Adventures in Neurosymbolic Machine Learning

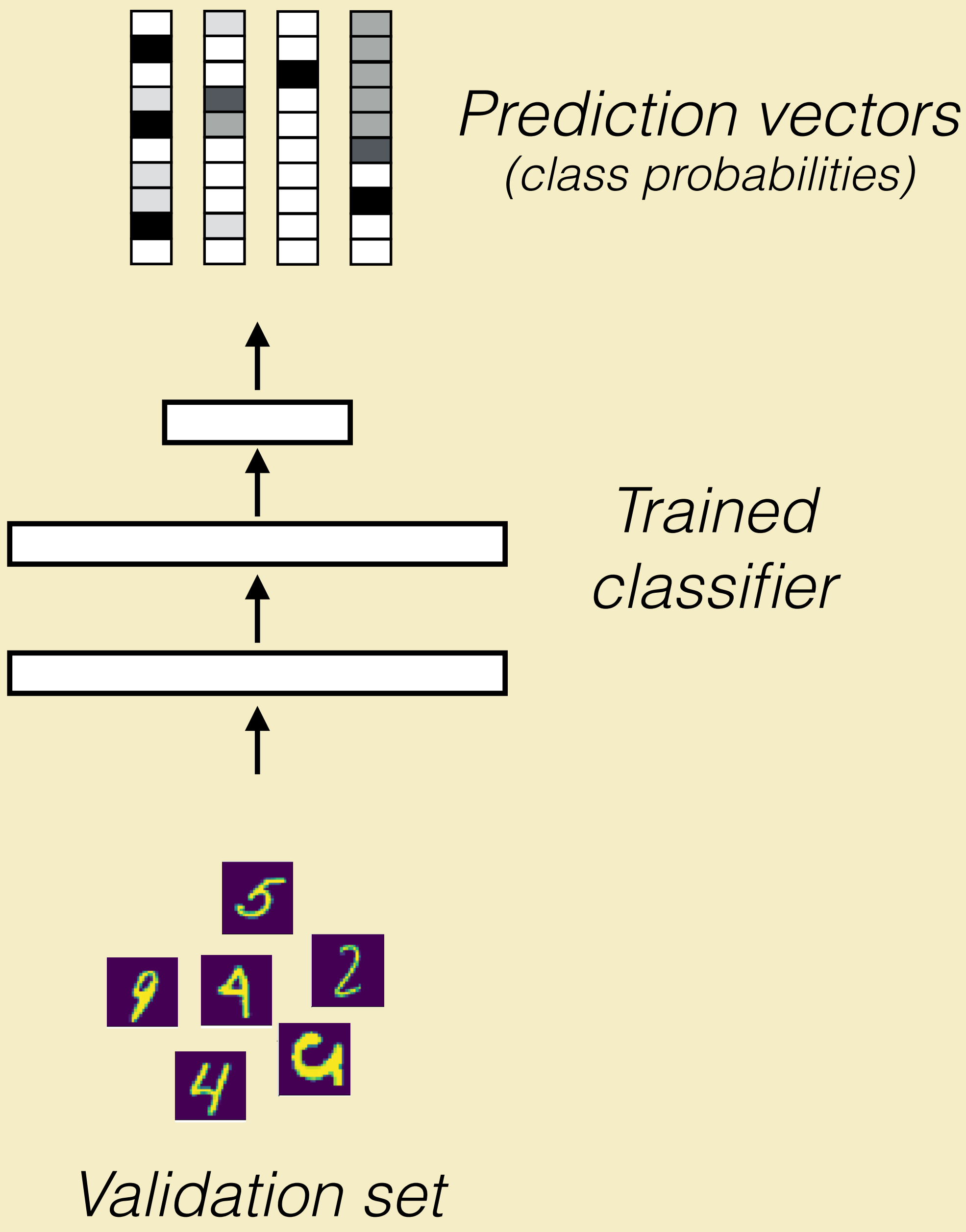
symbols to vectors
vectors to symbols
plus other stuff

Charles Sutton

9 April 2018

Companion Site: bit.ly/adventures-neurosymbolic

BONUS! Debugging tools

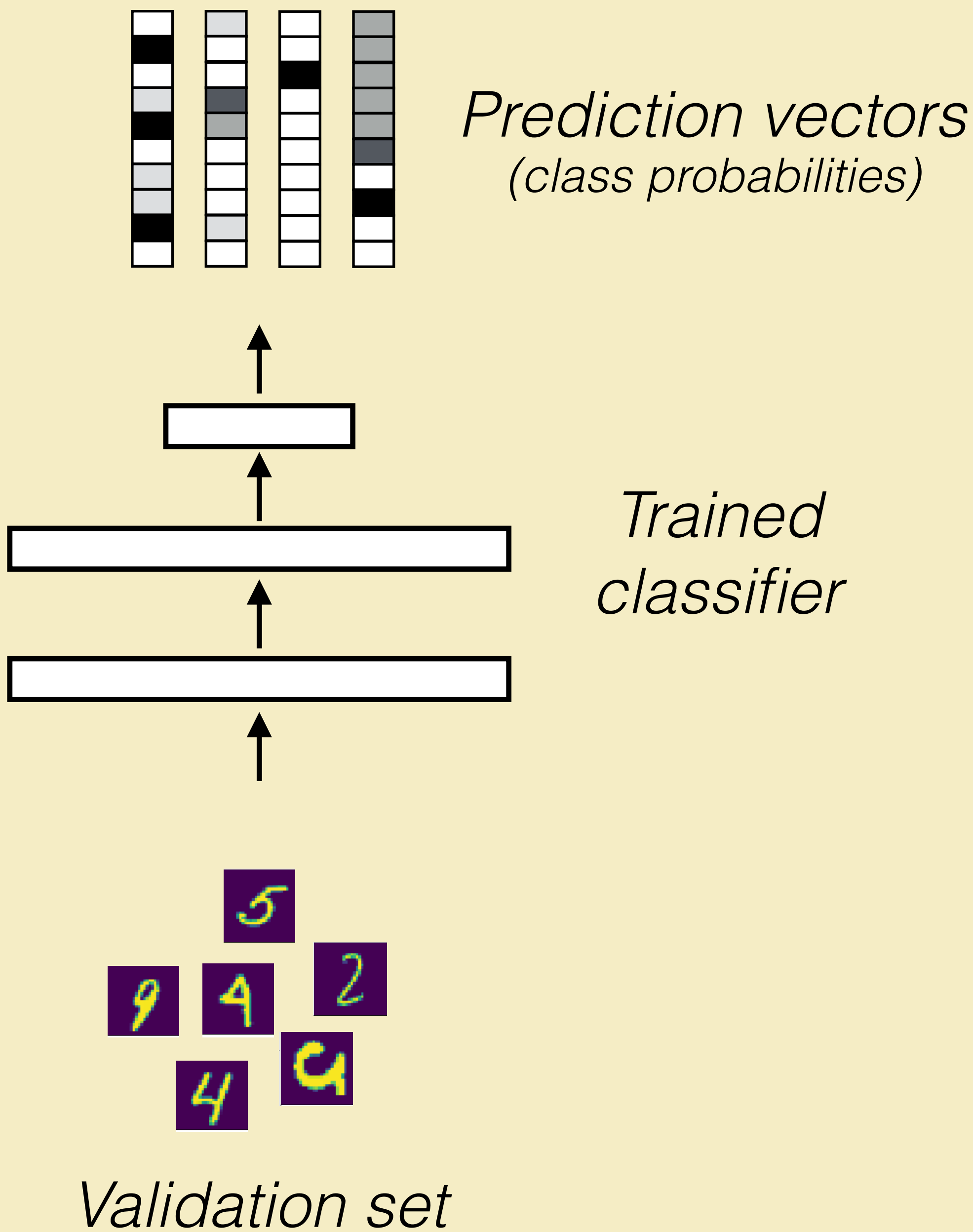


Programming is to debugging
as
Differentiable programming is to
error analysis

[Xu, Park, Chang, Sutton, 2018]

http://xuk.ai/darksight/

Error Analysis via Dark Knowledge



Interpret via visualization

Visualize via dimension reduction

Dimension reduction via dark knowledge

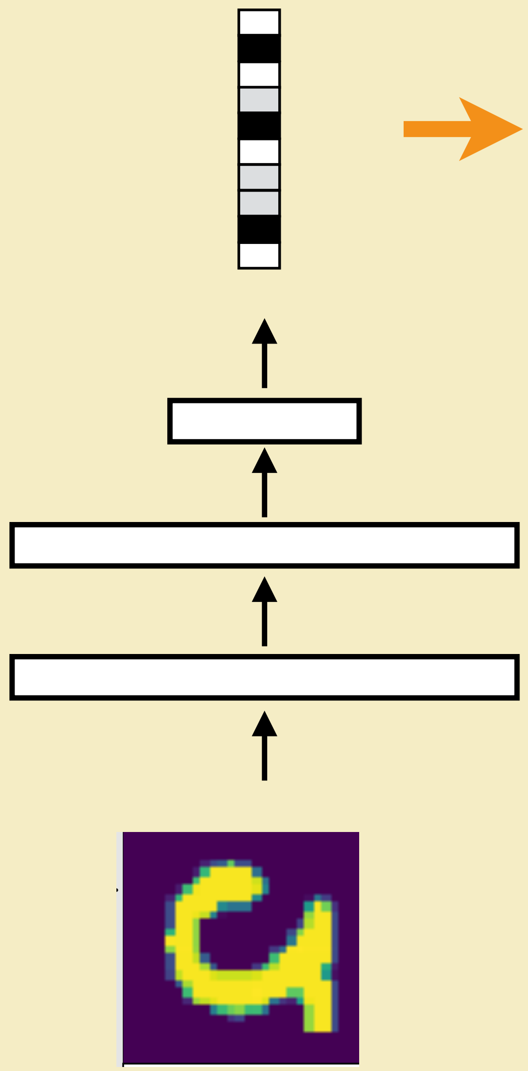
Why dark knowledge?

airplane	0.95	0.95
bird	0.04	0
motorcycle	0	0
truck	0	0
frog	0	0.05

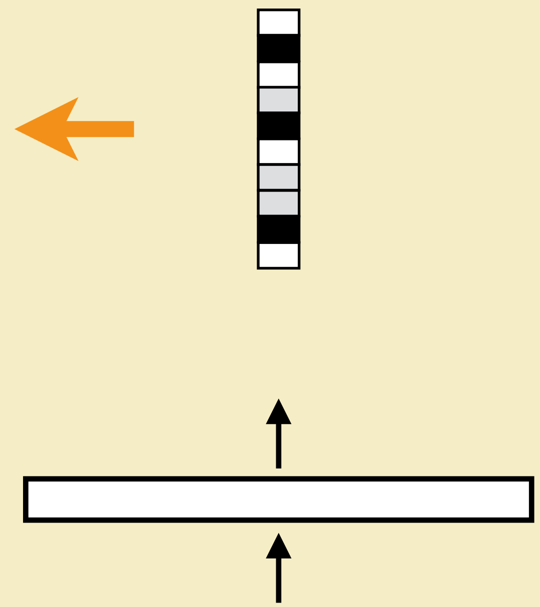
Maybe something's wrong

BONUS! *[Xu, Park, Chang, Sutton, 2018]*

Error Analysis via Dark Knowledge



minimize KL



Trained classifier

Class-conditional Gaussian

[1.5, 0.03]

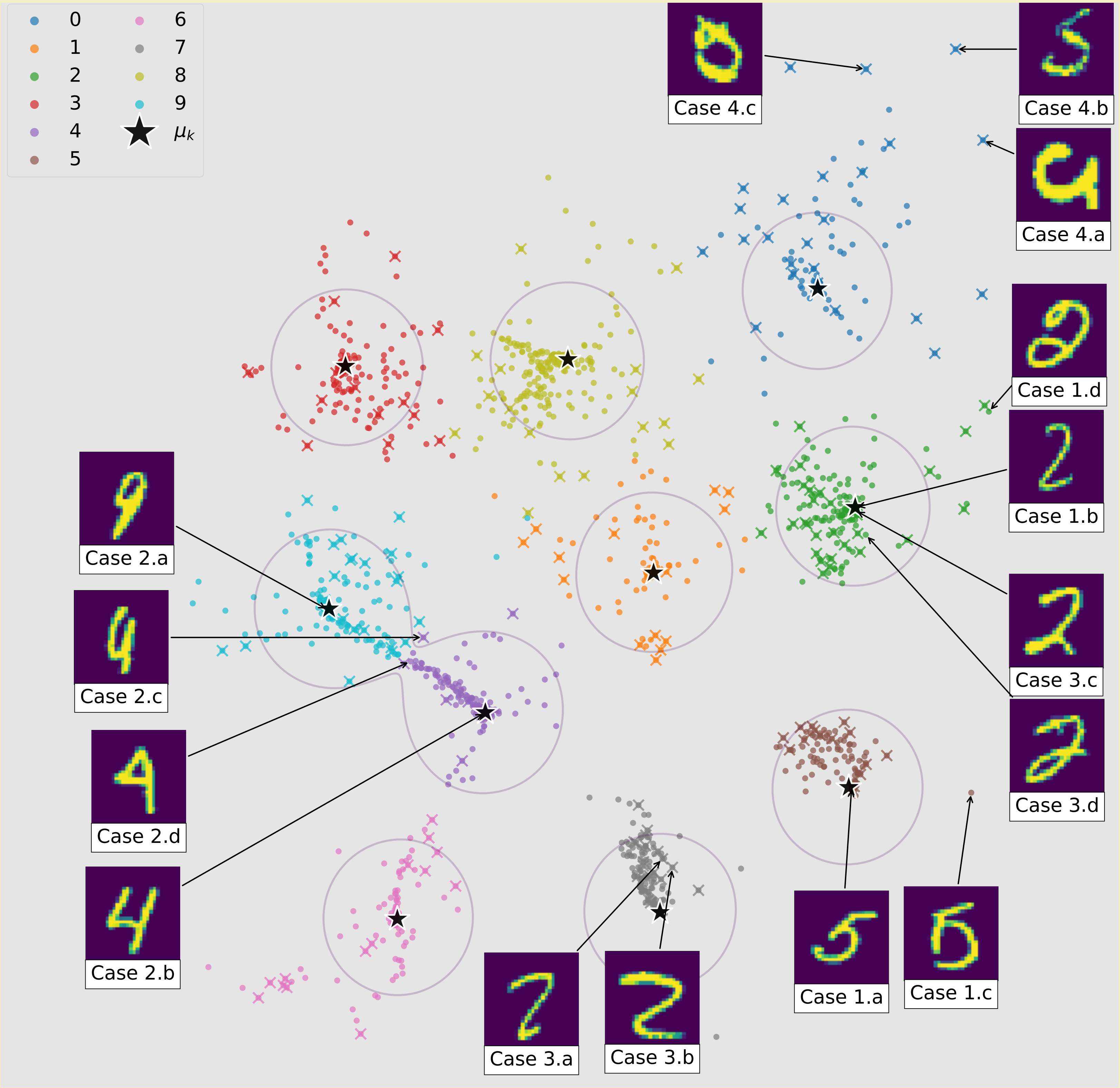
2-D representation

Validation Image

model compression and dimension reduction in one objective

BONUS! *[Xu, Park, Chang, Sutton, 2018]*

http://xuk.ai/darksight/



Each data item gets
2-D location

such that

Gaussian classifier 2-D

matches

original deep classifier
on original data

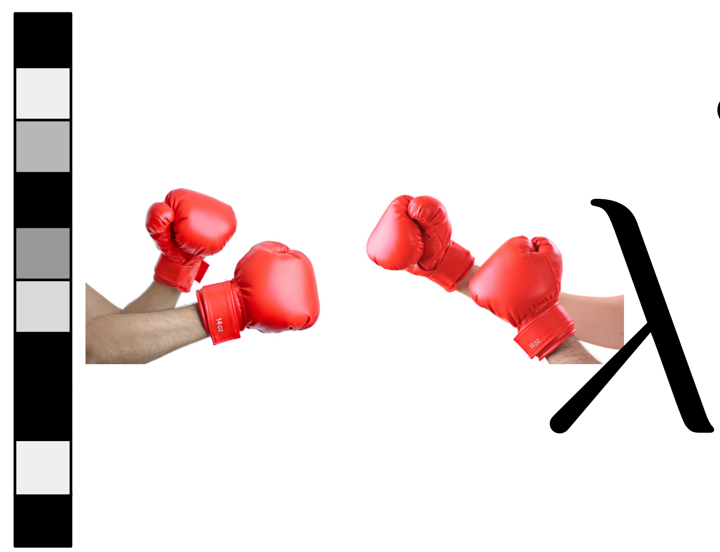
BONUS!

http://xuk.ai/darksight/

*Unify *this*!!!*

Subsymbolic son of a …

Mr Continuous Representation



Professor Symbolic AI

Great for perceptual data

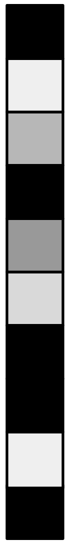
Efficient reasoning

Abstracts over lots of perceptual states

Logical and algorithmic reasoning

Why combine?

Mr Continuous Representation



- Great for perceptual data
- Efficient reasoning
- ✗ Only doing local search
- ✗ Requires lots of data for learning

λ Professor Symbolic AI

- Abstracts over lots of perceptual states
- Search has “non-local” effects
  - small change makes big difference
  - useful for transfer learning
- ✗ Impedance mismatch with perception (percepts —> symbol problem)
- ✗ Search intractable

Why not to combine?

Aristotle

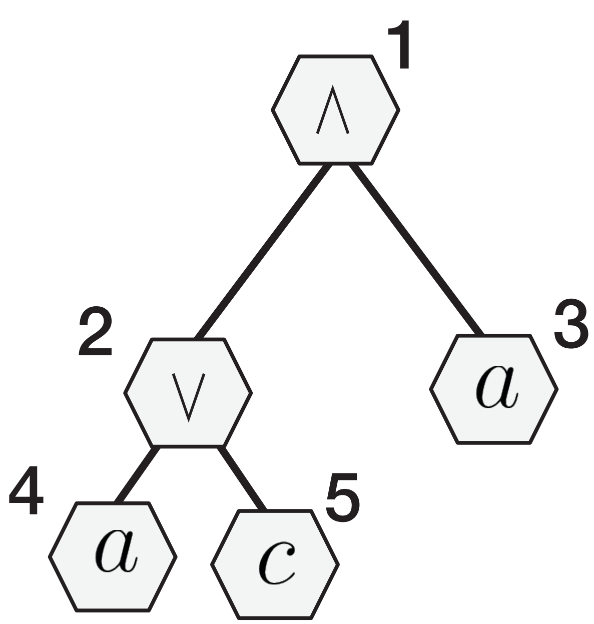
Over 2000 years of overpromising and underdelivering

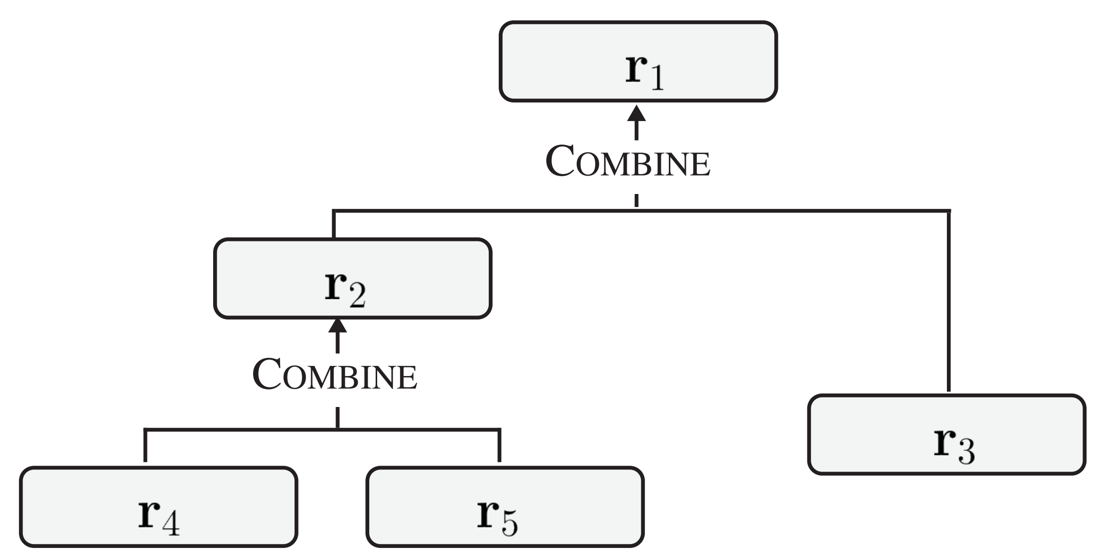
How to combine

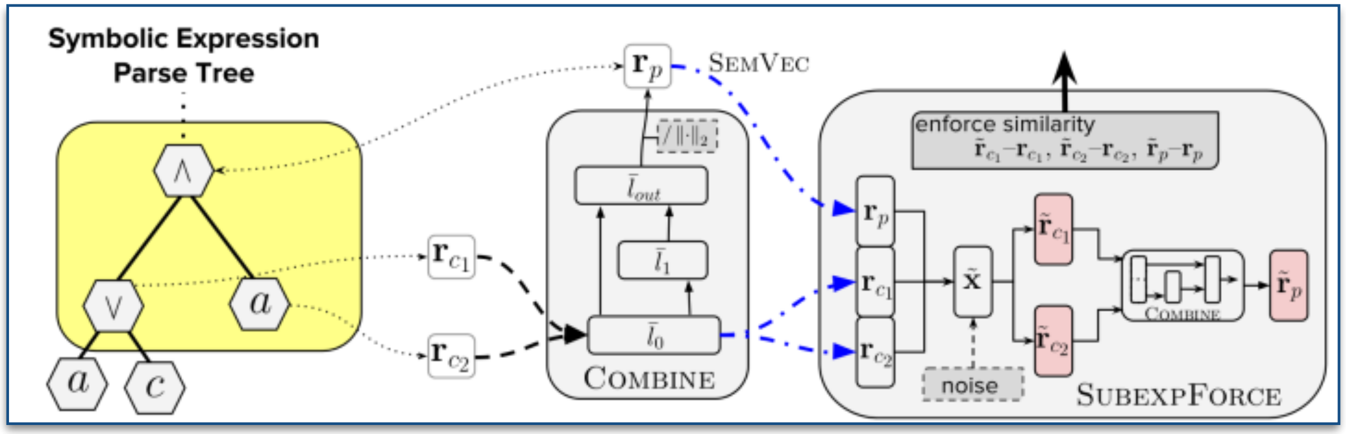
Symbols describe structure

Use gradient to learn parameters

$(a \vee c) \wedge a$

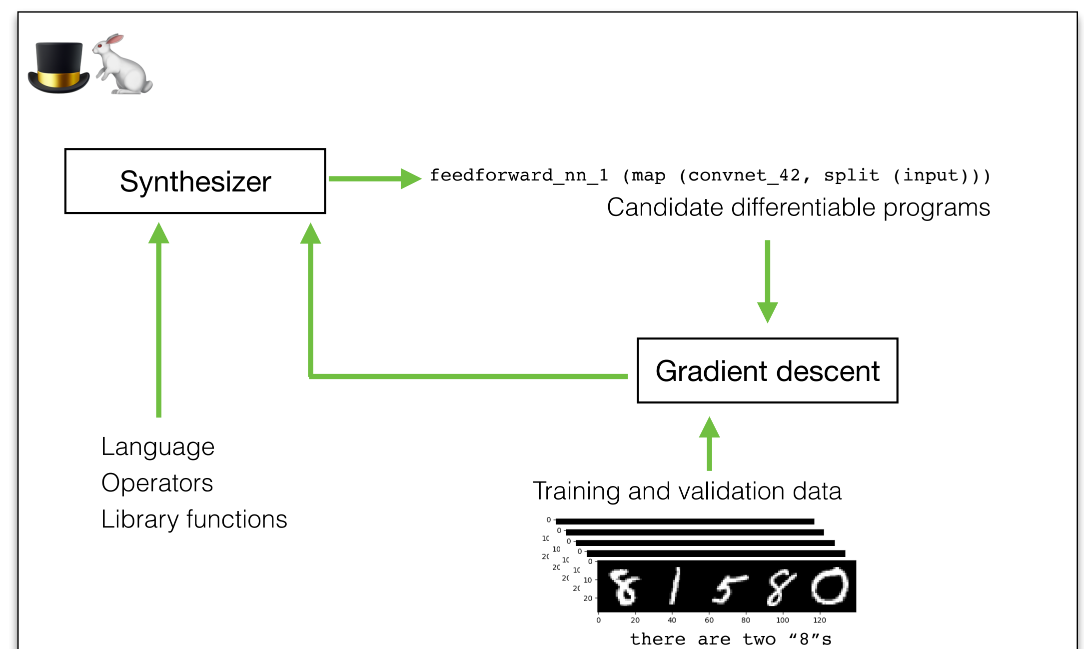






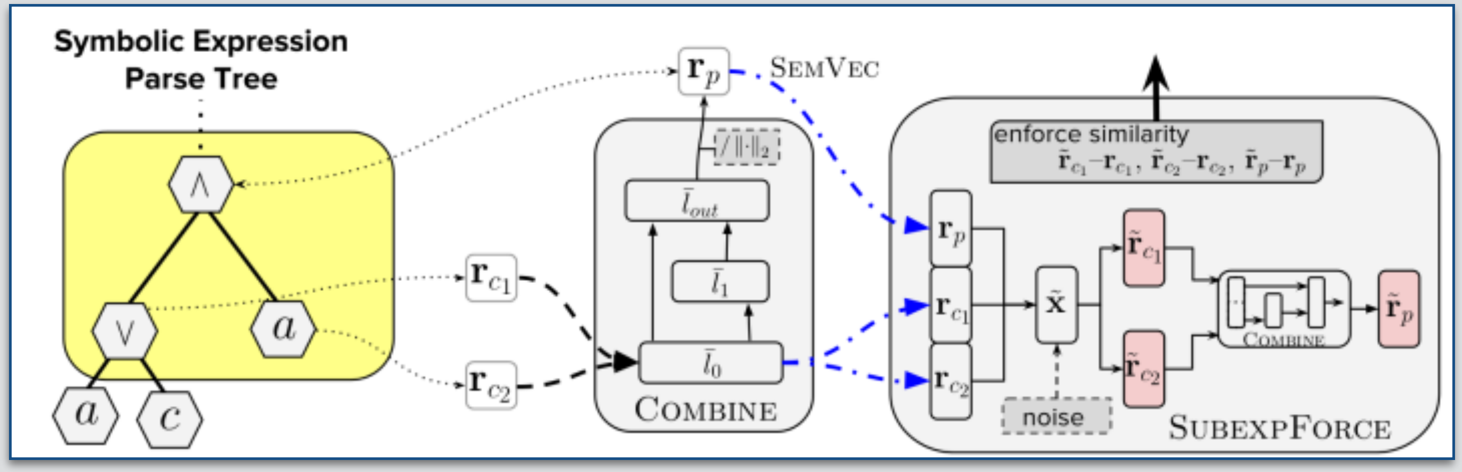
SemVecs

Representing symbolic expressions by vectors



Houdini

Representing differentiable functions by functional combinators

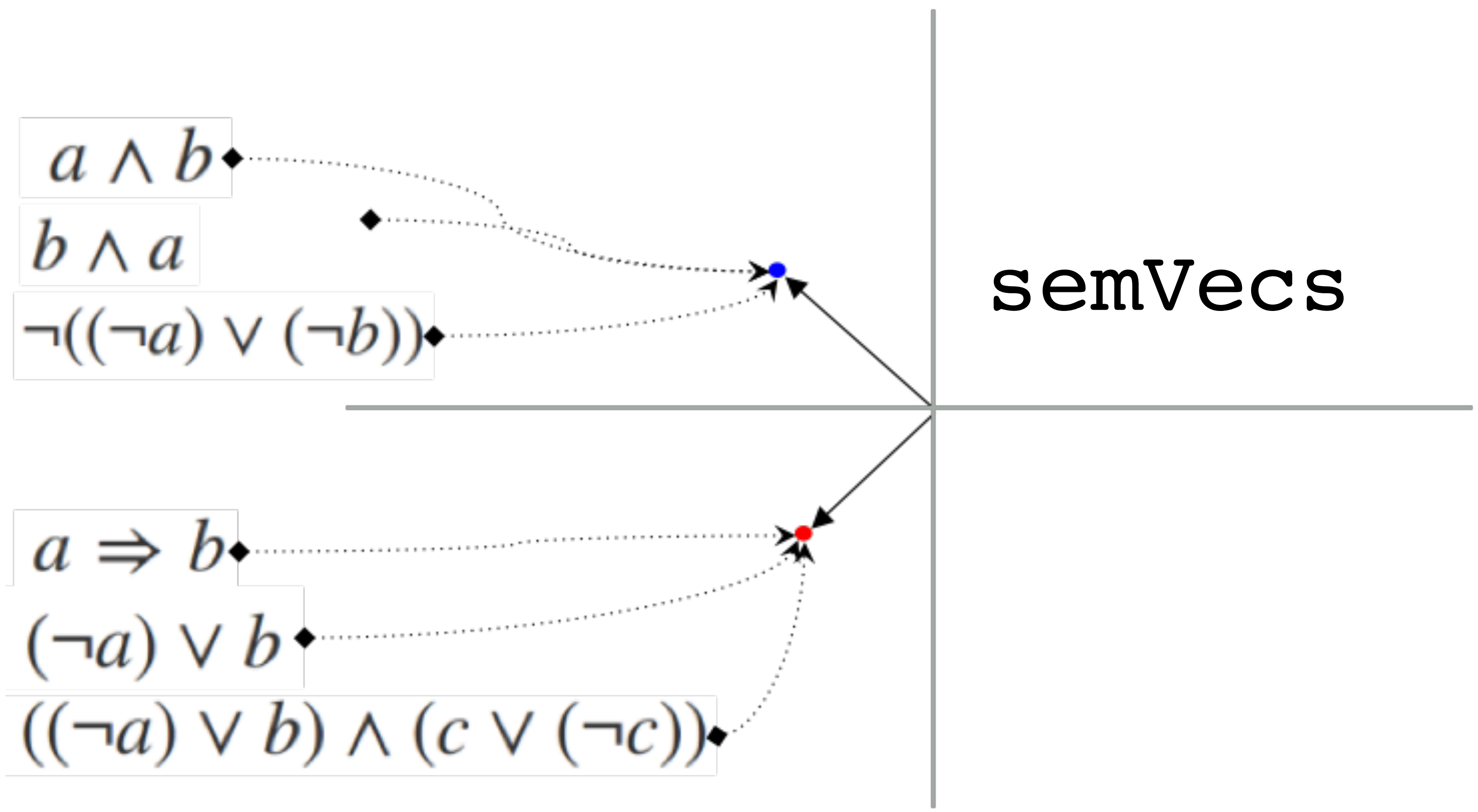


Continuous Representations of Symbolic Expressions

bit.ly/adventures-neurosymbolic

[Allamanis, Chanthirasegaran, Kohli, and Sutton, ICML 2017]

Can vectors help symbols?



In this work: semantics = equivalence

How much symbolic semantics (**semantic equivalence**) can we compress into **continuous** vector?

Want similar continuous vectors —> logically equivalent

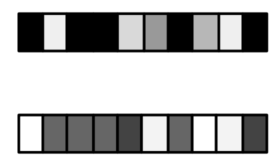
Potential Uses

Logical expressions

$$a \vee (b \implies c)$$

$$a \vee \neg b \vee c$$

Continuous vectors (**`semVecs`**)



Symbolic reasoning: ~~search~~ **pattern recognition**

Theorem Proving

[DeepMath: Irving et al, 2016] *[Zaremba et al, 2014]*

Program Synthesis

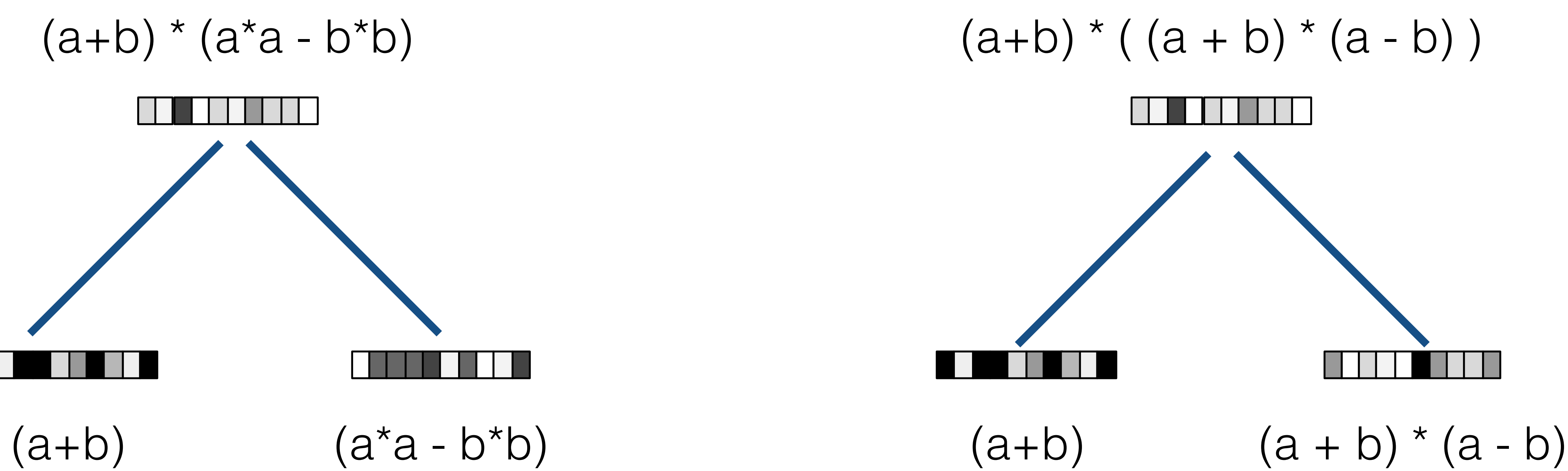
[Gulwani et al, CACM 2015]

Inductive Logic Programming

[Rocktaschel and Riedel, 2016] *[Rocktaschel and Riedel, arXiv 1705.11040 2017]*

Transfer Learning

Desiderata



Syntax directed: Semantics is compositional

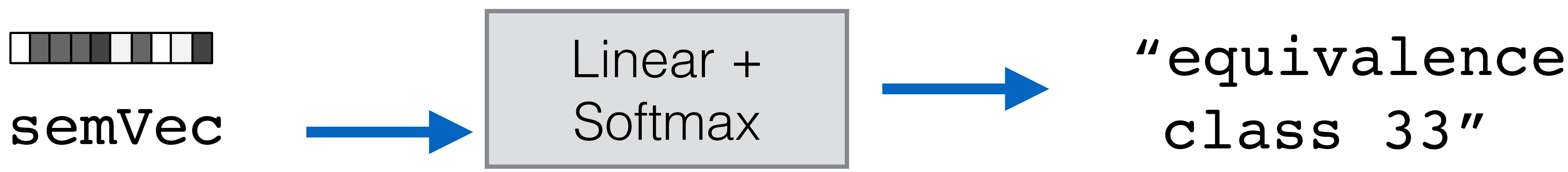
Not too much: Small syntax change —> big semantics

“man bites dog” problem

Computing semVecs

Training

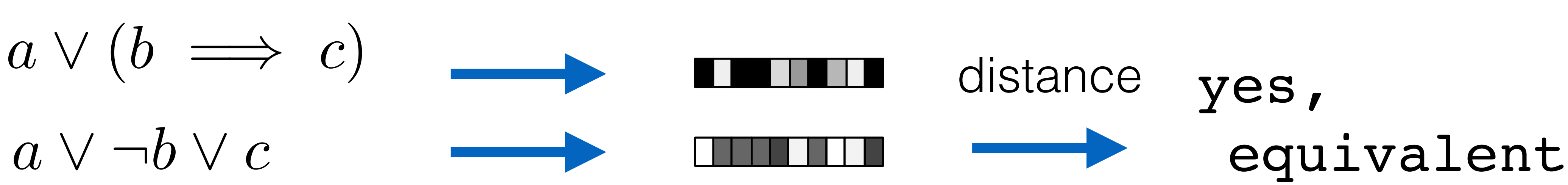
Partition training expressions into equivalence classes



Use a supervised max-margin loss

Testing

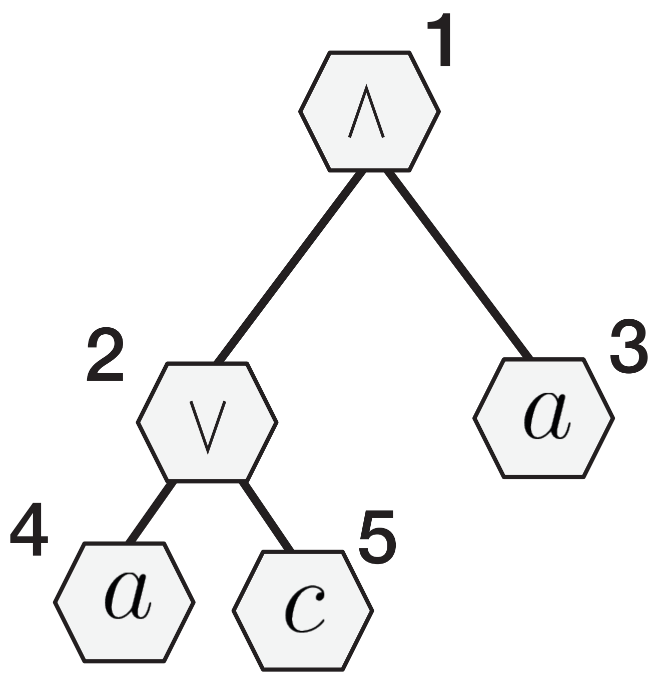
Use a semVec similarity only. Allows zero-shot learning on equiv classes.



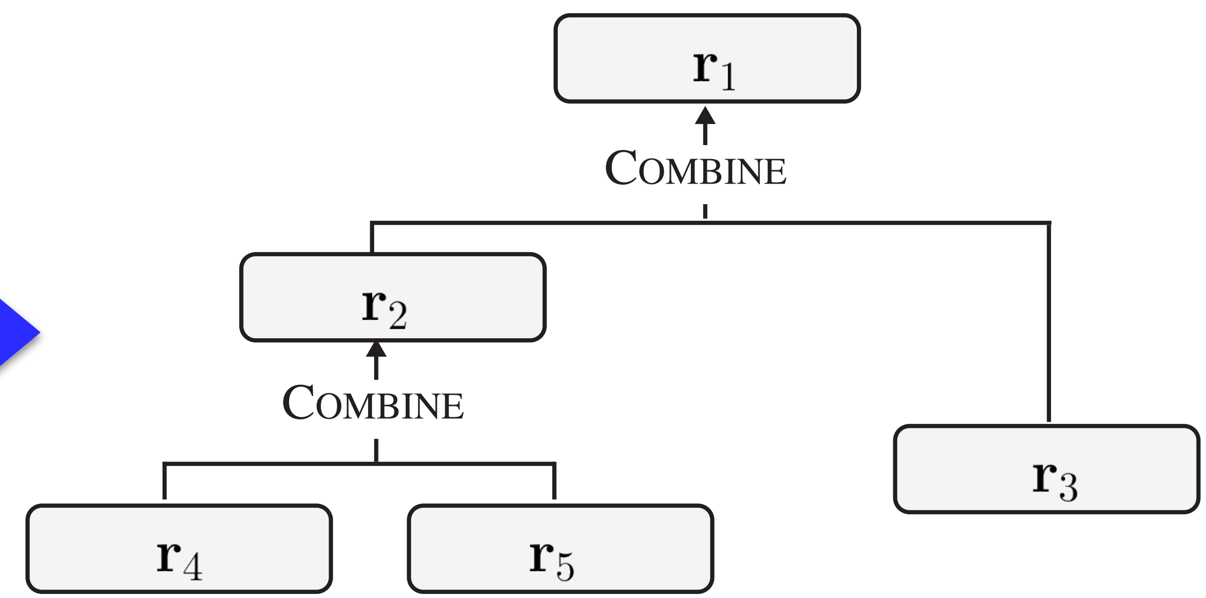
Allows zero-shot learning on equivalence classes.

Recursive NN (TreeNN)

$$(a \vee c) \wedge a$$



Syntax tree



Network architecture

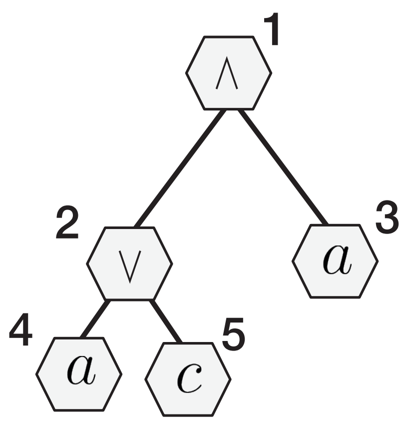
Problem: Representations mostly syntactic. Too much syntax!

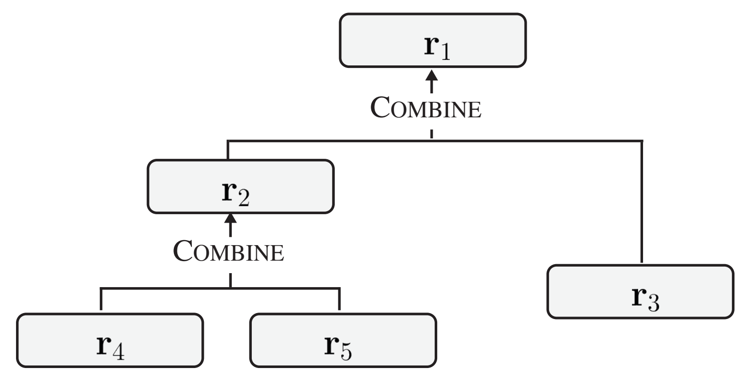
[Socher et al, 2011, 2013]

EqNet

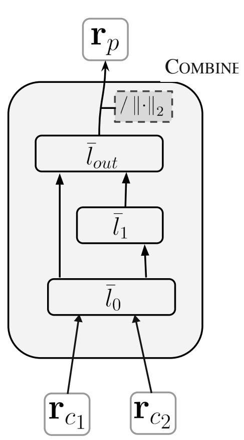
Start with TreeNNs

$(a \vee c) \wedge a$





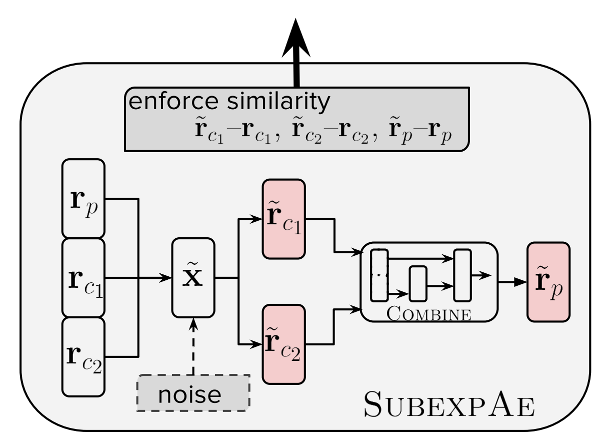
Add:



Moar! Layers!

$\|\cdot\|_2$

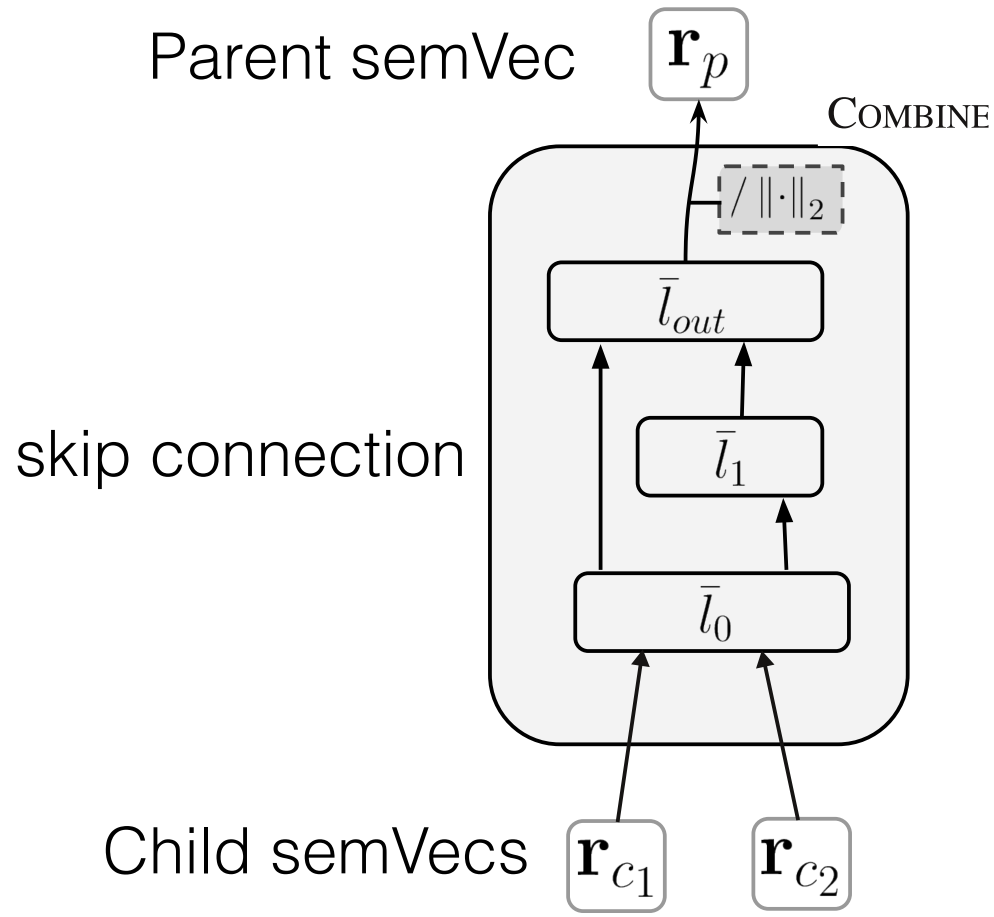
Normalization



Subexpression AE

Layers and Normalization

For one syntactic parent-child



$\textsc{Combine}\left(\mathbf{r}_{c_0}, \ldots, \mathbf{r}_{c_k}, \tau_p\right)$

$\bar{l}_0 \leftarrow [\mathbf{r}_{c_0}, \ldots, \mathbf{r}_{c_k}]$

$\bar{l}_1 \leftarrow \sigma\left(W_{i,\tau_p} \cdot \bar{l}_0\right)$

$\bar{l}_{out} \leftarrow W_{o0,\tau_p} \cdot \bar{l}_0 + W_{o1,\tau_p} \cdot \bar{l}_1$

$\textbf{return } \bar{l}_{out} / \left\|\bar{l}_{out}\right\|_2$

Big impact.

(Turns out you need both residual and normalisation together)

SubexprAE: Motivation

Semantic information is bidirectional

Not only do **children** provide info re **parents**

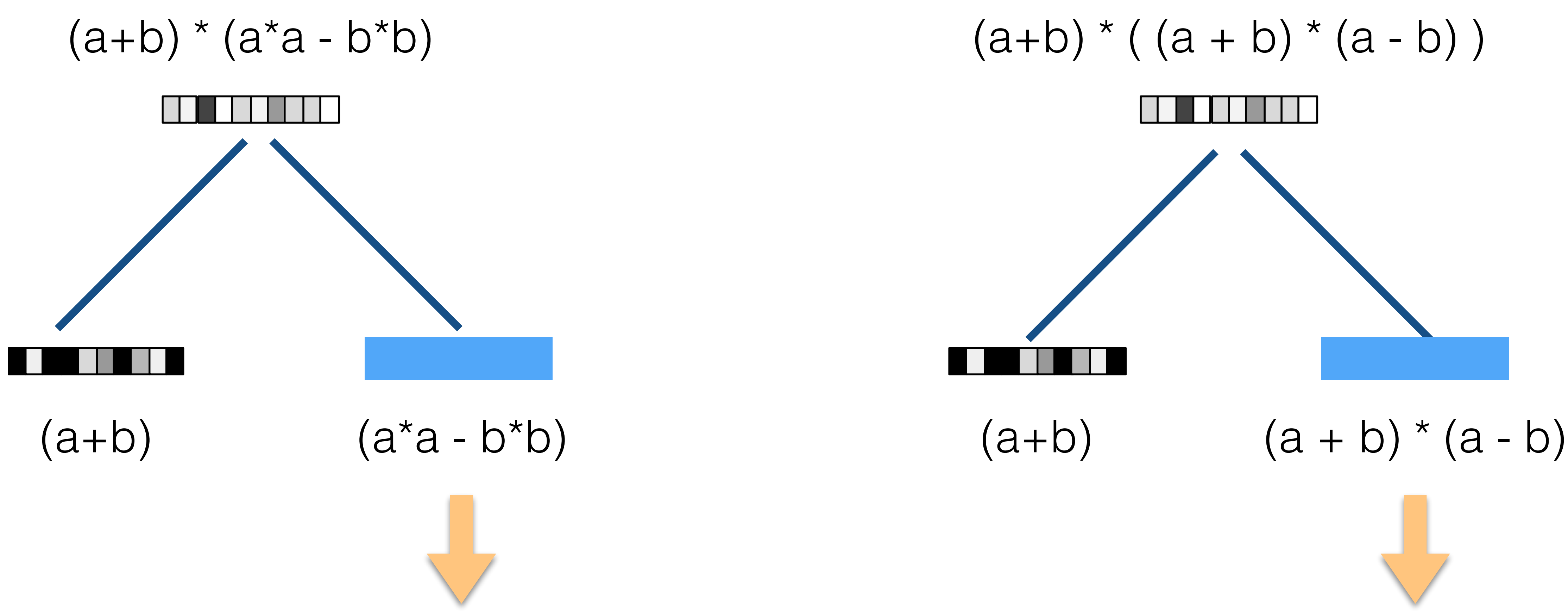
But **parents** provide info re **children**

```
uncle(?B,?A) :- parent(?Z,?A), brother(?Z,?B).
```

Unification propagates this info automatically

How to map to continuous space?

SubexprAE Motivation

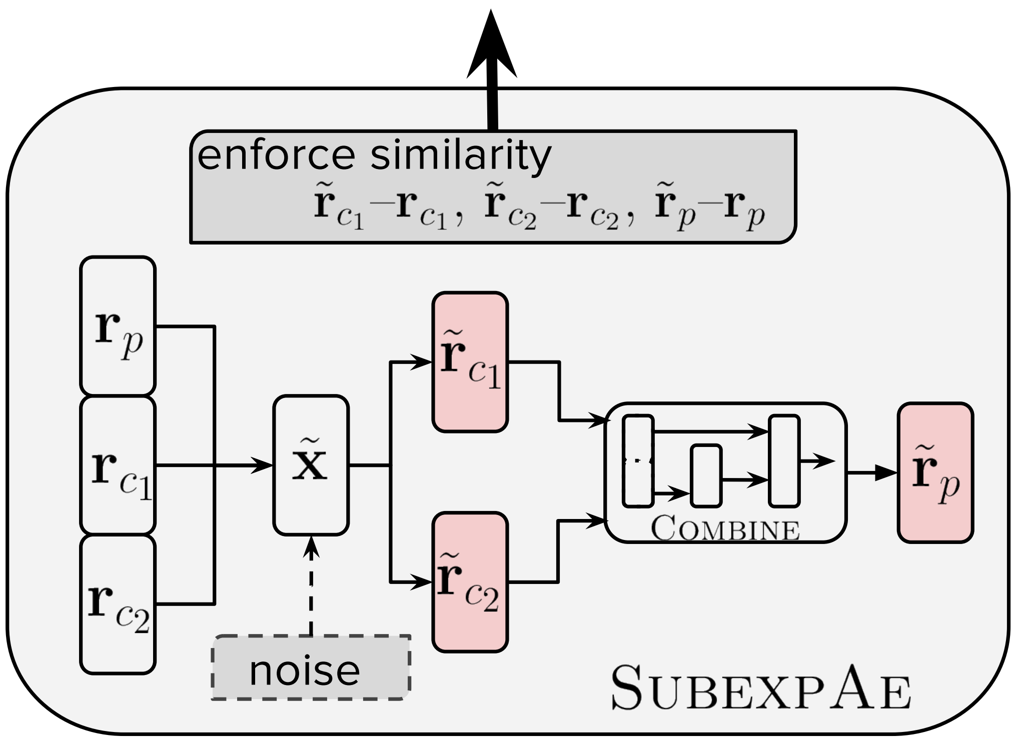


ensure this prediction problem is “easy”

semantic classes will be clustered together

Subexpression Autoencoder

For every node in syntax tree, add regularisation



Denoising autoencoder plus bottleneck on (parent, child1, child2) semVecs

Intention is

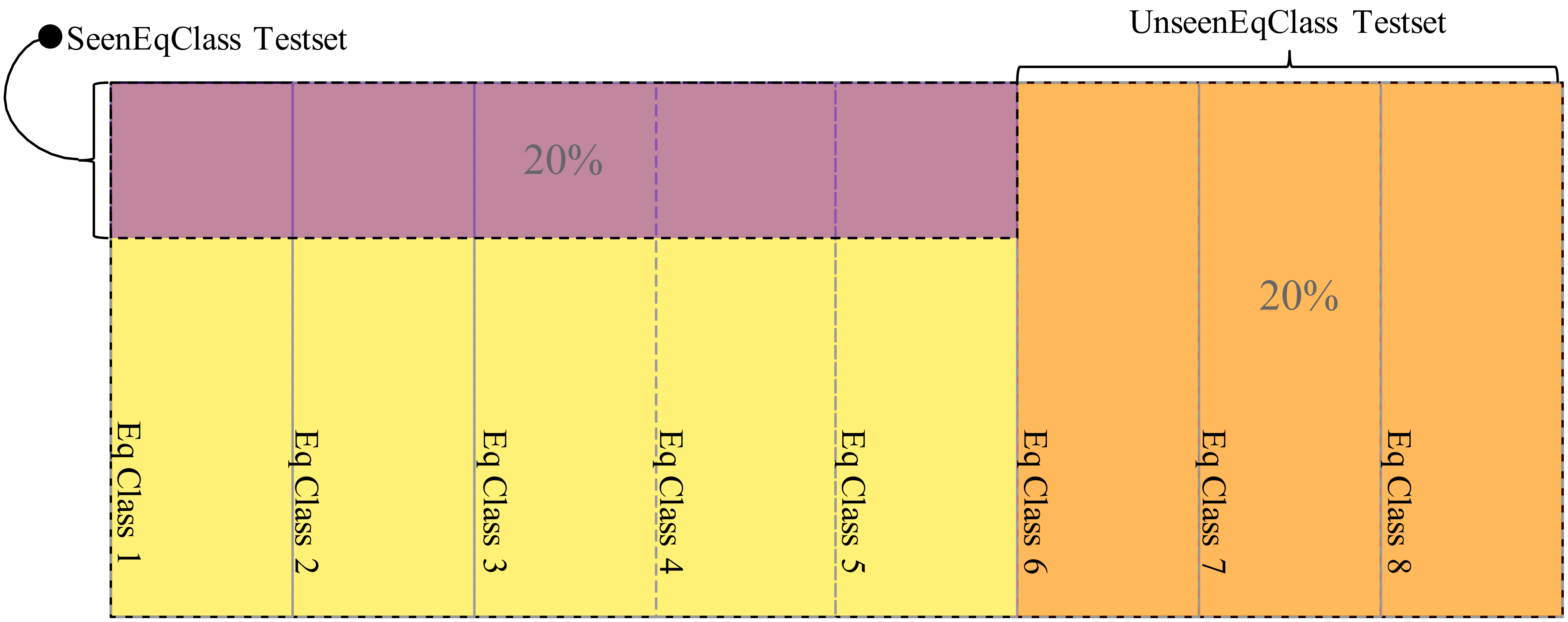
Bottleneck → Abstraction

Denoising → Reversibility

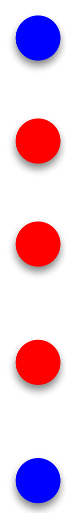
Evaluation

Dataset	# Vars	# Equiv Classes	# Exprs	H
SimpBool8	3	120	39,048	5.6
SimpBool10^S	3	191	26,304	7.2
Bool5	3	95	1,239	5.6
Bool8	3	232	257,784	6.2
Bool10^S	10	256	51,299	8.0
SimpBoolL5	10	1,342	10,050	9.9
BoolL5	10	7,312	36,050	11.8
SimpPoly5	3	47	237	5.0
SimpPoly8	3	104	3,477	5.8
SimpPoly10	3	195	57,909	6.3
oneV-Poly10	1	83	1,291	5.4
oneV-Poly13	1	677	107,725	7.1
Poly5	3	150	516	6.7
Poly8	3	1,102	11,451	9.0

Training / Test Split



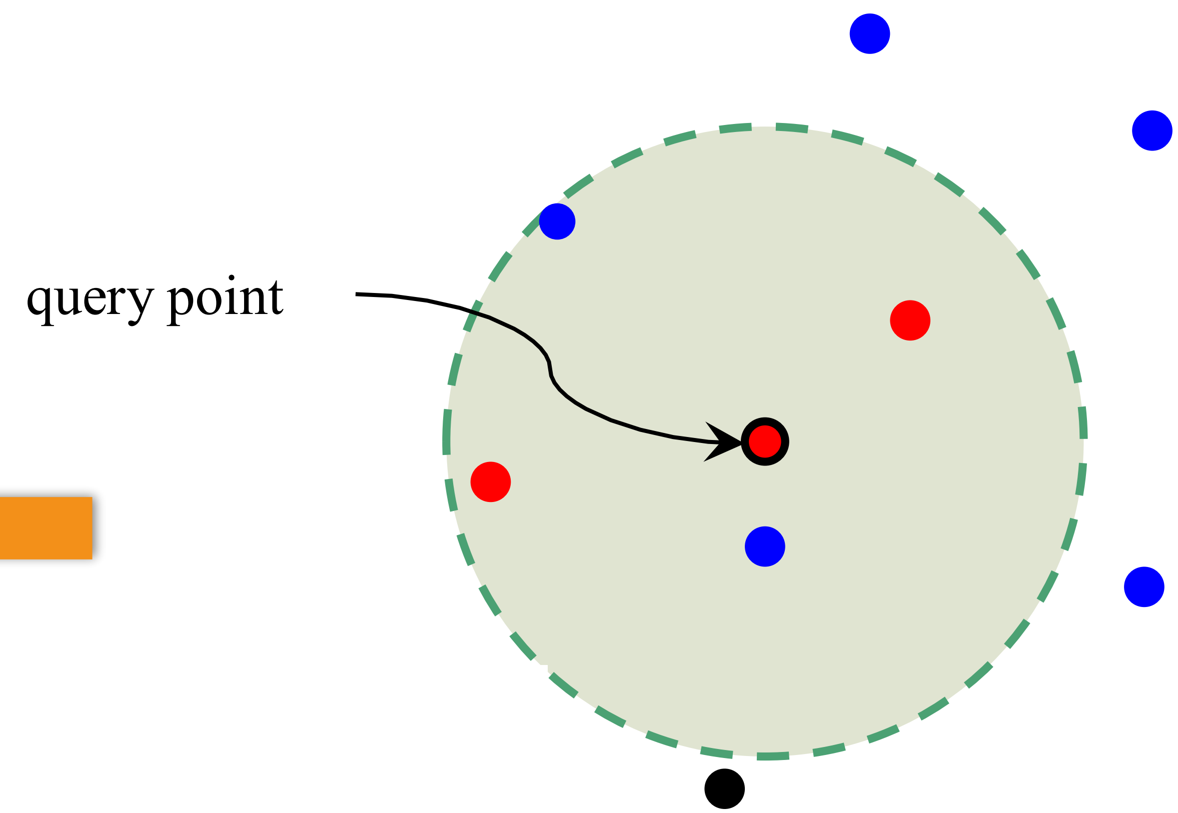
Evaluation Metric



Ranked list

(by distance to query)

Precision
and recall

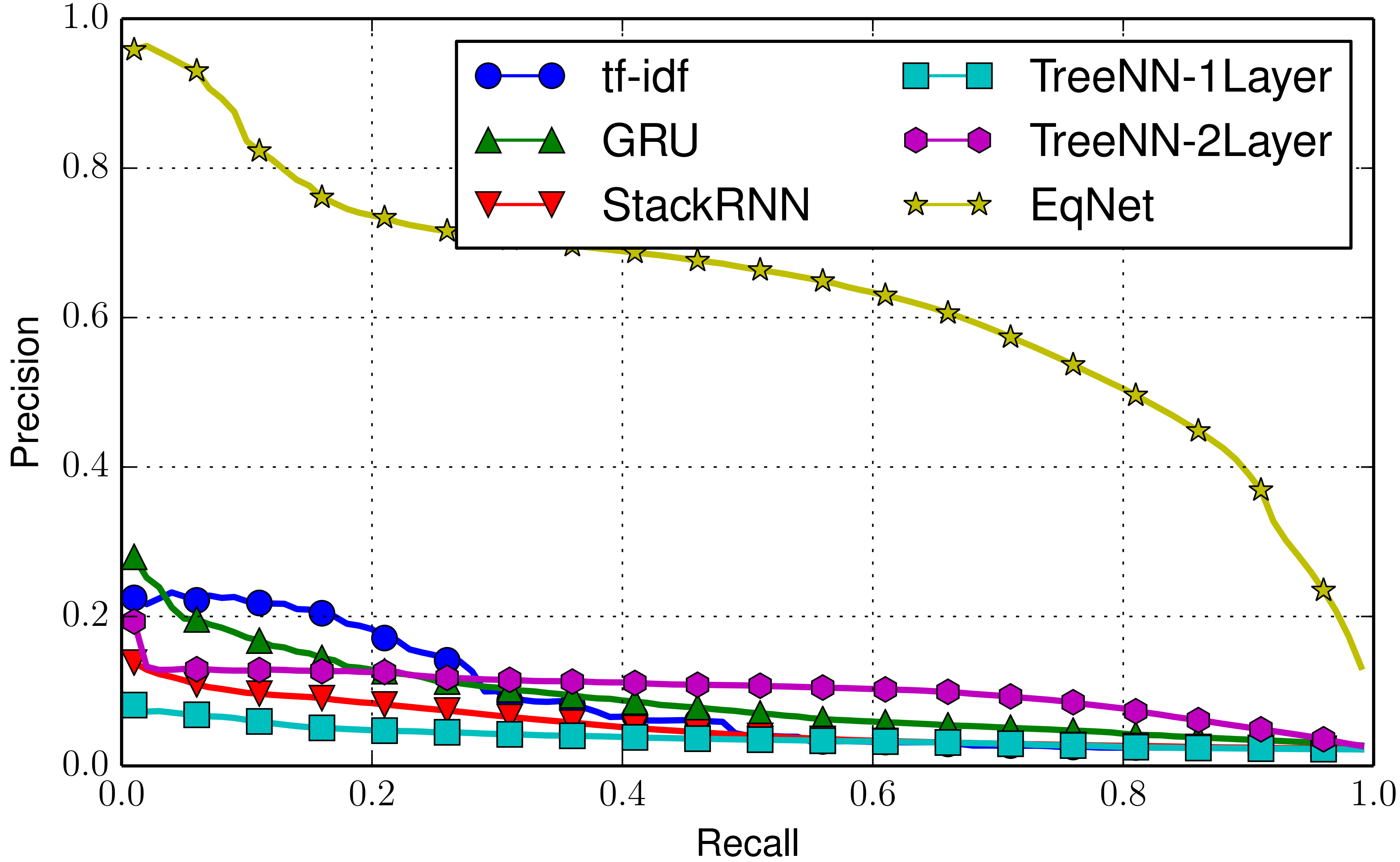


semantic space

Same equivalence class to query
Different equivalence class to query

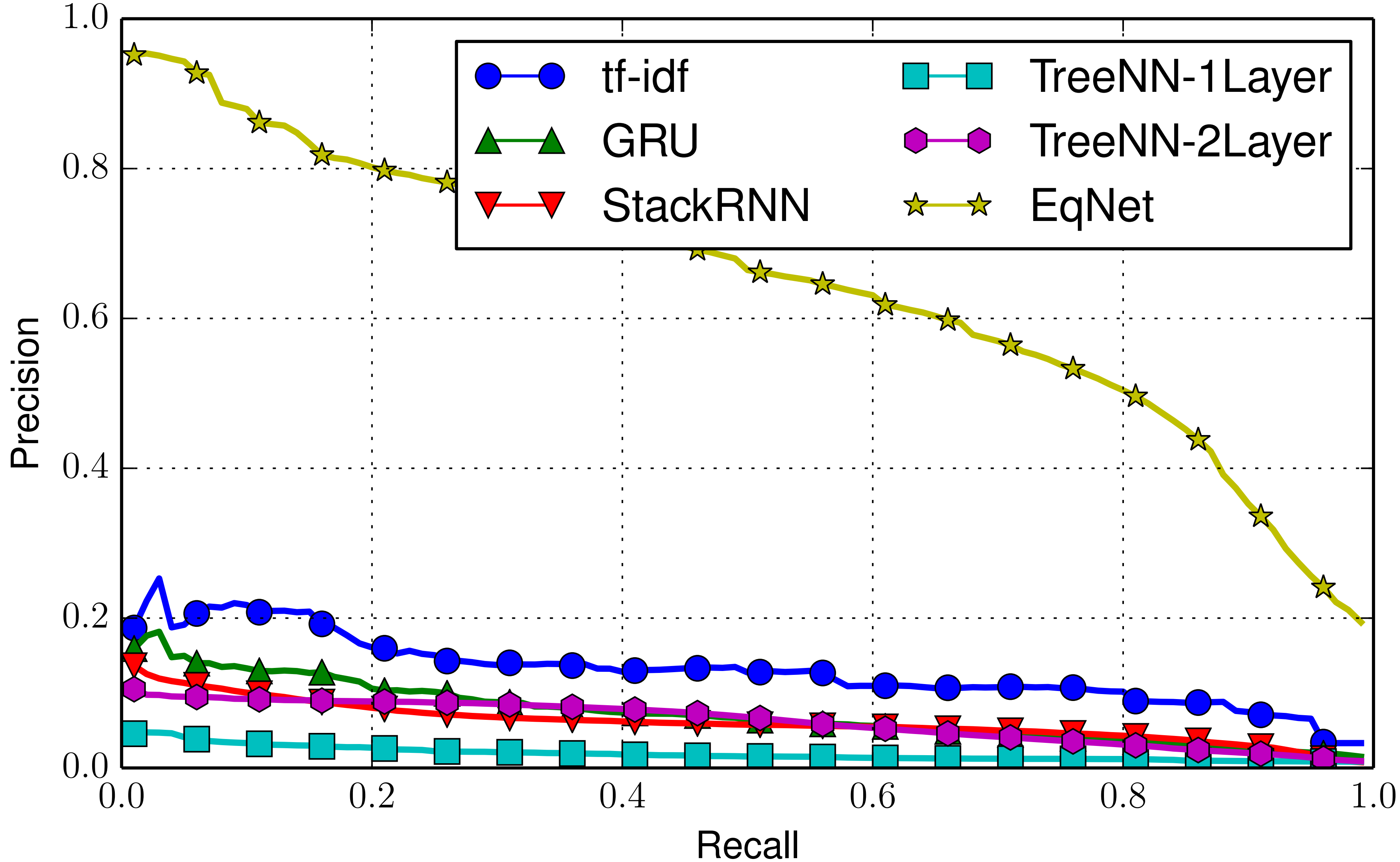
Seen equivalence classes

Equivalent expressions to the queries were in training set



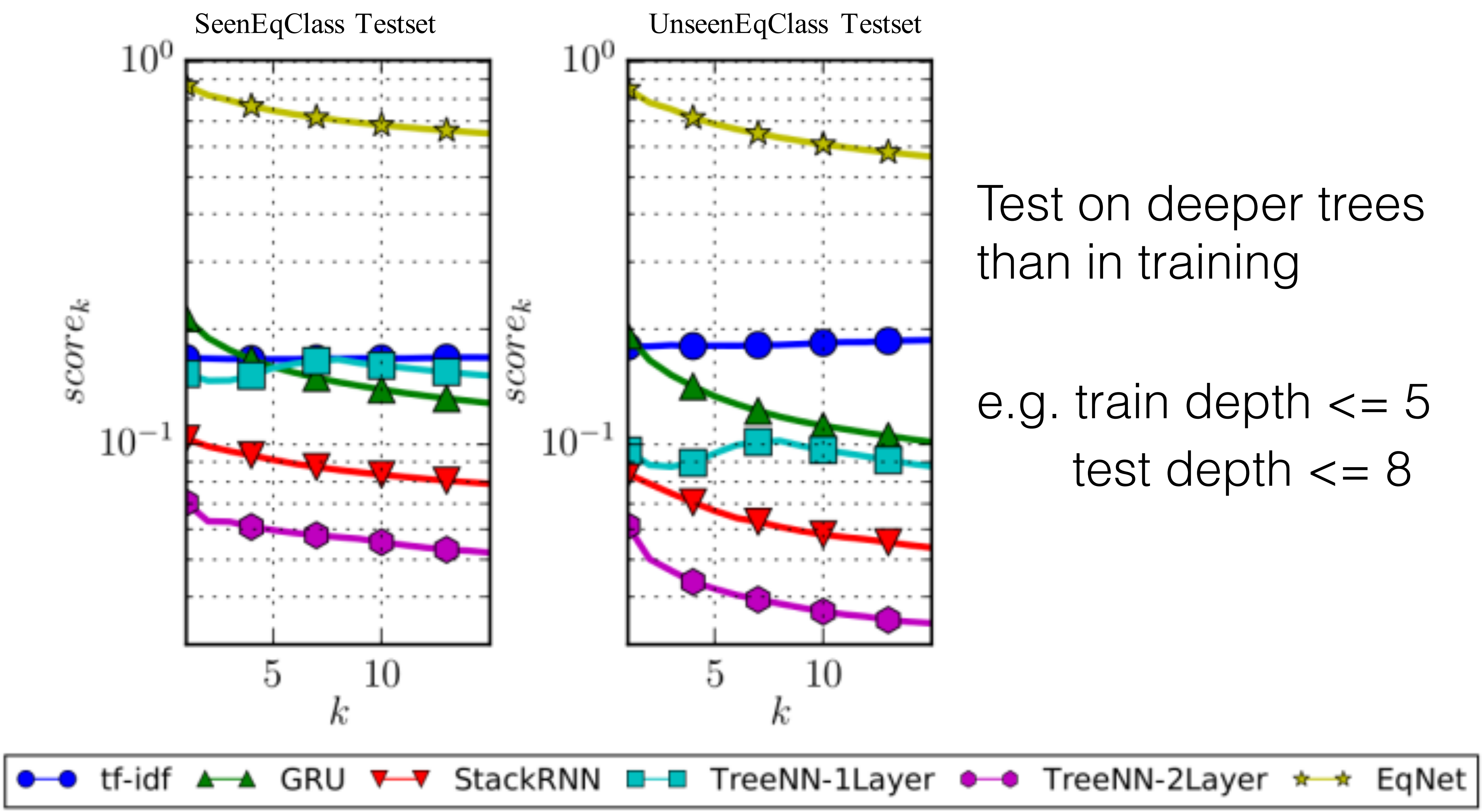
Unseen equivalence classes

Zero shot learning. No training examples of equivalent expressions.



EqNet performance on seen and unseen is similar!

Learned compositionality?



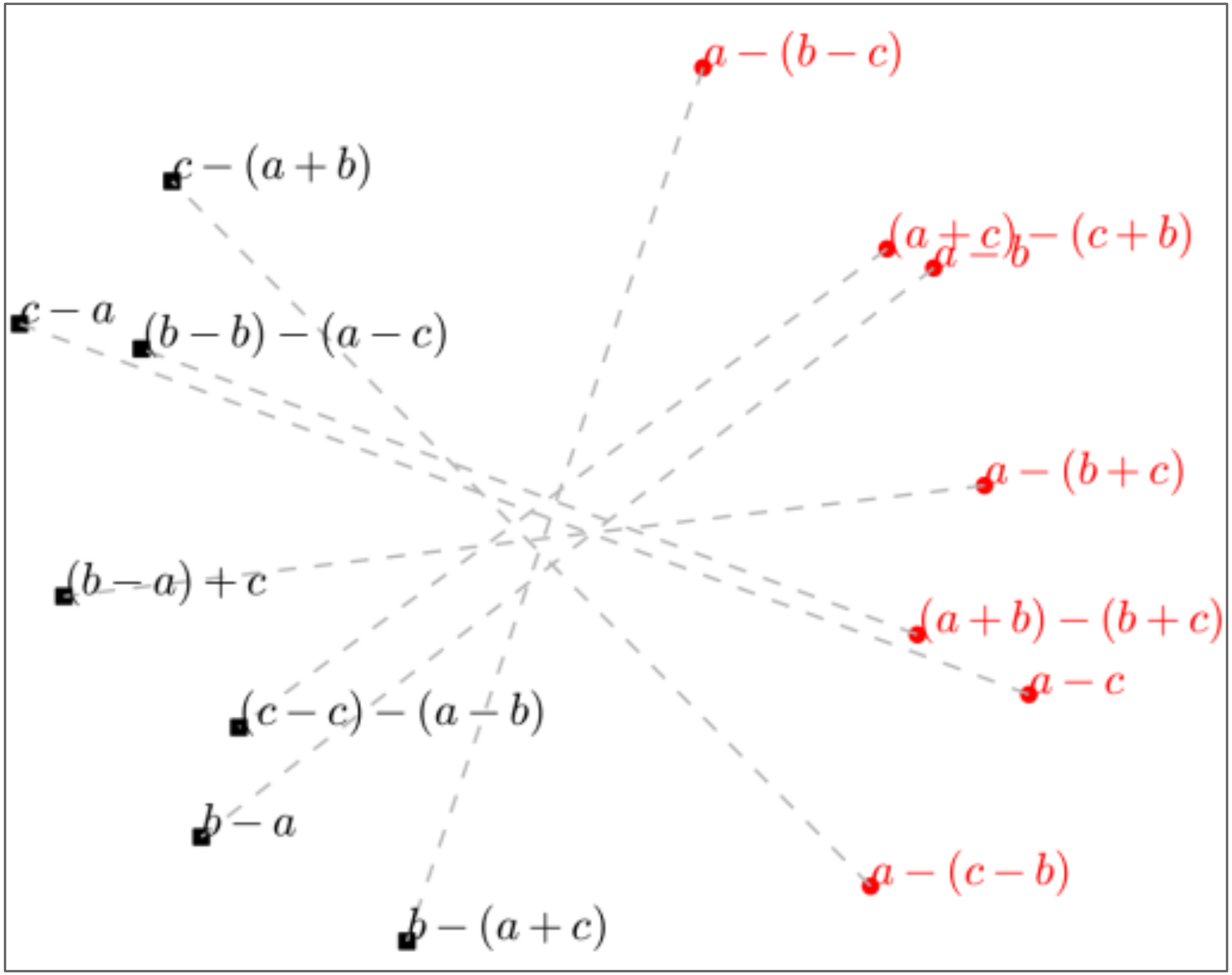
Test on deeper trees than in training

e.g. train depth <= 5

test depth <= 8

Visualizing polynomials

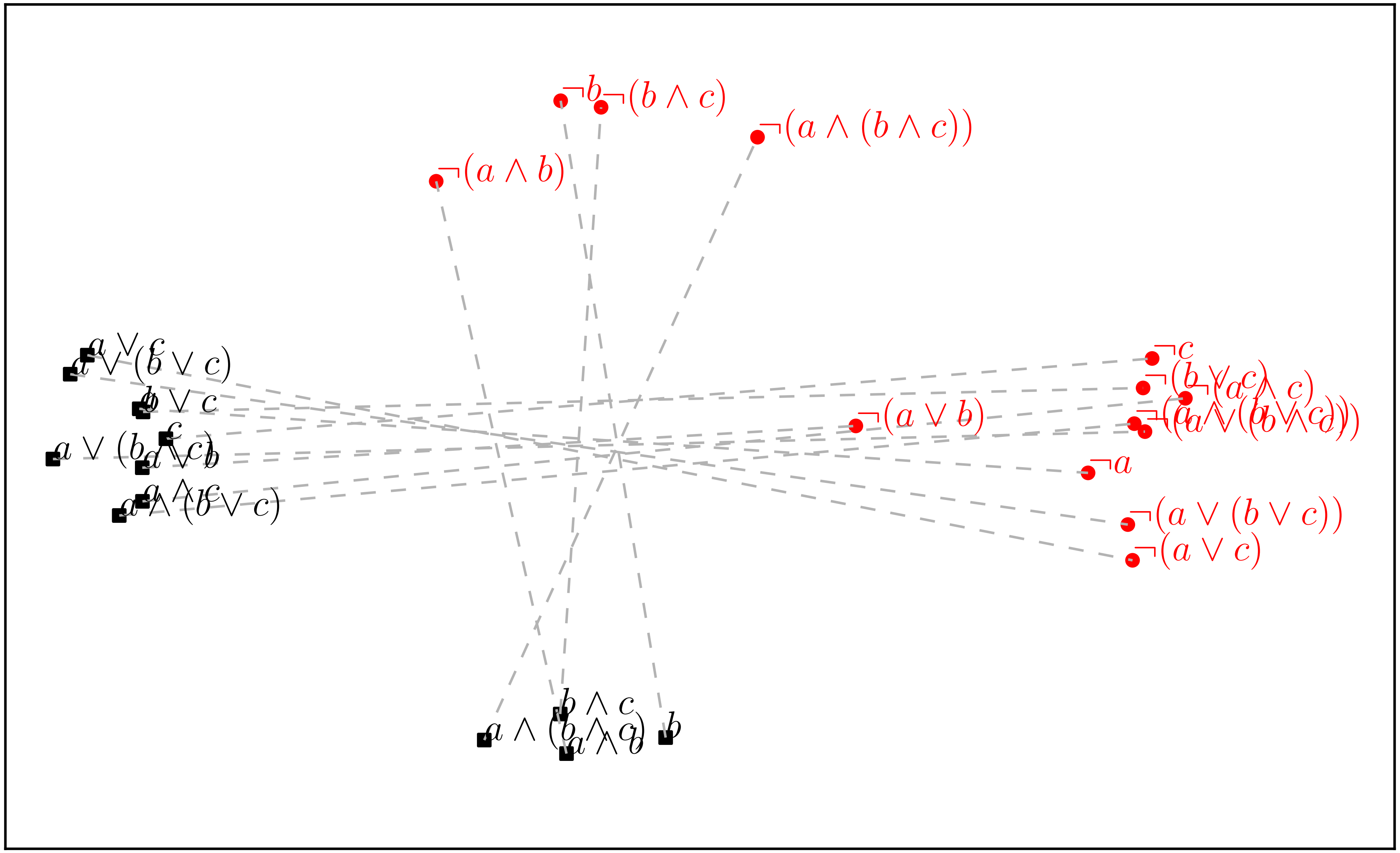
`multivariatePolynomial2vec?`



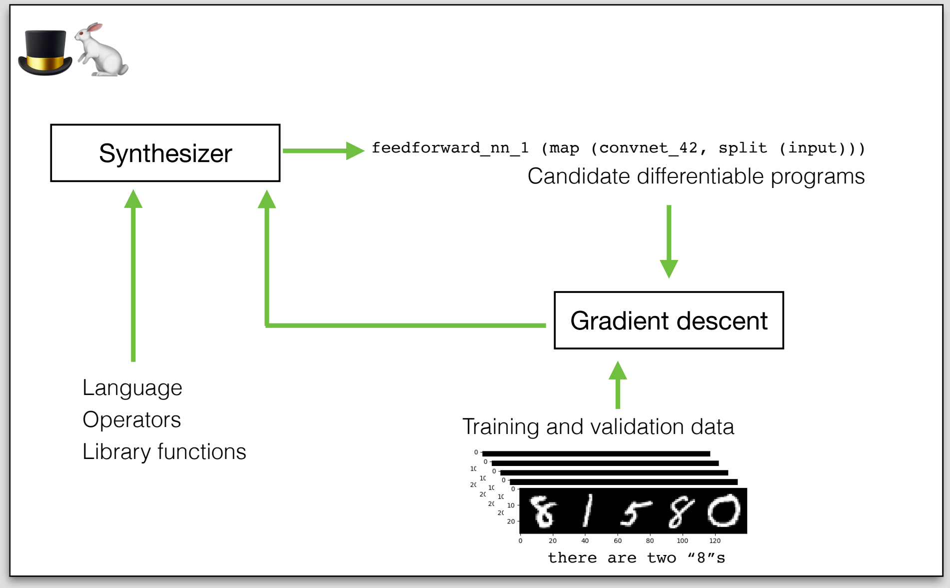
PCA visualization of semVecs

Visualizing boolean expression

`booleanExpression2vec?`



PCA visualization of semVecs

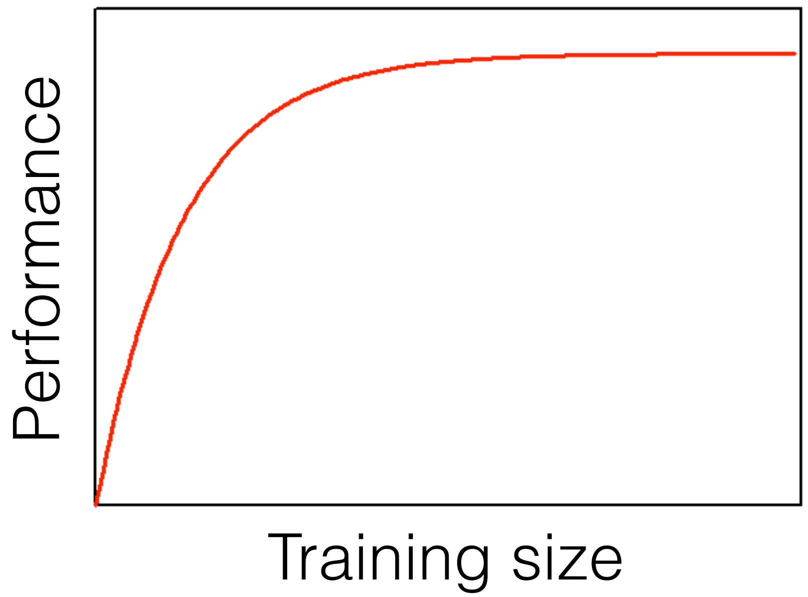


Synthesis of Differentiable Functional Programs for Lifelong Learning

[Valkov, Chaudhari, Srivastava, Sutton, and Chaudhuri, 2018]

bit.ly/adventures-neurosymbolic

The Problem with Learning



All learning curves stop.

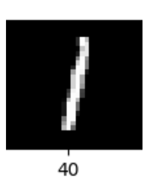
Why “data hungriness”?

Learning systems never get better at learning itself

Lifelong learning *[Thrun & Mitchell, 1995; Carlson et al, 2010]*

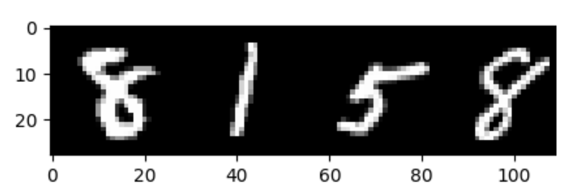
On series of tasks, accelerate learning by re-using learned structure

Task 1
Recognition



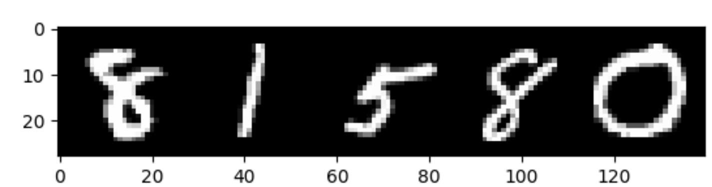
`that's a “one”`

Task 2
Counting



`I count two “8's”`

Task 3
Arithmetic



`The sum is 23`

Neural libraries

[Gaunt, Brockschmidt, Kushman, Tarlow ICML 2017]

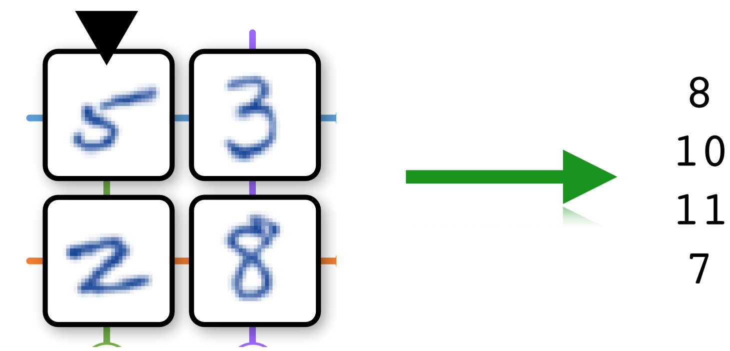
Digit Recognizer (ConvNet)

Toy Recognizer (ConvNet)

Street Sign Recognizer (ConvNet)

Networks from old tasks

New task



Controller in differentiable PL

Differentiable interpreter

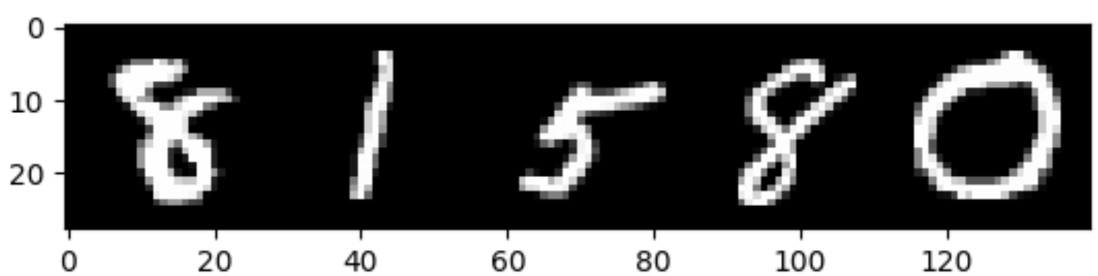
Learn programs plus perceptual networks

```
# initialization:
R0 = READ
# program:
R1 = MOVE_EAST
R2 = MOVE_SOUTH
R3 = SUM(R0, R1)
R4 = NOOP
return R3
```

end-to-end gradient descent

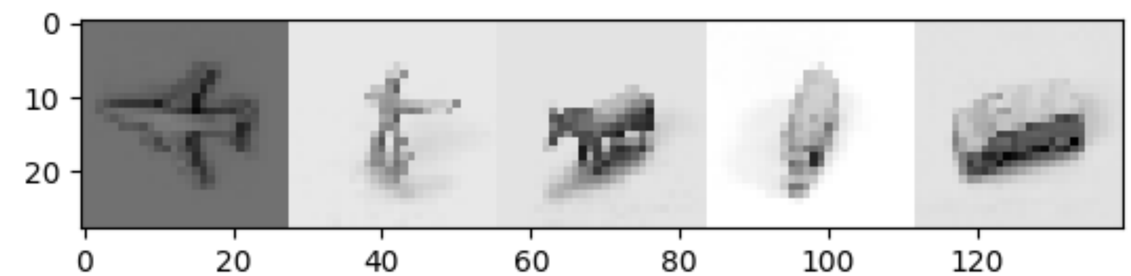
High level transfer

Task 1
Count digits

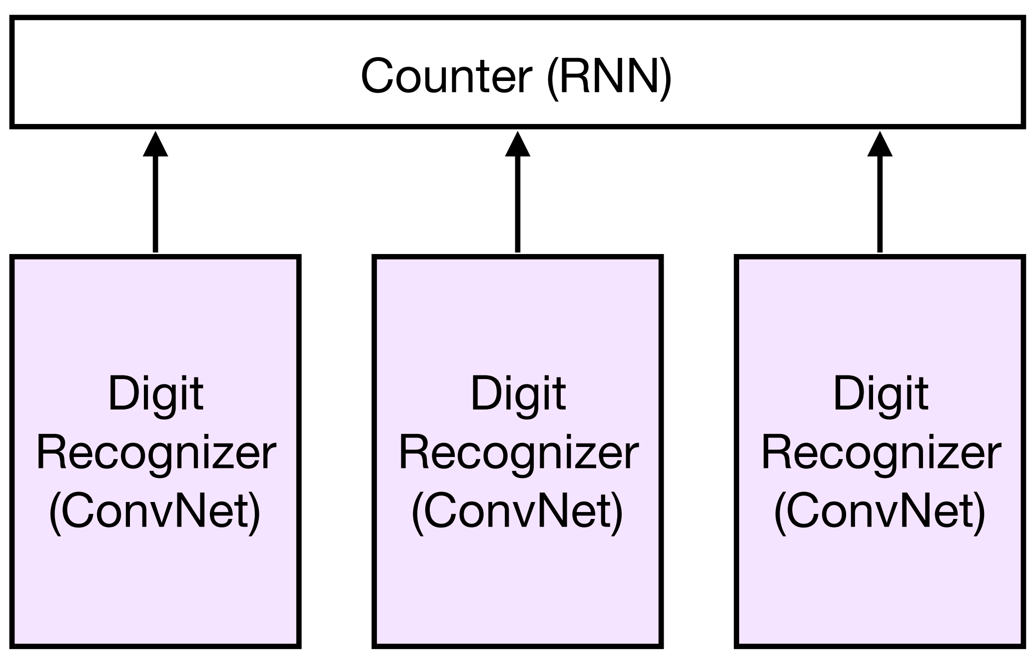


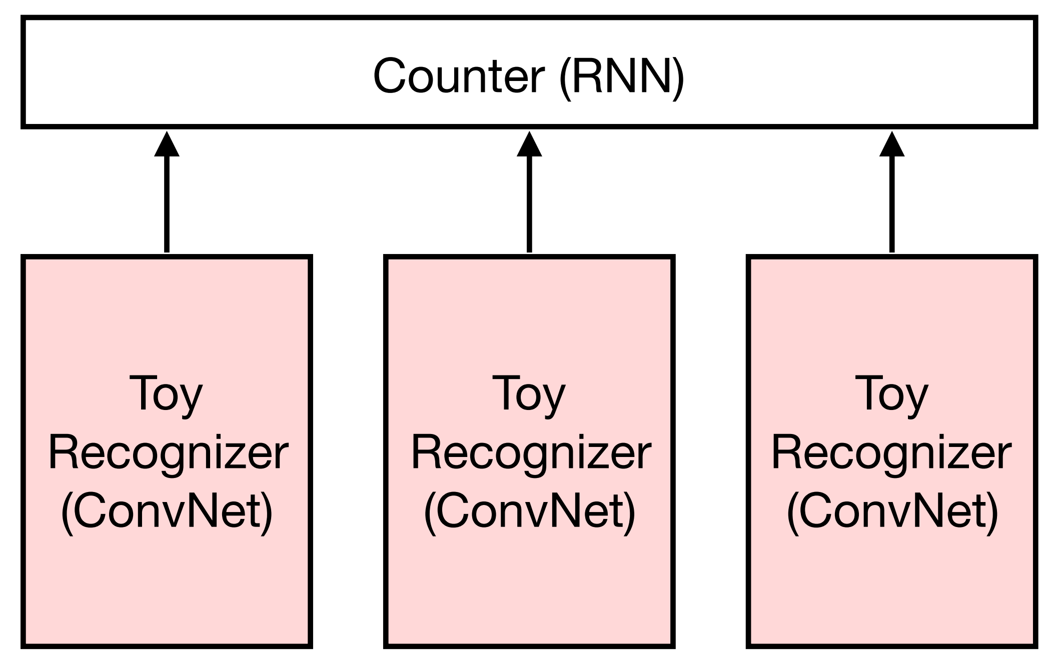
`there are two "8"s`

Task 2
Count toys



`there is one "toy airplane"`
`(and why don't I have two?)`

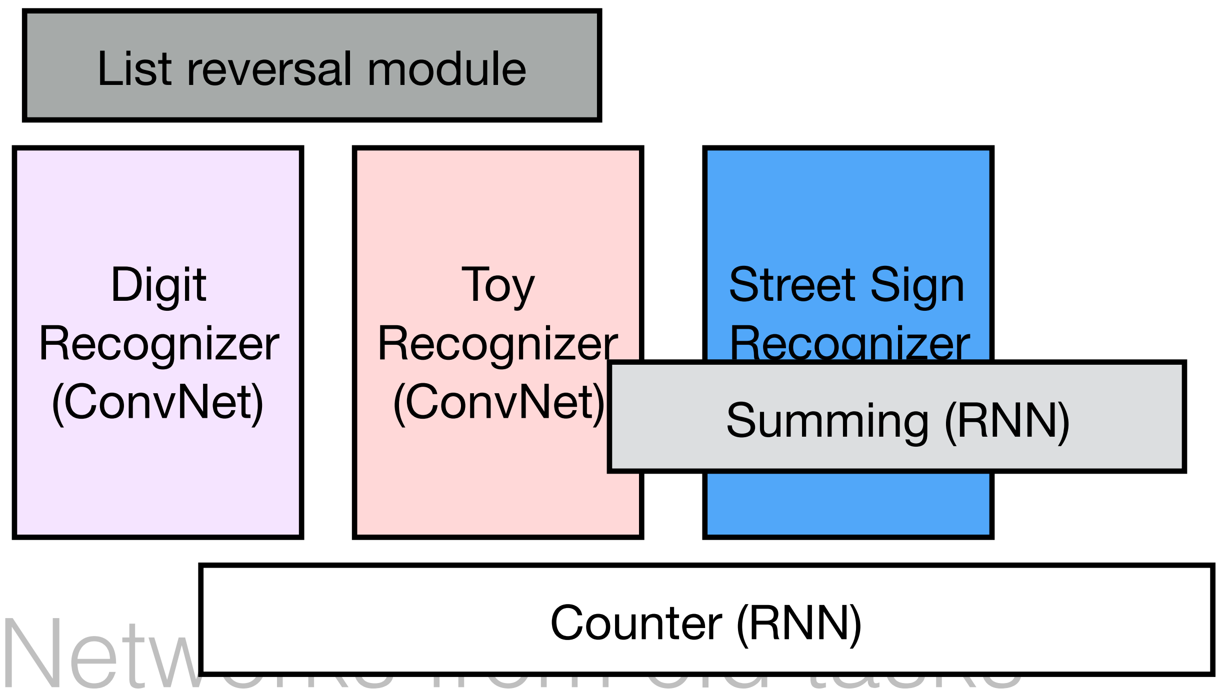




Reusing early layers not sufficient!

[Hinton & Salakhutdinov, 2006; Rusu et al 2016]

High-level neural libraries



New task

“High-level modules”

Other networks as input

Problem: How to combine?

Controller in differentiable PL

Differentiable interpreter

Learn programs plus perceptual networks

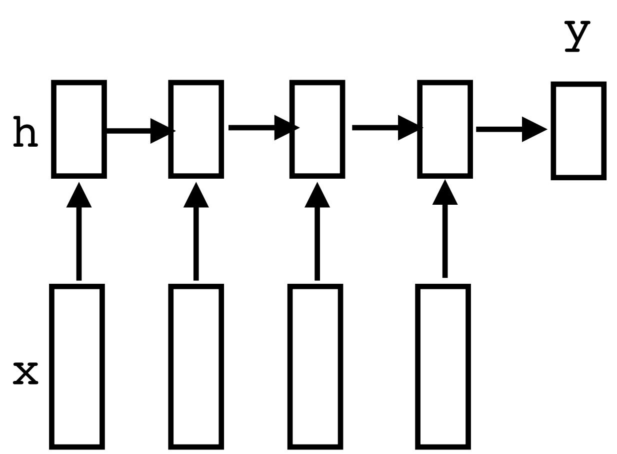
```
# initialization:
R0 = READ
# program:
R1 = MOVE_EAST
R2 = MOVE_SOUTH
R3 = SUM(R0, R1)
R4 = NOOP
return R3
```

end-to-end gradient descent

100 neural architectures, 1 weird trick

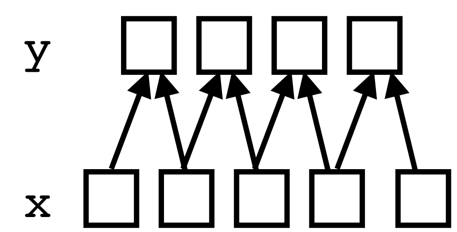
Functional programming

Recurrent neural network



`fold (rnn_1step)`

1-layer ConvNet




```
map (nn_kernel)
  o zipWithSelf
```

```
conv (nn_kernel_0)
```

multiple filters? change this

Combinators preserve differentiability

Often point-free
[Backus, 1978]

Deep feedforward net `relu o W_n o relu o … o relu o W_1`

Deep ConvNet `conv (nn_kernel_0) o conv (nn_kernel_1) o … o conv (nn_kernel_D)`

Attention mechanism `softmax o map(attn_network) o fold(rnn_1step)`

Graph convolutions `gconv (nn_kernel_0)`

Olah: http://colah.github.io/posts/2015-09-NN-Types-FP/

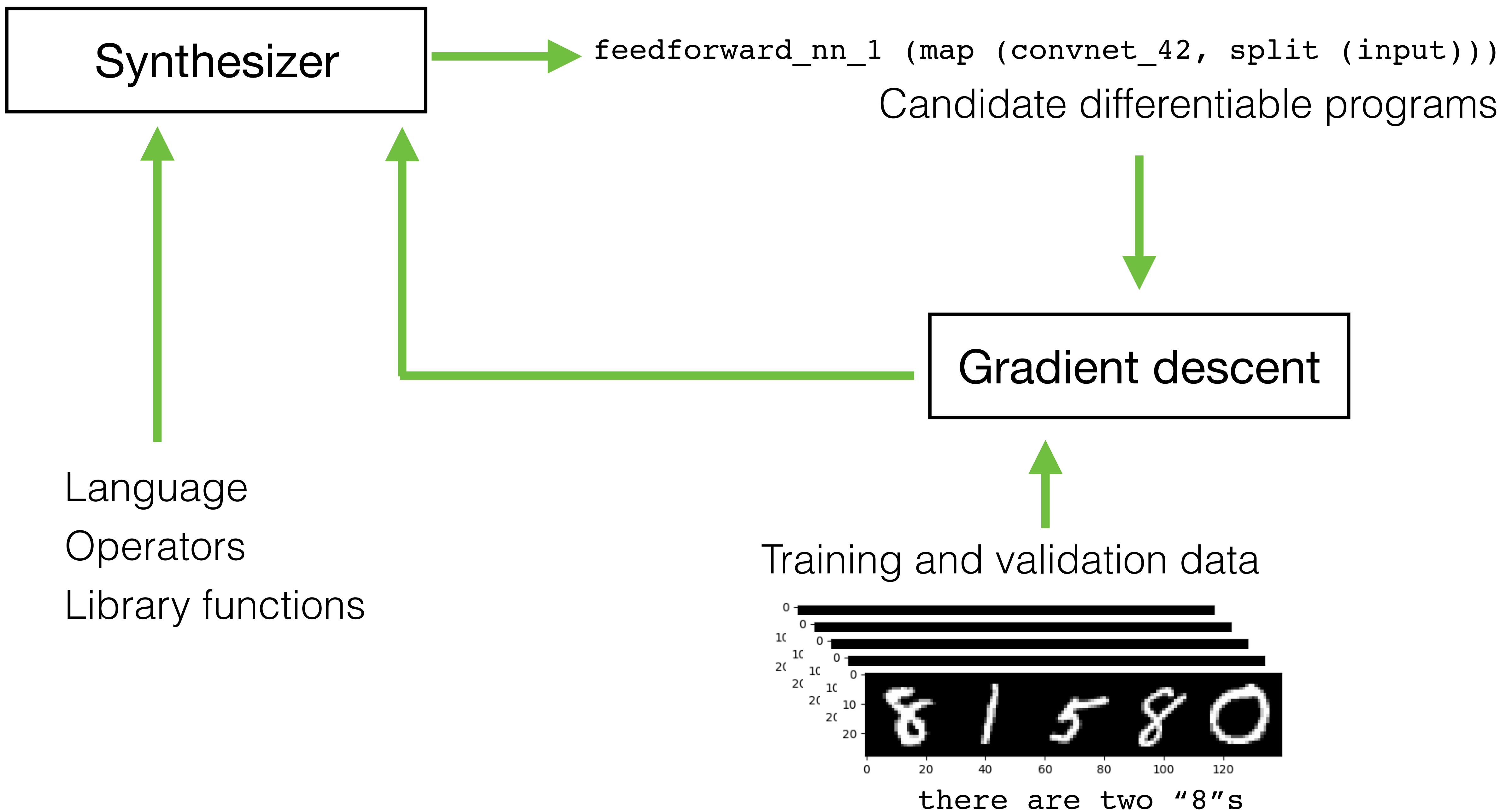
Main ideas

- Neural libraries for high level transfer
- Functional programs represent deep architecture
- **Symbolic program synthesis** to choose:
    - Which neural library functions for re-use
    - Architecture that puts them together

Houdini 🎩🐇

Houdini

Synthesis of Differentiable Functional Programs



HOUDINI: Heuristic Optimization for the Ultimate Development of Integrated Neurosymbolic Intelligence

Language

Types

guides the search space

$$
\begin{array}{lcl}
\tau & ::= & Atom \mid ADT \mid F \\
Atom & ::= & \texttt{bool} \mid \texttt{real} \\
TT & ::= & Atom \mid \texttt{Tensor}\langle Atom \rangle [m_1][m_2]\ldots[m_k] \\
ADT & ::= & TT \mid \alpha\langle TT \rangle \\
F & ::= & ADT \mid F_1 \to F_2.
\end{array}
$$

Programs

defines the search space

$$
e \quad ::= \quad \langle\!\langle \tau \rangle\!\rangle \mid \oplus_w \mid e_0 \circ e_1 \mid \mathbf{map}_\alpha\ e \mid \mathbf{fold}_\alpha\ e \mid \mathbf{conv}_\alpha\ e.
$$

list and graph versions

“neural functions”, e.g., two-layer ff network

Search

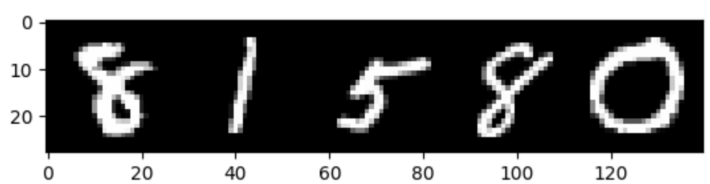
Enumeration of programs in language

Shortest to longest

Types reduce search space

Limit number of trainable functions per candidate

Training set



→ 2

Search

`<< Tensor[28,148] –> real  >>`

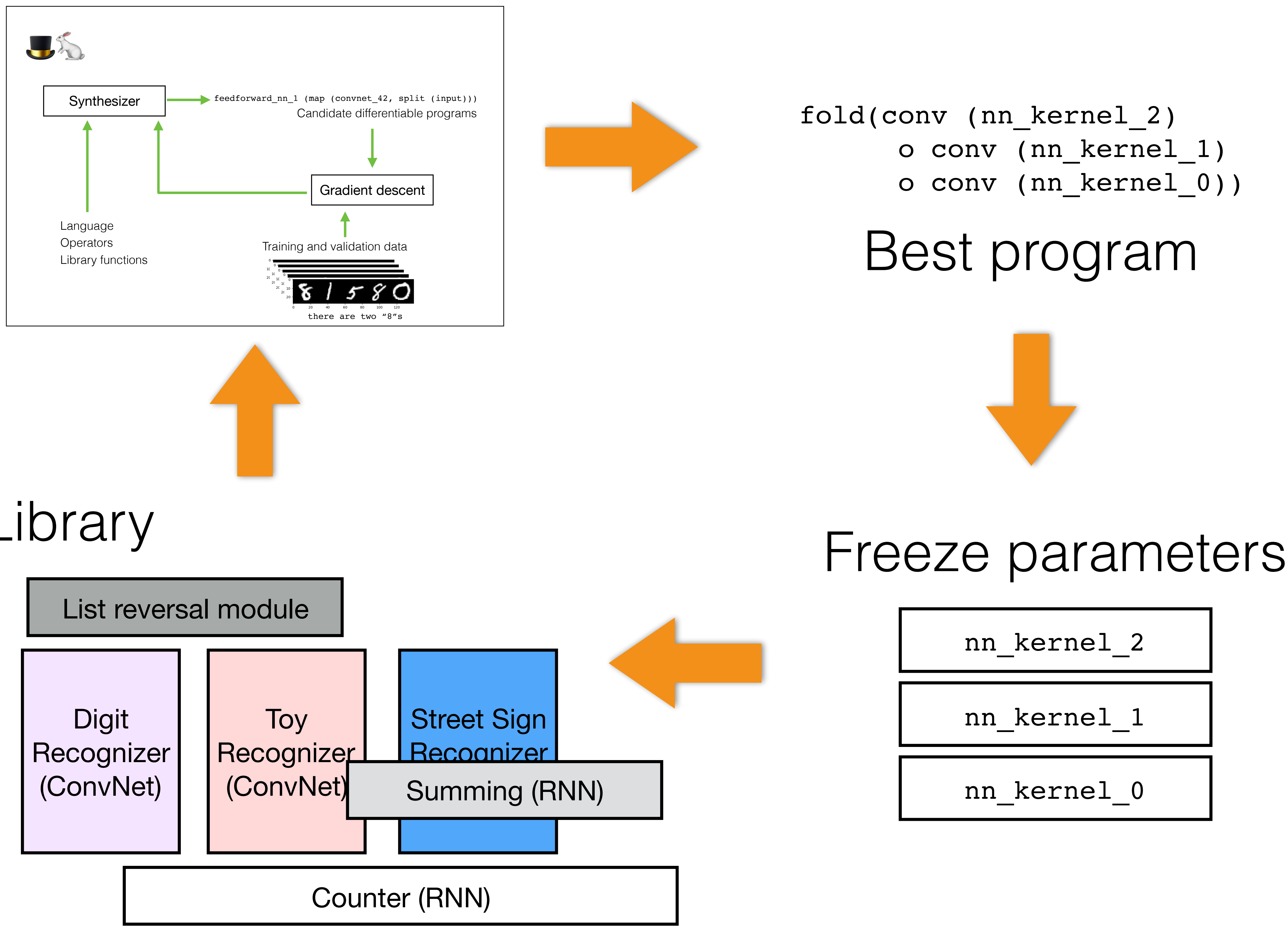
Partial program with type

Expand using grammar

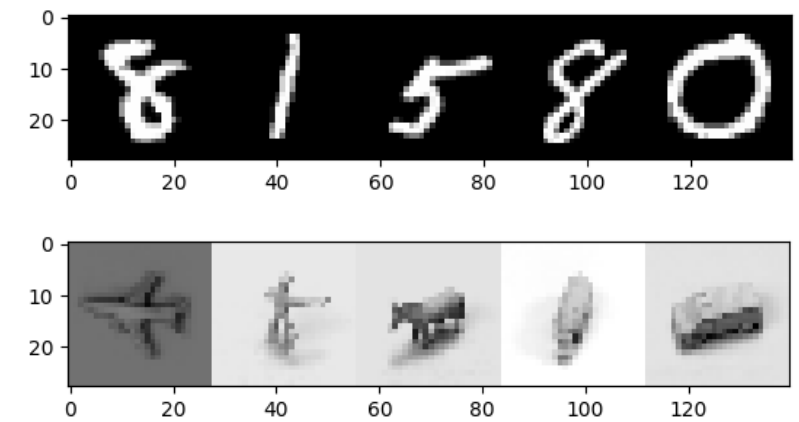
`map`

`fold[list] << Tensor[28,148] –> list[T]  >>`

Synthesis for Lifelong Learning



Experiments



Counting

MNIST digits:
Recognize(5); Recognize(8); Count(8)

MNIST digits:
Recognize(5), Count(5); Count(8); Recognize(5)

MNIST/NORB:
Recognize(5), count(5), count(toy airplane), recognize(toy airplane)

Summing

Classify(1…10), Sum(sequence)

All-pairs shortest path

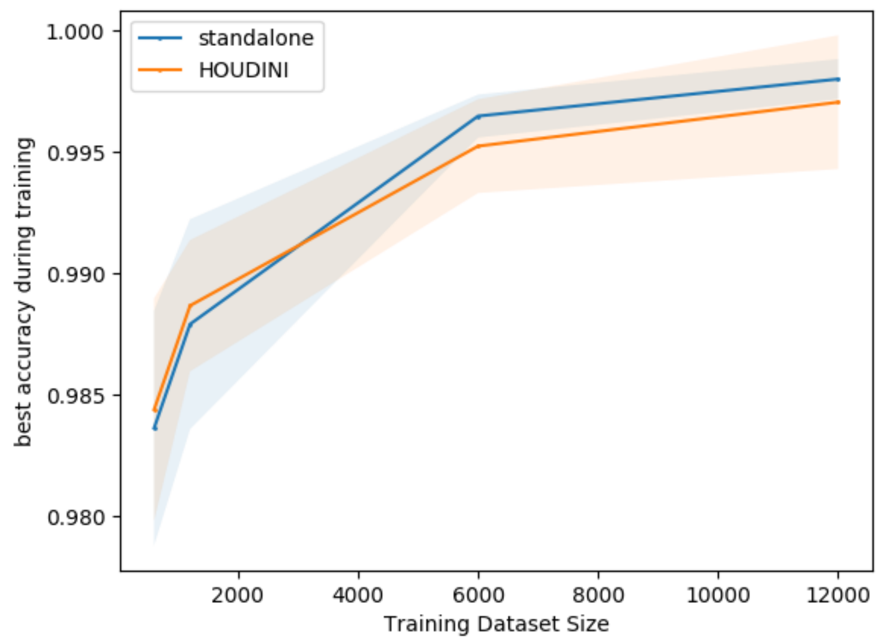
$$(\mathbf{conv}^{i}_{\texttt{graph}\langle\texttt{Tensor}[2]\rangle}\ nn\_relax) \circ (\mathbf{map}_{\texttt{graph}\langle\texttt{Tensor}[32][32][3]\rangle}\ perceive)$$

Classify(sign), Shortest_path(sign)

Classify(mnist), Shortest_path(mnist), Shortest_path(mnist)

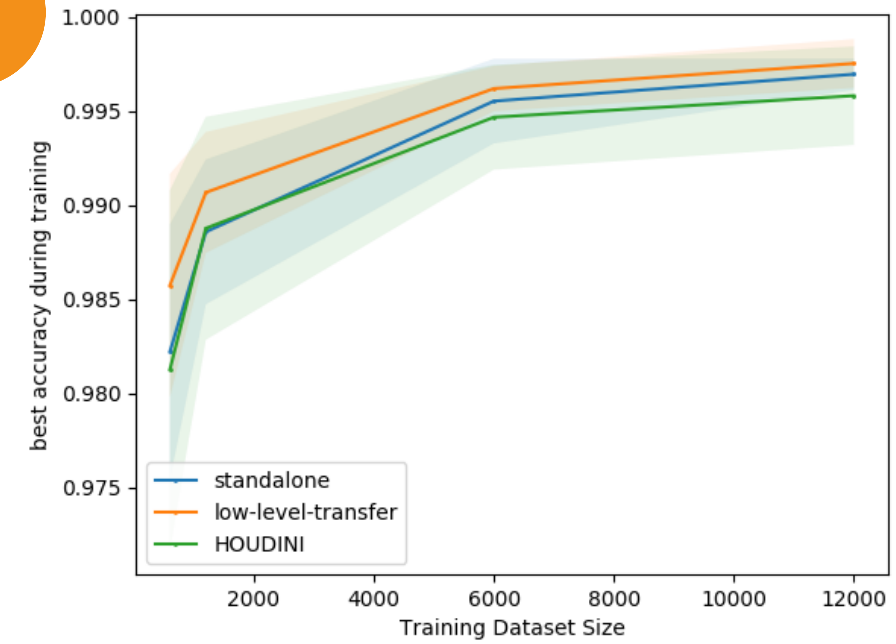
Results: Learning to Count

1



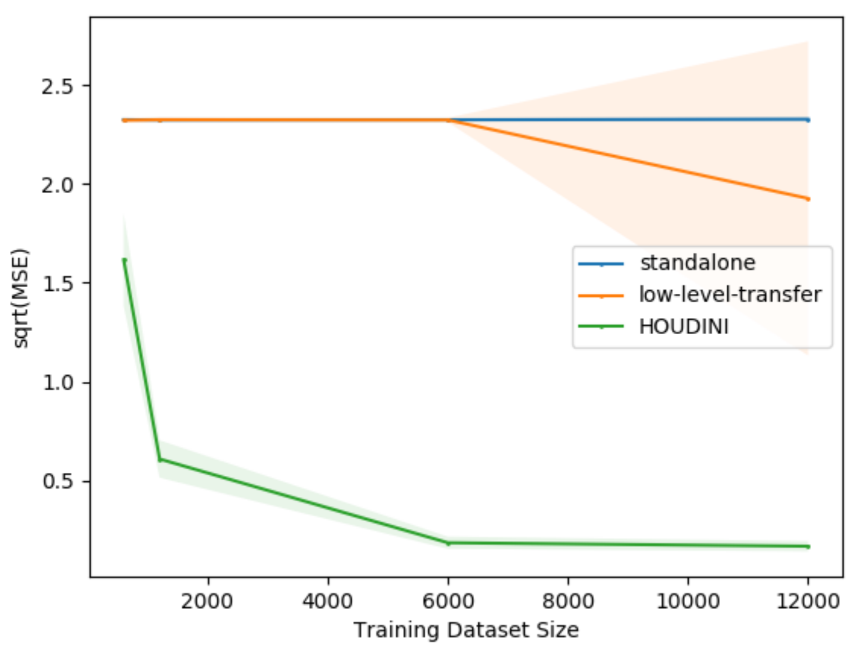
Recognise digit d_1
(binary classification)

2



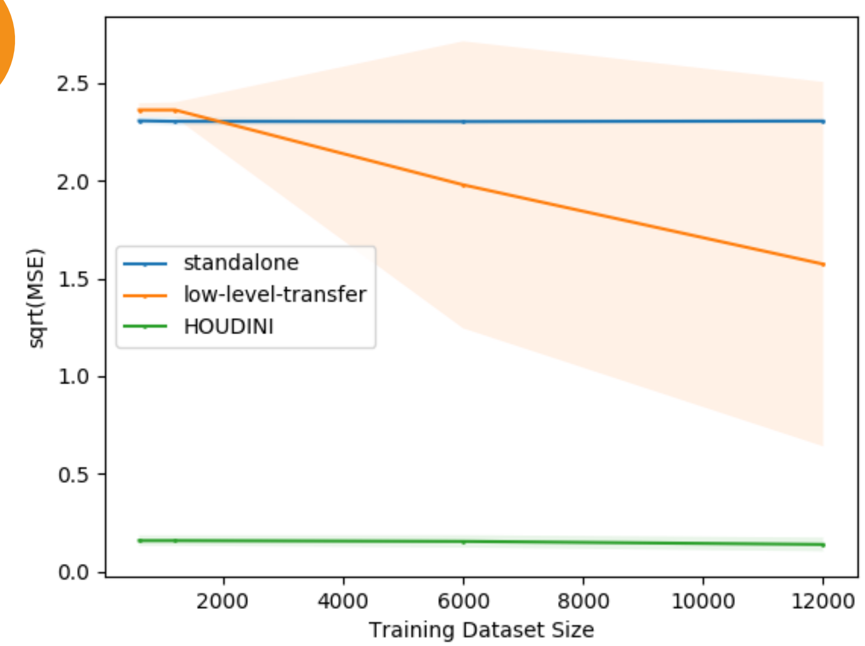
Recognise digit d_2
(binary classification)

3



Count digits d_1

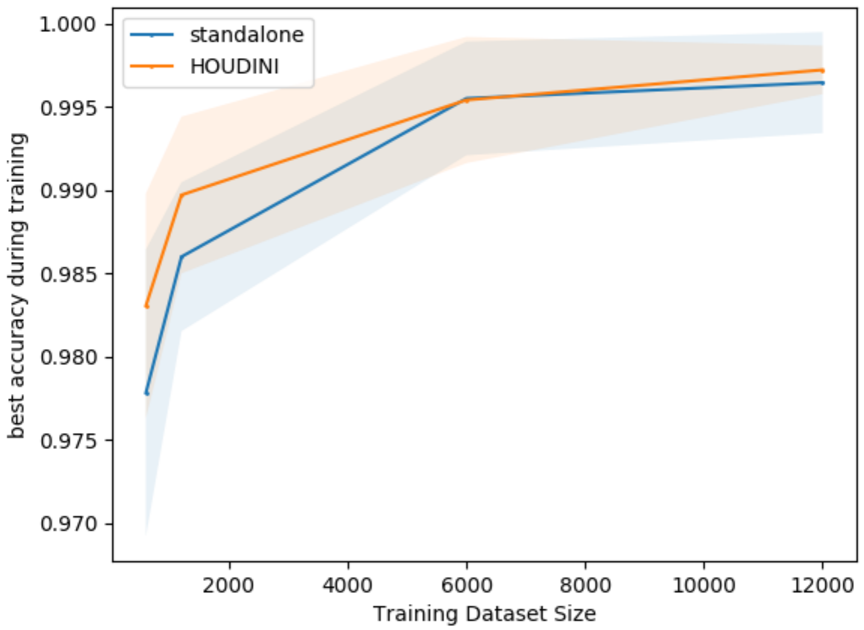
4



Count digits d_2

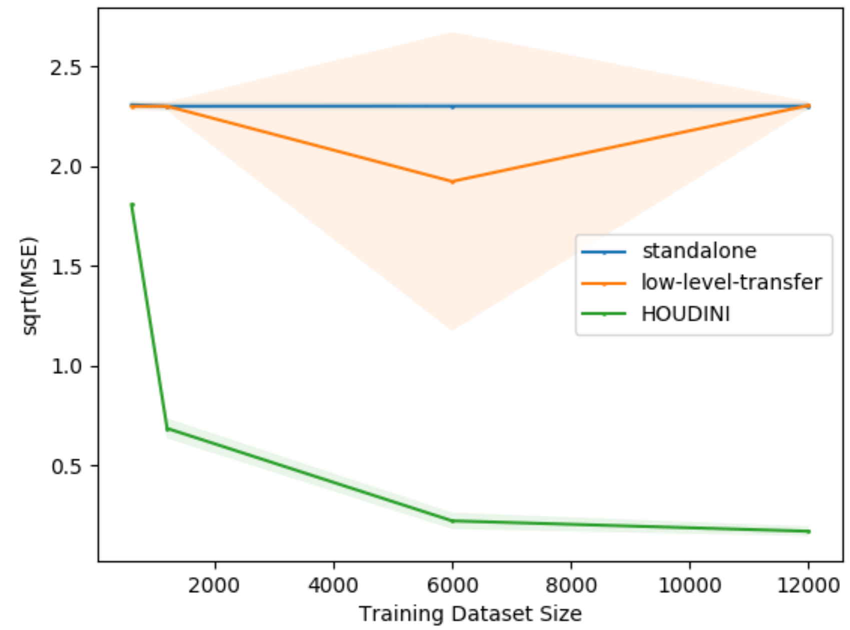
Results: Counting Toys

1



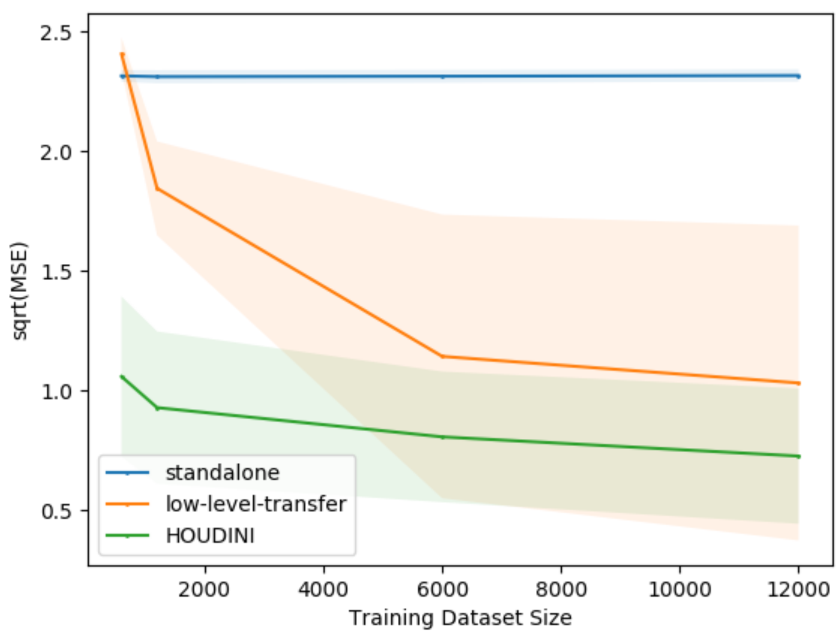
Recognise digit d_1

2



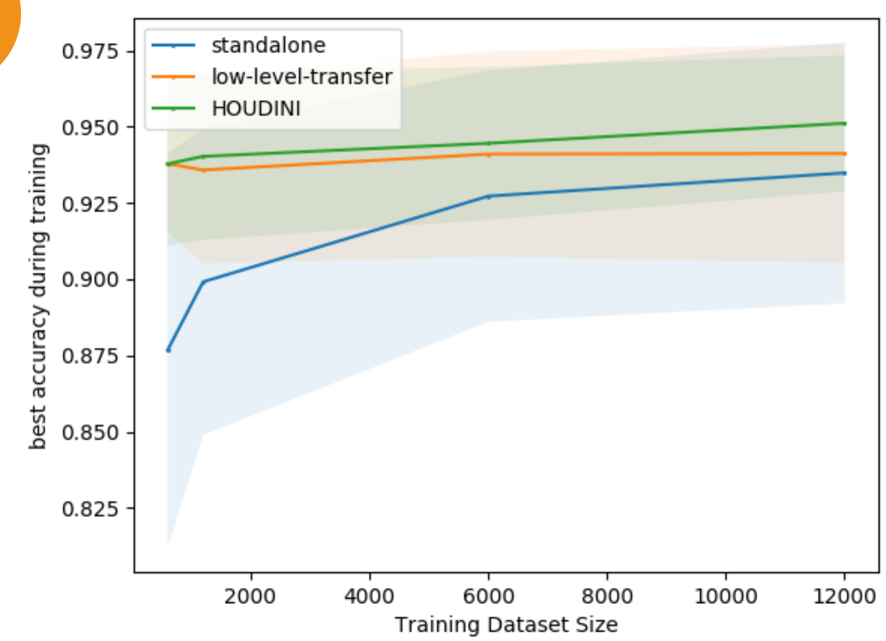
Count digit d_1

3



Count toys t_1

4



Recognise toy t_1

Results: Shortest path

Training: Grids of size 2x2, 3x3, 4x4

Testing: Grids of size 5x5

	Image —> Node cost (RMSE)	Shortest path length (RMSE)
Vanilla CNN—>LSTM	0.37	5.97
Houdini	0.38	**1.53**
	Street sign images	*Street sign images*

Results: Shortest path (transfer)

Training: Grids of size 2x2, 3x3, 4x4

Testing: Grids of size 5x5

	Image —> Node cost (RMSE)	Shortest path length (RMSE)	Shortest path length (RMSE)
Vanilla CNN—>LSTM	1.21	5.33	6.16
Houdini	1.29	**1.62**	**4.98**
	MNIST images	*MNIST images*	*Street sign images*

Impact of type system

Program depth	4	5	6
No type system	15633	247589	3449845
Type system	25	155	444

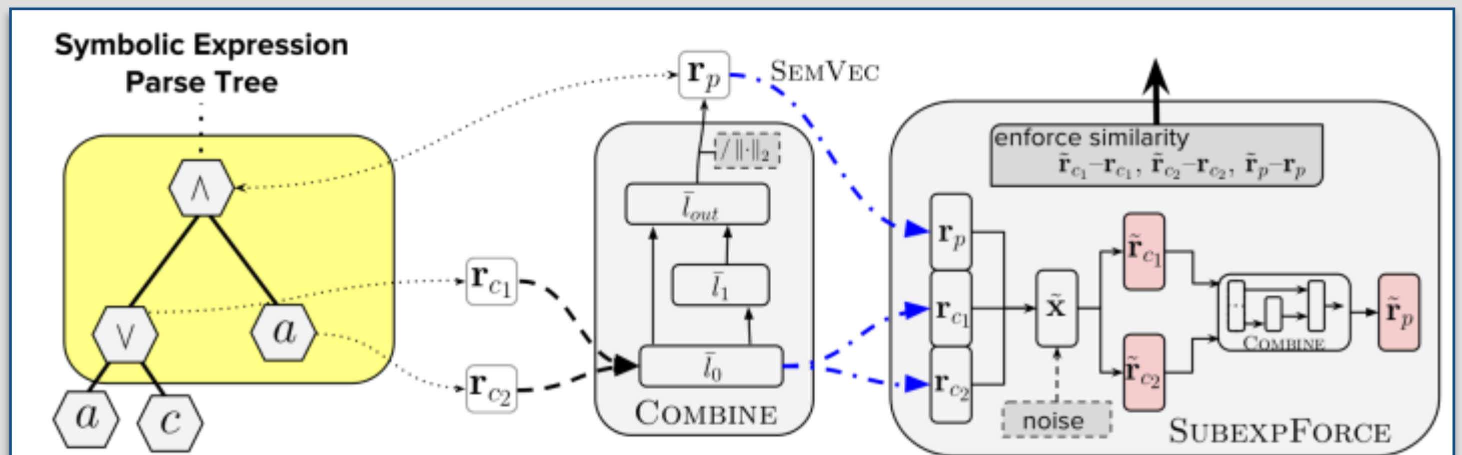
Types of transfer

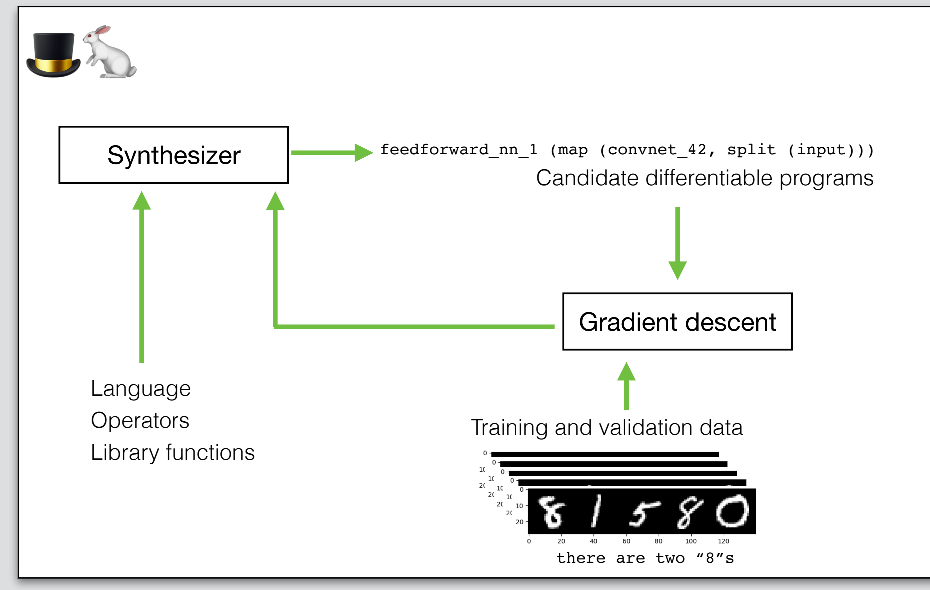
- Low-level transfer
  - Reuse perceptual network across high-level tasks
- High-level transfer
- “Infilling”
  - Learn perceptual concepts given only supervision at high level
- Selective transfer
  - Synthesis algorithm decides whether and when to re-use

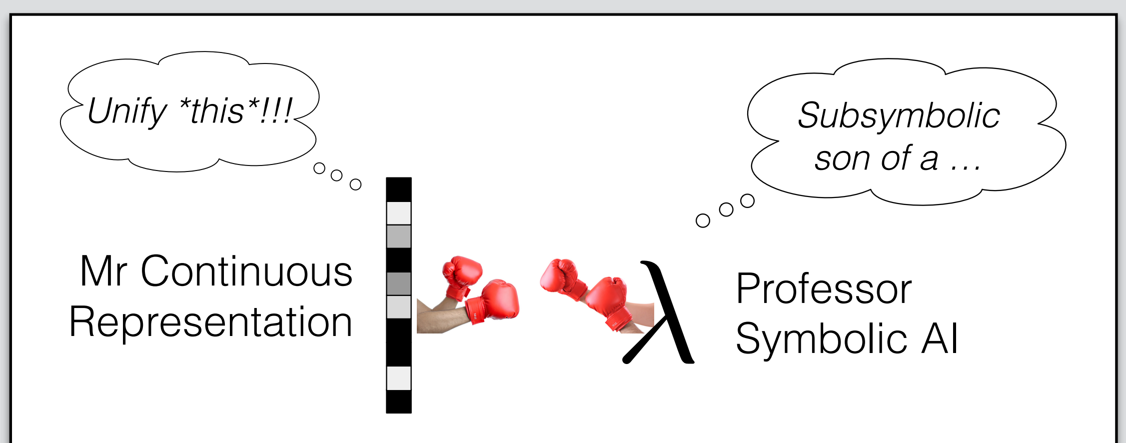
Adventures in Neurosymbolic Learning

Charles Sutton

bit.ly/adventures-neurosymbolic







Thanks!

- Miltos Allamanis
- Pushmeet Kohli
- Pankajan Chanthirasegaran
- Kai Xu
- Lazar Valkov
- Dipak Chaudhari
- Akash Srivastava
- Swarat Chaudhuri

Microsoft Research
The Alan Turing Institute
EPSRC Engineering and Physical Sciences Research Council
The University of Edinburgh informatics