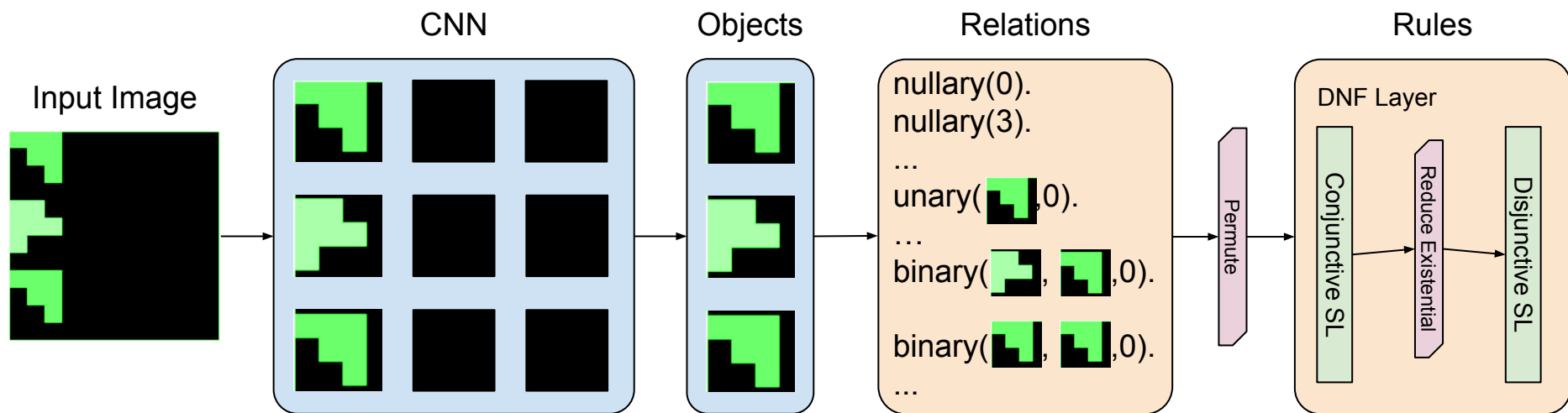
Step 5: Pruning and Thresholding

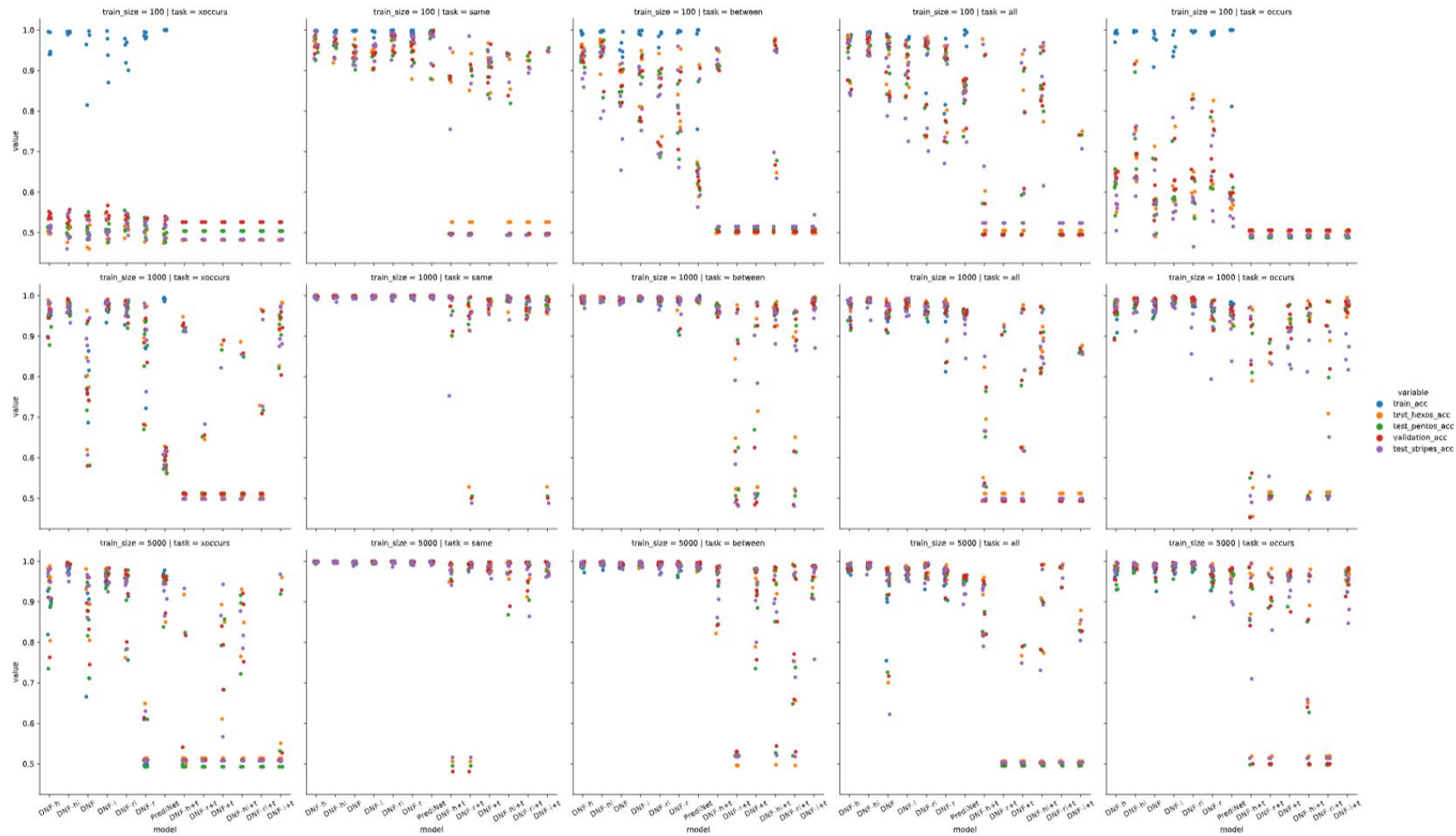
In order to obtain symbolic formulas, we prune then threshold the weights after training. Similar to decision tree pruning methods, each weight is set to zero and pruned if the performance has not dropped by a fixed epsilon. Then a threshold value is picked by sweeping over potential values $[\min|w_i|, \max|w_i|]$ with a similar performance check to pruning. Each weight is then set to $\text{sign}(w)t$. We use $t=6$ which gives sufficient saturation $\tanh(6) \approx 0.999$. Finally, we repeat the pruning step to remove any wrongly amplified weights.

Stage	Weights	Test Pent. Acc
Preprune	[1.56 -1.72 7.45 -2.22 -2.27 2.76 -1.46 1.75]	0.996
Pruned	[1.56 -1.72 7.45 -0.00 0.00 0.00 -1.46 1.75]	0.991
Threshold	[0.00 -0.00 6.00 0.00 0.00 0.00 -0.00 0.00]	0.942
Threshold + Pruned	[0.00 -0.00 6.00 0.00 0.00 0.00 -0.00 0.00]	0.938

between :- unary(X,4), not unary(Y,6), binary(Y,Z,6), not binary(Y,Z,9),
not binary(Z,Y,3), binary(Z,Y,5), not binary(Z,Y,9), binary(Z,Y,14).

End-to-end Neuro-symbolic Rule Learning





Is the DNF model more data efficient?

Set	Task Model	All			Between			Occurs			Same			XOccurs		
		100	1000	5000	100	1000	5000	100	1000	5000	100	1000	5000	100	1000	5000
Hex.	DNF	0.94	0.97	0.98	0.90	0.99	0.99	0.56	0.99	0.99	0.94	1.00	1.00	0.49	0.80	0.93
	→ DNF-h	0.98	0.99	0.99	0.95	1.00	0.99	0.62	0.99	0.99	0.97	1.00	1.00	0.50	0.98	0.96
	DNF-h+t	0.51	0.55	0.92	0.91	0.97	0.98	0.50	0.79	0.96	0.53	1.00	0.98	0.48	0.51	0.51
	DNF-hi	0.98	0.99	1.00	0.97	0.99	1.00	0.69	0.99	0.99	0.97	1.00	1.00	0.51	0.99	0.99
	DNF-r	0.94	0.98	0.98	0.84	1.00	0.99	0.63	0.99	0.99	0.96	1.00	1.00	0.51	0.94	0.51
	→ PrediNet	0.85	0.95	0.96	0.66	0.99	0.99	0.57	0.95	0.97	0.99	1.00	1.00	0.50	0.58	0.95

Comparing DNF models to Predinet, in particular DNF-h, we observe it performs better in median accuracy.

Can the DNF model generalise to unseen shapes and colours?

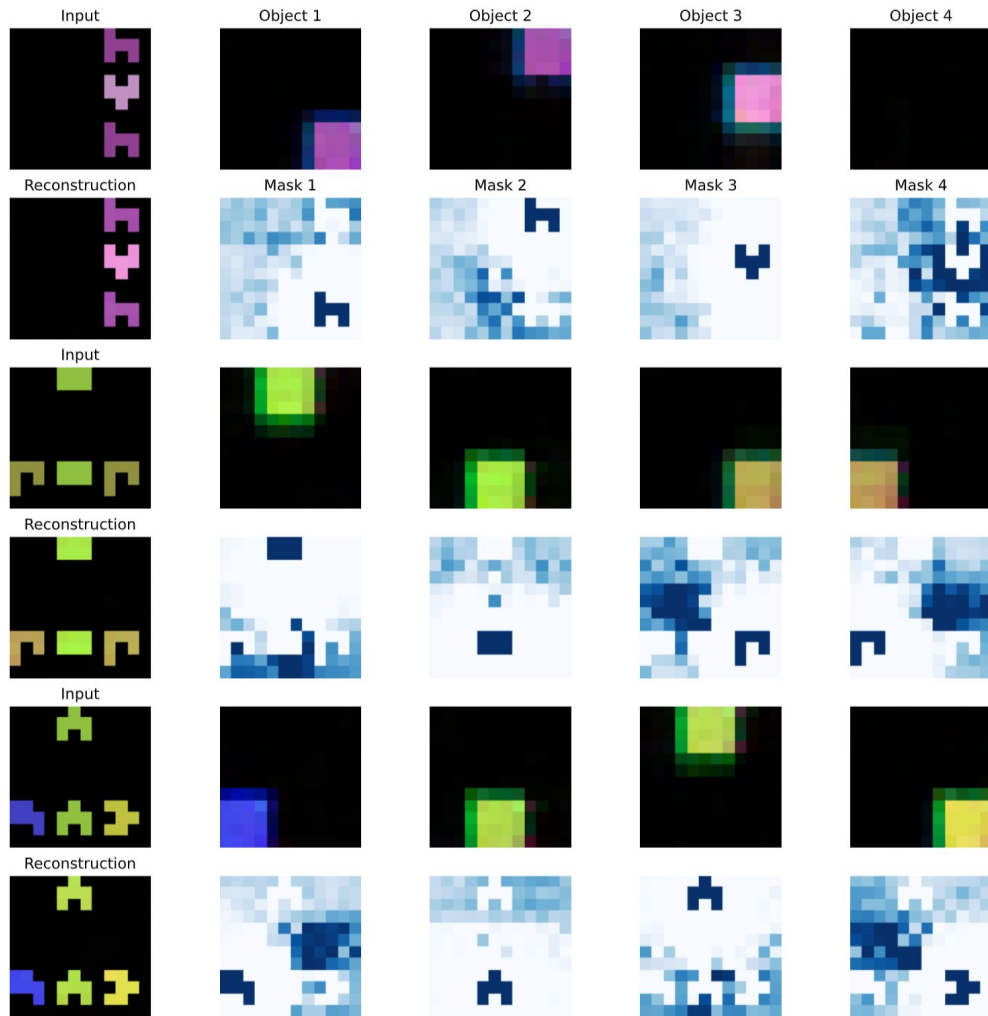
Set	Task Model	All			Between			Occurs			Same			XOccurs		
		100	1000	5000	100	1000	5000	100	1000	5000	100	1000	5000	100	1000	5000
Hex.	DNF	0.94	0.97	0.98	0.90	0.99	0.99	0.56	0.99	0.99	0.94	1.00	1.00	0.49	0.80	0.93
	DNF-h	0.98	0.99	0.99	0.95	1.00	0.99	0.62	0.99	0.99	0.97	1.00	1.00	0.50	0.98	0.96
	DNF-h+t	0.51	0.55	0.92	0.91	0.97	0.98	0.50	0.79	0.96	0.53	1.00	0.98	0.48	0.51	0.51
	DNF-hi	0.98	0.99	1.00	0.97	0.99	1.00	0.69	0.99	0.99	0.97	1.00	1.00	0.51	0.99	0.99
	DNF-r	0.94	0.98	0.98	0.84	1.00	0.99	0.63	0.99	0.99	0.96	1.00	1.00	0.51	0.94	0.51
	PrediNet	0.85	0.95	0.96	0.66	0.99	0.99	0.57	0.95	0.97	0.99	1.00	1.00	0.50	0.58	0.95
Stripe	DNF	0.91	0.97	0.95	0.81	0.98	0.99	0.57	0.97	0.99	0.93	0.99	1.00	0.49	0.88	0.94
	DNF-h	0.93	0.98	0.99	0.89	0.99	0.99	0.57	0.97	0.99	0.96	1.00	1.00	0.51	0.98	0.97
	DNF-h+t	0.52	0.53	0.93	0.92	0.97	0.95	0.49	0.84	0.86	0.49	0.99	0.97	0.48	0.50	0.51
	DNF-hi	0.95	0.99	0.99	0.94	0.99	0.99	0.63	0.96	0.99	0.97	1.00	1.00	0.49	0.96	0.97
	DNF-r	0.92	0.97	0.98	0.81	0.99	0.99	0.64	0.95	0.98	0.98	1.00	1.00	0.52	0.92	0.51
	PrediNet	0.84	0.93	0.92	0.64	0.99	0.99	0.54	0.94	0.92	0.99	0.99	1.00	0.51	0.61	0.93

The performance of all models in the hexominoes and stripes test sets are similar to pentominoes.

Does image reconstruction improve DNF model performance?



Set	Task Model	All			Between			Occurs			Same			XOccurs		
		100	1000	5000	100	1000	5000	100	1000	5000	100	1000	5000	100	1000	5000
Hex.	DNF	0.94	0.97	0.98	0.90	0.99	0.99	0.56	0.99	0.99	0.94	1.00	1.00	0.49	0.80	0.93
	→ DNF-h	0.98	0.99	0.99	0.95	1.00	0.99	0.62	0.99	0.99	0.97	1.00	1.00	0.50	0.98	0.96
	DNF-h+t	0.51	0.55	0.92	0.91	0.97	0.98	0.50	0.79	0.96	0.53	1.00	0.98	0.48	0.51	0.51
	→ DNF-hi	0.98	0.99	1.00	0.97	0.99	1.00	0.69	0.99	0.99	0.97	1.00	1.00	0.51	0.99	0.99
	DNF-r	0.94	0.98	0.98	0.84	1.00	0.99	0.63	0.99	0.99	0.96	1.00	1.00	0.51	0.94	0.51
	PrediNet	0.85	0.95	0.96	0.66	0.99	0.99	0.57	0.95	0.97	0.99	1.00	1.00	0.50	0.58	0.95

Comparing DNF-h to DNF-hi which has image reconstruction from selected objects, we do not observe any improvement above 5% despite the extra computation required to reconstruct the images.



Example image reconstructions

Can we extract symbolic rules in an image classification task?

Set	Task Model	All			Between			Occurs			Same			XOccurs		
		100	1000	5000	100	1000	5000	100	1000	5000	100	1000	5000	100	1000	5000
Hex.	DNF	0.94	0.97	0.98	0.90	0.99	0.99	0.56	0.99	0.99	0.94	1.00	1.00	0.49	0.80	0.93
	DNF-h	0.98	0.99	0.99	0.95	1.00	0.99	0.62	0.99	0.99	0.97	1.00	1.00	0.50	0.98	0.96
	DNF-h+t	0.51	0.55	0.92	0.91	0.97	0.98	0.50	0.79	0.96	0.53	1.00	0.98	0.48	0.51	0.51
	DNF-hi	0.98	0.99	1.00	0.97	0.99	1.00	0.69	0.99	0.99	0.97	1.00	1.00	0.51	0.99	0.99
	DNF-r	0.94	0.98	0.98	0.84	1.00	0.99	0.63	0.99	0.99	0.96	1.00	1.00	0.51	0.94	0.51
	PrediNet	0.85	0.95	0.96	0.66	0.99	0.99	0.57	0.95	0.97	0.99	1.00	1.00	0.50	0.58	0.95

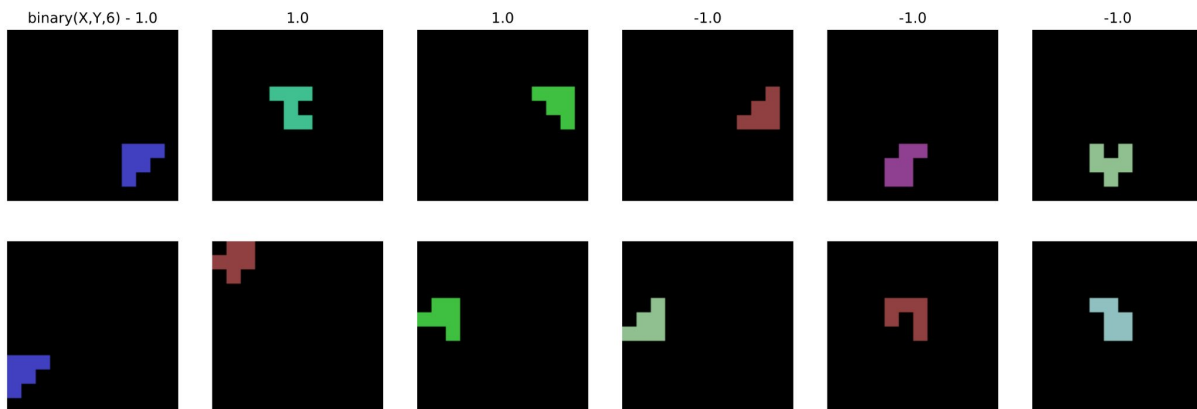
Comparing DNF-h to DNF-h+t (thresholded weights), only if DNF-h achieves >0.99 accuracy does DNF-h+t consistently yield better than random performance. This suggests the difficulty of learning both the meaning of the predicates as well as the rules in tandem.

What do the learnt predicates mean?

We remove one atom at a time and compute the drop in accuracy. We then plot the truth cases for the most important atom. For the example rule, $\text{binary}(X,Y,6)$ drops accuracy by 18%.

Object arguments (top and bottom rows) that make $\text{binary}(X,Y,6)$ true and false exhibit no common pattern to principal concepts of the dataset.

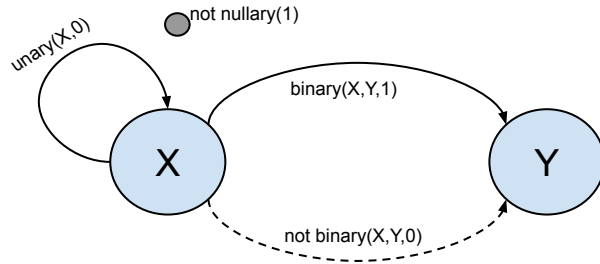
between :- unary(X,4), not unary(Y,6), binary(Y,Z,6), not binary(Y,Z,9),
not binary(Z,Y,3), binary(Z,Y,5), not binary(Z,Y,9), binary(Z,Y,14).



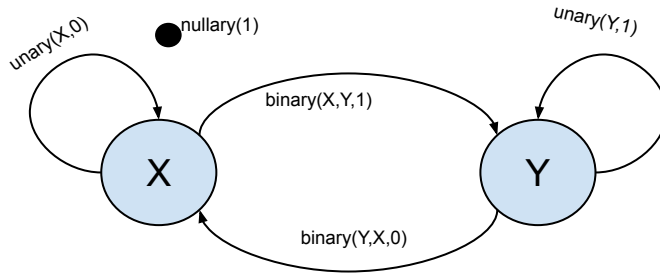
Thank you for listening

Subgraph Set Isomorphism

given a graph \mathcal{G} and a set of graphs $\mathbb{H} = \{\mathcal{H}_1, \mathcal{H}_2, \dots, \mathcal{H}_n\}$, determine whether $\exists \mathcal{L}, i \mid \mathcal{L} \subseteq \mathcal{G} \wedge \mathcal{L} \simeq \mathcal{H}_i$ where \simeq is graph isomorphism. Given positive E^+ and negative E^- examples of graphs that are subgraph isomorphic to some \mathcal{H}_i , the objective is to learn an \mathbb{H} .



$t :- \text{not nullary}(1), \text{unary}(X,0),$
 $\text{not binary}(X,Y,0), \text{binary}(X,Y,1).$




$t :- \text{nullary}(1), \text{unary}(X,0), \text{unary}(Y,1),$
 $\text{binary}(X,Y,1), \text{binary}(Y,X,0).$

Subgraph Set Isomorphism

given a graph \mathcal{G} and a set of graphs $\mathbb{H} = \{\mathcal{H}_1, \mathcal{H}_2, \dots, \mathcal{H}_n\}$, determine whether $\exists \mathcal{L}, i \mid \mathcal{L} \subseteq \mathcal{G} \wedge \mathcal{L} \simeq \mathcal{H}_i$ where \simeq is graph isomorphism. Given positive E^+ and negative E^- examples of graphs that are subgraph isomorphic to some \mathcal{H}_i , the objective is to learn an \mathbb{H} .

Average rule length



Difficulty	$ V(\mathcal{G}) $	Nullary	Unary	Binary	$ V(\mathcal{H}_i) $	Max. $ \mathbb{H} $	Avg. $ E(\mathcal{H}_i) $
Easy	3	2	2	2	2	3	7.29
Medium	4	4	5	6	3	4	37.27
Hard	4	6	7	8	3	5	50.62

Example Medium Size Rule

Example target rule generated for the medium dataset size. This is the task that FastLAS does not terminate after 16 hours. Note that `obj()` predicate as well as the uniqueness of variables are added for safe ASP representation.

t :- nullary(0), nullary(2), not nullary(3), unary(V0,0), unary(V0,1), not unary(V0,2),
not unary(V0,3), not unary(V1,0), not unary(V1,1), not unary(V1,2), not unary(V1,3),
unary(V2,0), unary(V2,2), binary(V0,V1,0), binary(V0,V1,1), binary(V0,V1,2),
binary(V0,V1,3), not binary(V0,V1,4), not binary(V0,V1,5), not binary(V0,V2,0),
binary(V0,V2,2), not binary(V0,V2,3), not binary(V0,V2,5), binary(V1,V0,1),
not binary(V1,V0,2), not binary(V1,V0,3), not binary(V1,V0,4), not binary(V1,V0,5),
binary(V1,V2,0), binary(V1,V2,3), not binary(V1,V2,4), binary(V2,V0,0),
not binary(V2,V0,2), binary(V2,V0,3), not binary(V2,V0,4), not binary(V2,V1,1),
not binary(V2,V1,3), not binary(V2,V1,4), not binary(V2,V1,5), obj(V2), V2 != V0,
V2 != V1, obj(V0), V0 != V1, obj(V1).

t :- nullary(0), not nullary(1), not nullary(2), unary(V0,0), unary(V0,1), unary(V0,3),
unary(V0,4), not unary(V1,0), not unary(V1,2), not unary(V1,4), not unary(V2,0),
not unary(V2,1), unary(V2,2), not unary(V2,3), not unary(V2,4), not binary(V0,V1,2),
binary(V0,V1,3), binary(V0,V1,4), binary(V0,V1,5), not binary(V0,V2,1), binary(V0,V2,2),
binary(V0,V2,3), not binary(V0,V2,4), binary(V0,V2,5), not binary(V1,V0,2),
binary(V1,V0,3), not binary(V1,V0,4), not binary(V1,V0,5), not binary(V1,V2,4),
not binary(V1,V2,5), not binary(V2,V0,0), not binary(V2,V0,1), not binary(V2,V0,2),
binary(V2,V0,4), not binary(V2,V1,0), not binary(V2,V1,1), not binary(V2,V1,4),
obj(V2), V2 != V0, V2 != V1, obj(V0), V0 != V1, obj(V1).

Subgraph Set Isomorphism Results

Difficulty	Test Accuracy			Training Time		
	Easy	Medium	Hard	Easy	Medium	Hard
DNF	1.0 ± 0.0	1.00 ± 0.0	1.00 ± 0.00	127.13 ± 4.17	136.90 ± 10.00	129.56 ± 5.77
DNF+t	1.0 ± 0.0	0.99 ± 0.0	0.99 ± 0.01	125.02 ± 6.53	135.67 ± 8.56	143.67 ± 22.60
FastLASv3	1.0 ± 0.0			29.85 ± 0.69		
ILASP-2i	1.0 ± 0.0			3336.00 ± 994.45		

DNF Layer with Input Noise

We add input noise to the subgraph set isomorphism dataset by randomly flipping the truth values of $E(G)$ in the training set with a fixed probability. We observe that the DNF layer performs well up to 0.3 where the median accuracy drops below 0.9. The pruning and thresholding steps seem to improve the performance with lower levels of noise, likely because incorrect weights are removed against a non-noisy validation set.

Difficulty	Easy			Medium			Hard		
	Noise	0.00	0.15	0.30	0.00	0.15	0.30	0.00	0.15
DNF	1.00	0.89	0.82	1.00	0.98	0.89	1.00	0.98	0.86
DNF+t	1.00	1.00	0.84	0.99	0.99	0.99	0.99	0.99	0.74