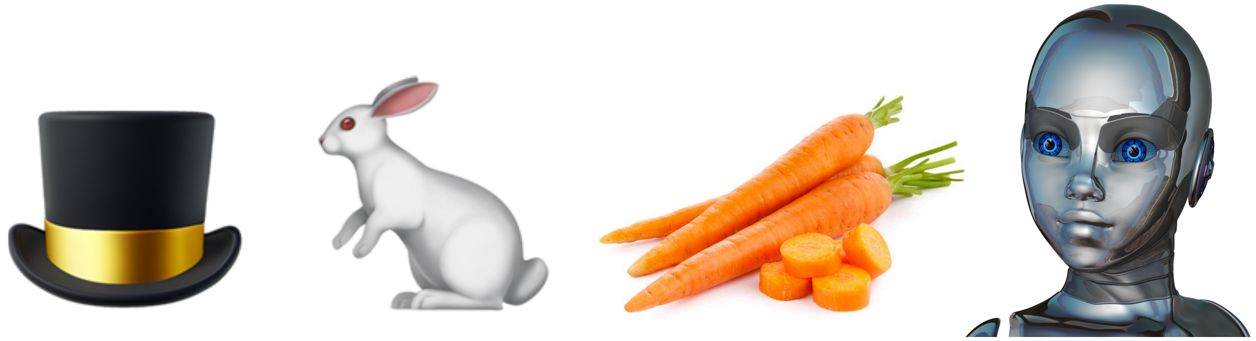


Deep Learning, Language, and Code:

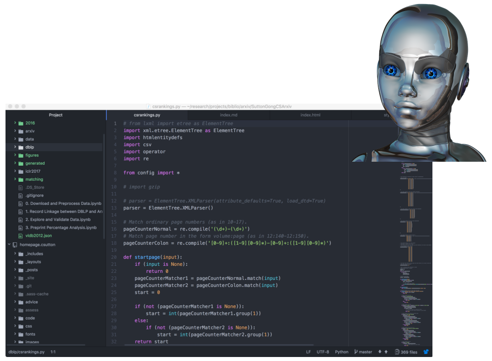
From Methodology to Applications and Back

Charles Sutton
University of Edinburgh
& The Alan Turing Institute
& Google Brain

http://bit.ly/sutton-dllc

THE UNIVERSITY of EDINBURGH informatics
The Alan Turing Institute
Microsoft Research
EPSRC Engineering and Physical Sciences Research Council

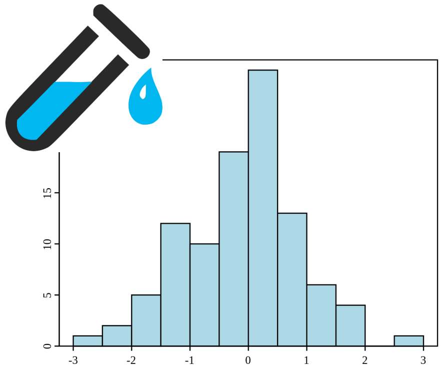
Applications ⟷ Methodology



```
# from lxml import etree as ElementTree
import xml.etree.ElementTree as ElementTree
import htmlentitydefs
import csv
import operator
import re

from config import *

# import gzip

# parser = ElementTree.XMLParser(attribute_defaults=True, load_dtd=True)
parser = ElementTree.XMLParser()

# Match ordinary page numbers (as in 10-17).
pageCounterNormal = re.compile('(\d+)-(\d+)')
# Match page number in the form volume:page (as in 12:140-12:150).
pageCounterColon = re.compile('[0-9]+:([1-9][0-9]*)-[0-9]+:([1-9][0-9]*)')

def startpage(input):
    if (input is None):
        return 0
    pageCounterMatcher1 = pageCounterNormal.match(input)
    pageCounterMatcher2 = pageCounterColon.match(input)
    start = 0

    if (not (pageCounterMatcher1 is None)):
        start = int(pageCounterMatcher1.group(1))
    else:
        if (not (pageCounterMatcher2 is None)):
            start = int(pageCounterMatcher2.group(1))
    return start
```

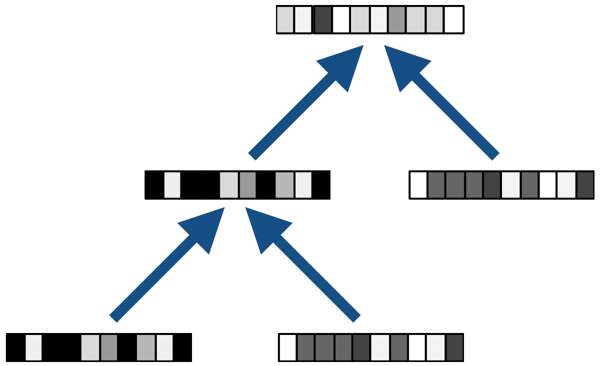
Intelligent tools for software development

[Allamanis, et al, MSR 2012; FSE 2014; ACM CSUR 2018]



Accelerating practical data science

[Sutton, Hobson, Geddes, Caruana 2018]

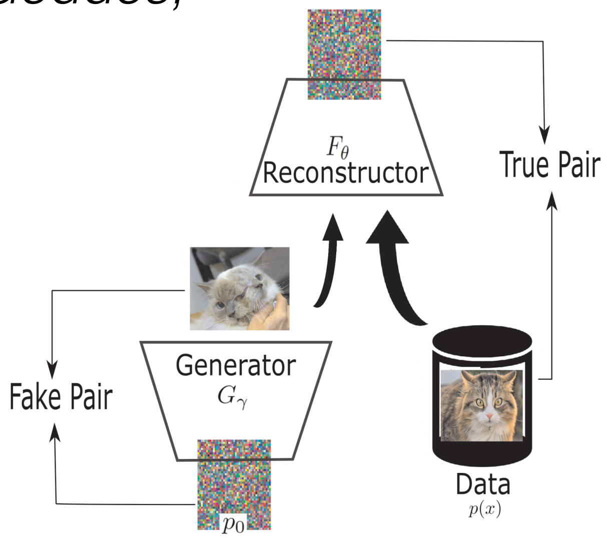


Equivalence networks

[Allamanis et al, ICML 2017]

Household energy usage

[Zhang, Zhong, Goddard, Sutton, AAAI 2018; Zhong, Goddard, Sutton NIPS 2014, NIPS 2015]

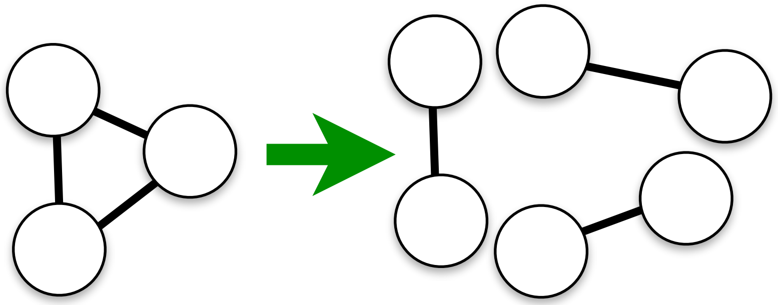


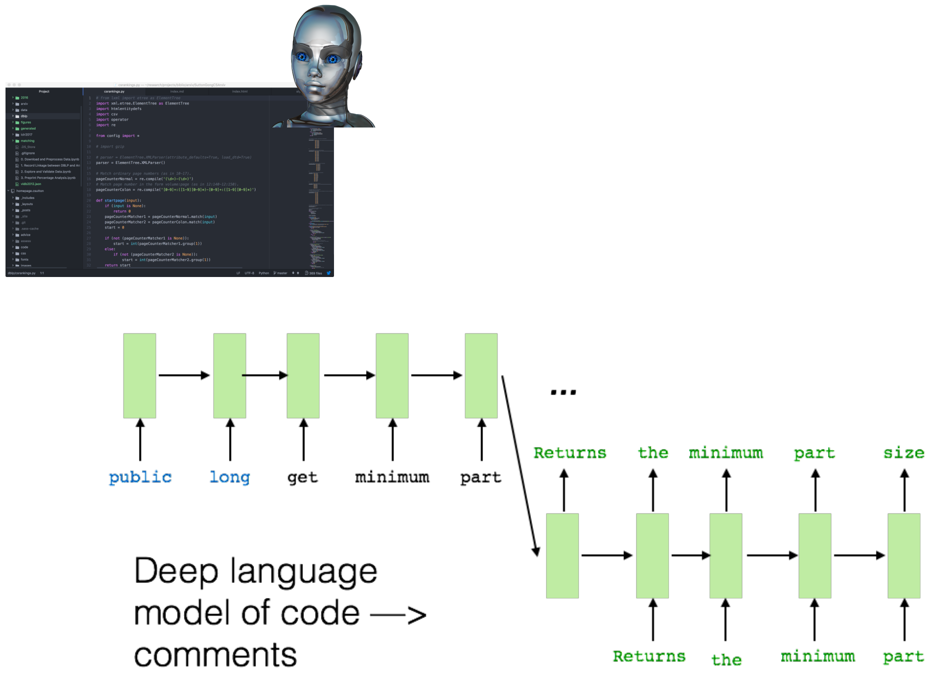
Cyclic consistency for deep generative models

[Srivastava et al; NIPS 2017]

Local “piecewise” training of conditional random fields

[Sutton & McCallum UAI 2005; ICML 2007]

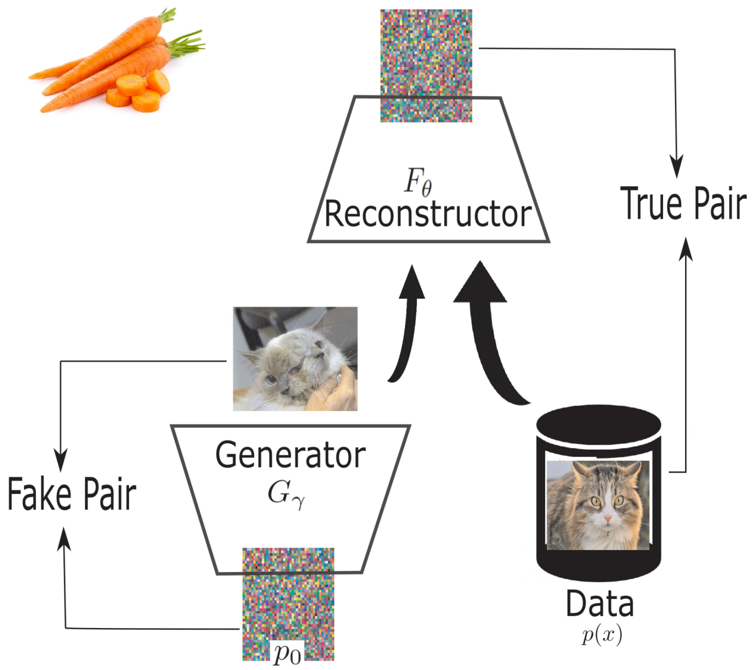




Deep language model of code —> comments

Identifying uninformative comments using deep learning

[Louis, Barr, Dash, Sutton, arXiv 2018]

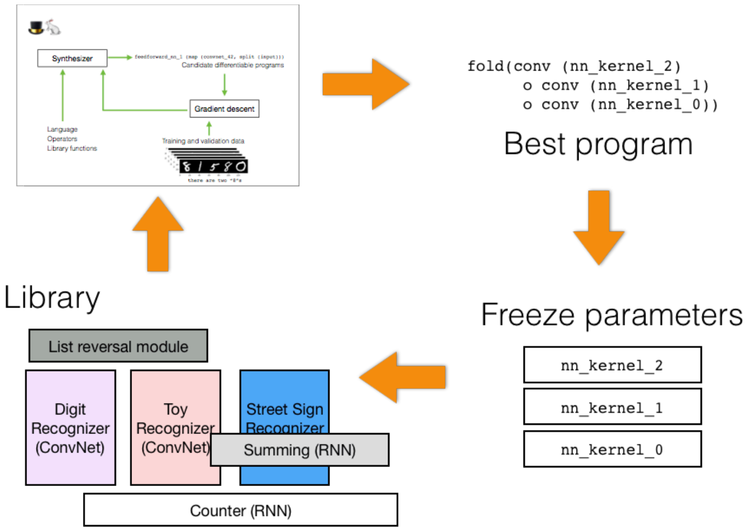


Cyclic consistency for deep generative models

[Srivastava et al; NIPS 2017]

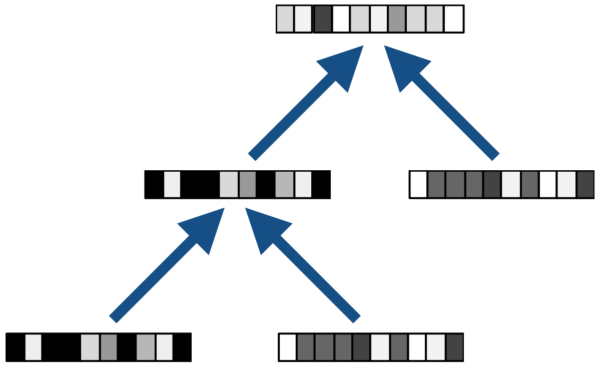
Program synthesis for neurosymbolic transfer learning

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



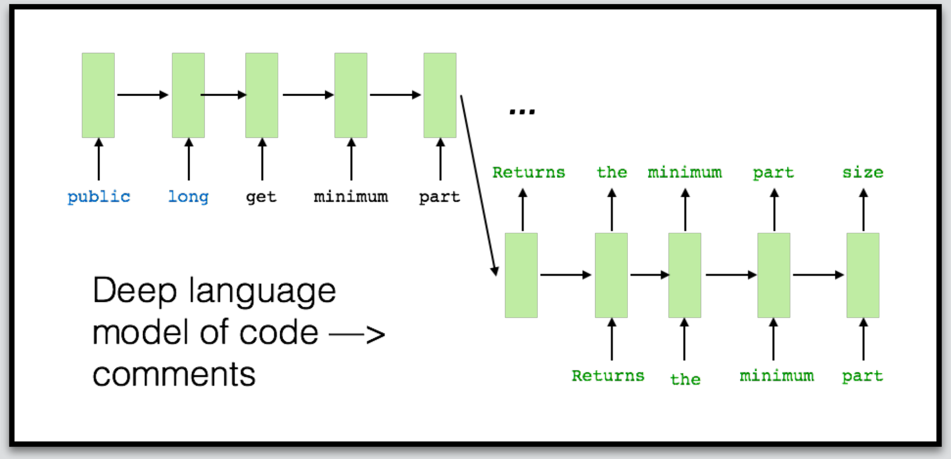
```
fold(conv (nn_kernel_2)
     o conv (nn_kernel_1)
     o conv (nn_kernel_0))
```

Best program



Equivalence networks

[Allamanis et al, ICML 2017]



Deep language model of code —> comments

Finding Uninformative Comments

http://bit.ly/sutton-dllc

[Louis, Barr, Dash, Sutton, arXiv 2018]

Source code is a means of human communication

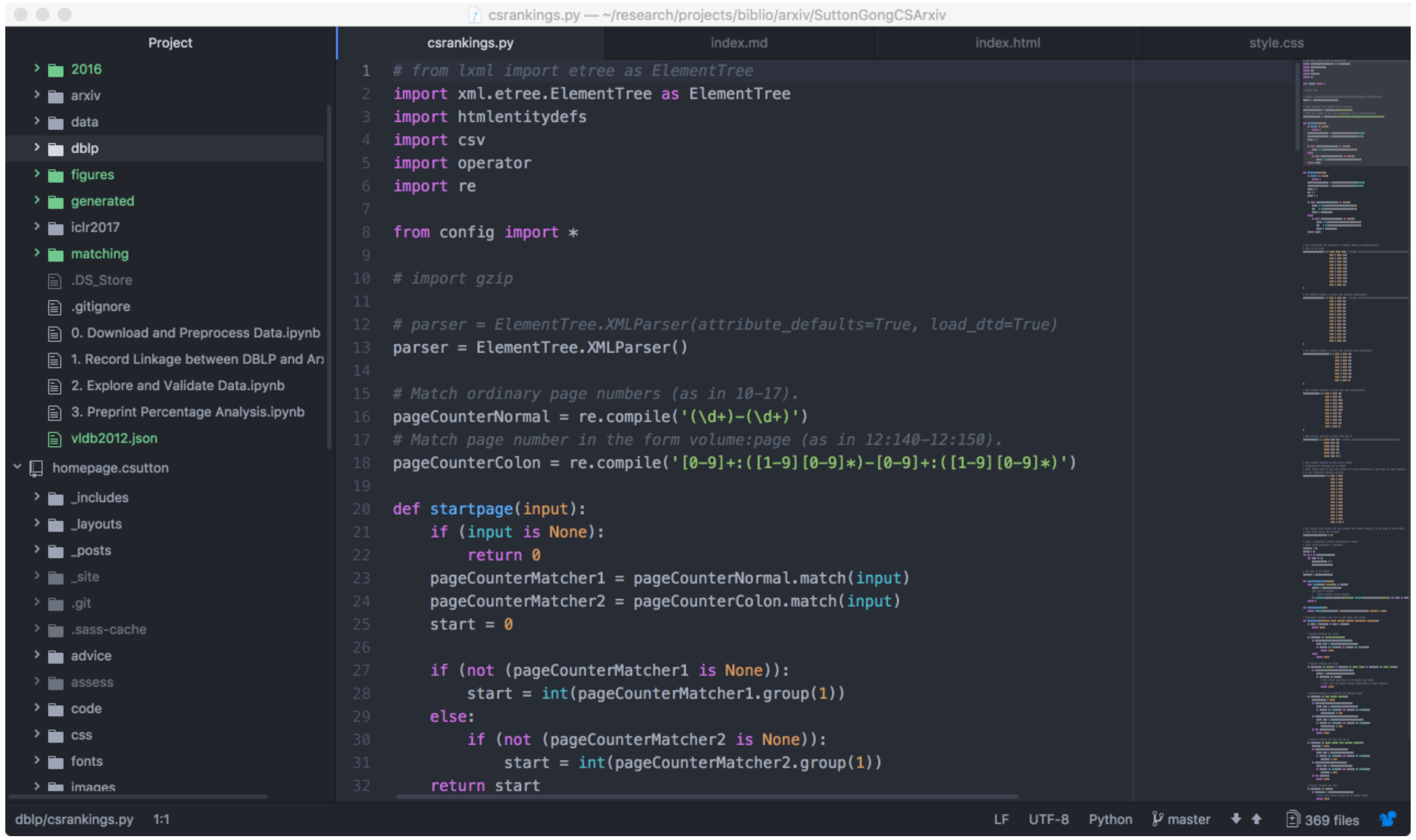
```
try{
  Node $name=$methodInvoc();
  $BODY$
}finally{
  $(Transaction).finish();
}
```

```
public static final
  String $name = $StringLit;
```

DEI: Development Environment with Intelligence

AI support for the full software lifecycle

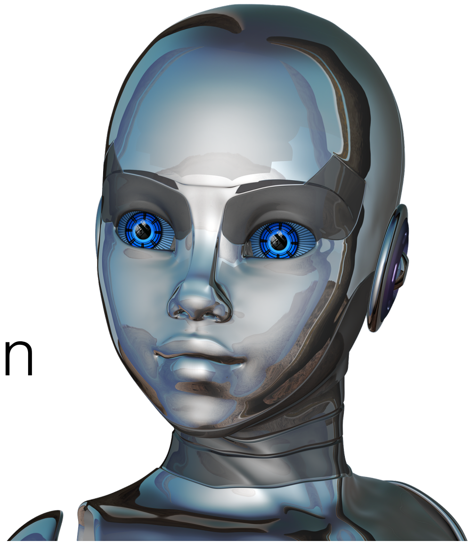
Cyberpair programming



```python
# from lxml import etree as ElementTree
import xml.etree.ElementTree as ElementTree
import htmlentitydefs
import csv
import operator
import re

from config import *

# import gzip

# parser = ElementTree.XMLParser(attribute_defaults=True, load_dtd=True)
parser = ElementTree.XMLParser()

# Match ordinary page numbers (as in 10-17).
pageCounterNormal = re.compile('(\d+)-(\d+)')
# Match page number in the form volume:page (as in 12:140-12:150).
pageCounterColon = re.compile('[0-9]+:([1-9][0-9]*)-[0-9]+:([1-9][0-9]*)')

def startpage(input):
    if (input is None):
        return 0
    pageCounterMatcher1 = pageCounterNormal.match(input)
    pageCounterMatcher2 = pageCounterColon.match(input)
    start = 0

    if (not (pageCounterMatcher1 is None)):
        start = int(pageCounterMatcher1.group(1))
    else:
        if (not (pageCounterMatcher2 is None)):
            start = int(pageCounterMatcher2.group(1))
    return start
```

- Managing avalanche of details in code
- Automate tasks without business value
- Transfer knowledge to newer developers

Suggestions on: Coding style
Bug fixes
Documentation
Debugging



Not all comments are the same…

```
/* Returns the minimum part size for upload parts.
     Decreasing the minimum part size
 causes multipart uploads to be split into a larger number
      of smaller parts. Setting
this value too low has a negative effect on transfer
     speeds, causing extra latency
and network communication for each part.
@return The minimum part size for upload parts. */
public long getMinimumUploadPartSize() {
  return minimumUploadPartSize;
}
```

The Good

The Ugly

```
/* Returns the projects entry persistence.
   @return the projects entry persistence */
public ProjectsEntryPersistence
      getProjectsEntryPersistence() {
  return projectsEntryPersistence;
}
```

Let’s discourage repetitive comments!

Shouldn’t comments repeat the code?

> “Avoid comments that just repeat what the code does.”
>
> — Google Testing Blog

> “Good comments don't repeat the code or explain it. They clarify its intent. Comments should explain, at a higher level of abstraction than the code, what you're trying to do.”
>
> — Steve McConnell, *Code Complete*

Comments a waste of time?

Downsides of comments

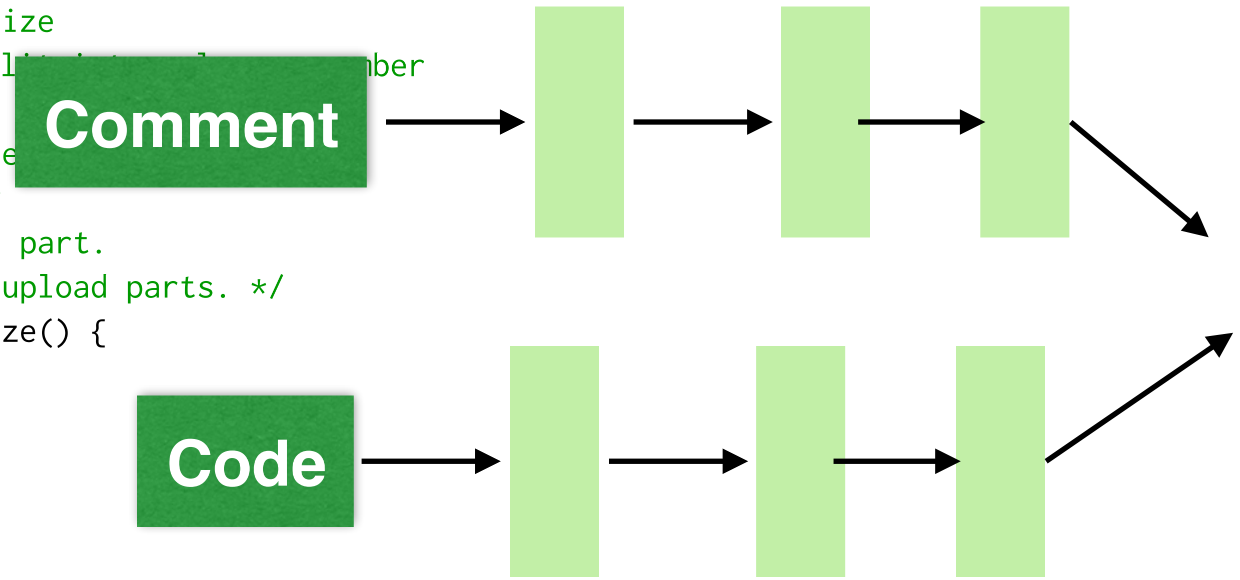
- Bad comments cause bloat
- Good comments take time
- Comments go stale

Advice: “Rewrite code instead”

Opportunity for ML / NLP

Bimodal software engineering

```
/* Returns the minimum part size for upload parts.
     Decreasing the minimum part size
 causes multipart uploads to be spl[illegible]ber
      of smaller parts. Setting
this value too low has a negative e[illegible]
     speeds, causing extra latency
and network communication for each part.
@return The minimum part size for upload parts. */
public long getMinimumUploadPartSize() {
  return minimumUploadPartSize;
}
```



Readability, staleness, completeness

…

Deep models

Predictions

Comment entailment problem

```
Returns the minimum part size for upload parts.
```

Comment sentence

```java
public long getMinimumUploadPartSize() {
  return minimumUploadPartSize;
}
```

Code

Code logically entails comment?

Code provides enough information to judge that comment sentence is true.

Inspired by textual entailment

[Dagan et al, 2013]

Examples of comment entailment

```
/**
 * Return the current registration id.
 * If result is empty, the registration has failed.
 * @return registration id, or empty string if the
      registration is not complete.
 */
public static String getRegistrationId(Context context) {
  final SharedPreferences prefs =
       context.getSharedPreferences(PREFERENCE,
       Context.MODE_PRIVATE);
  String registrationId =
       prefs.getString(``dm_registration'','');
  return registrationId;
}
```

ENTAILED

NOT ENTAILED

PARTIAL

Entailment is good? Bad?

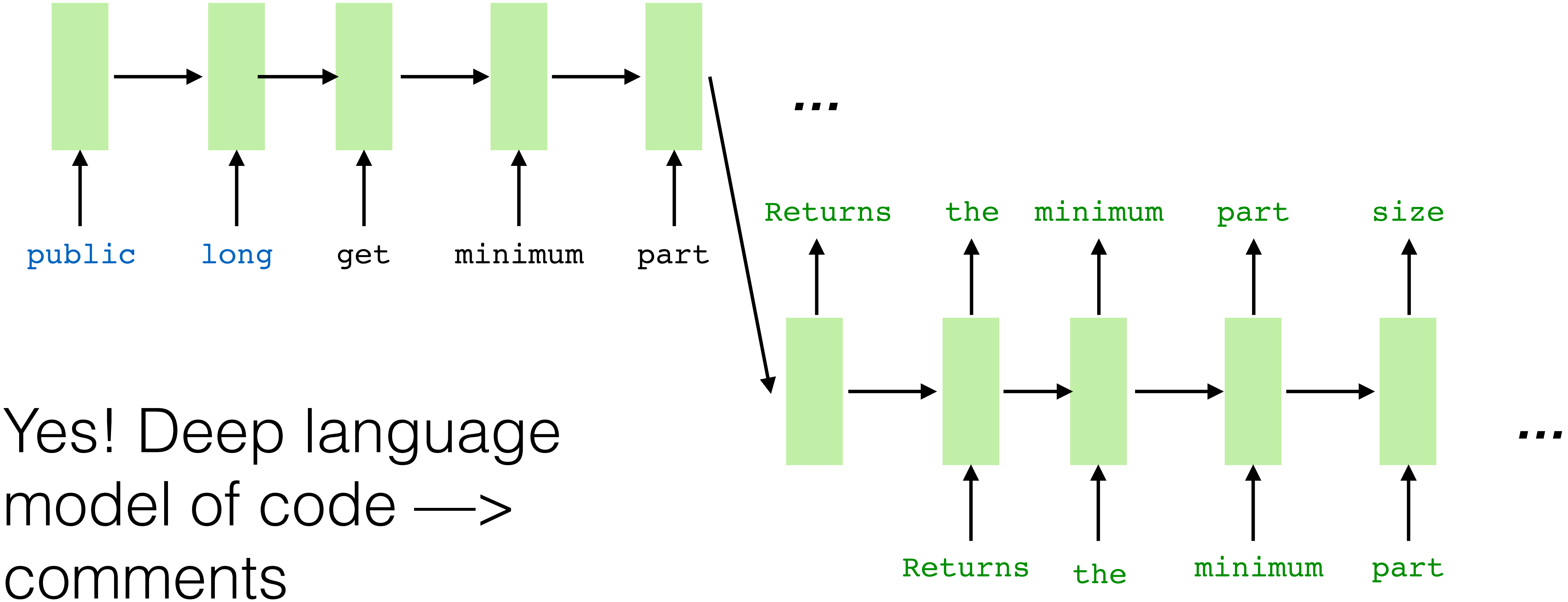
Academic: Entailment is **good** because the point of comments is to explain the code, right?

Industry: Entailment is **bad** because you're bloating the code with maintenance burden

We say: Both right! Both wrong!

	Entailed	Non-entailed
Often Good!	High-level summaries	Design rationale
Probably Bad	Restate the method signature	Copy-paste mistakes

Seq2seq for entailment



Yes! Deep language model of code —> comments

Key idea: If my deep network can predict your comment, *it wasn't a good comment!*

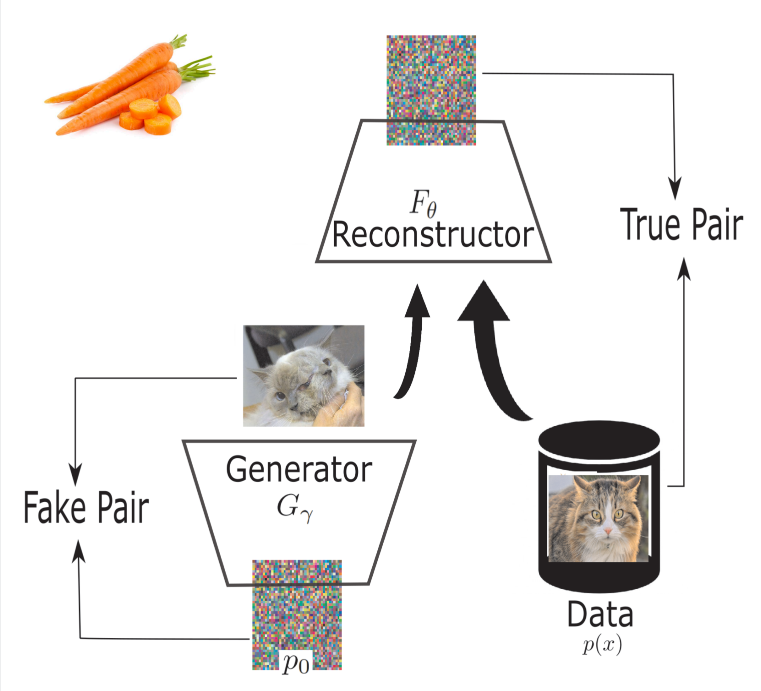
Predictive performance

	Perplexities		
Model	**Train**	**Valid**	**Test**
LM	7.80	10.34	9.87
s2s-signature	5.70	6.90	8.26
s2s-begin-end	3.44	4.18	5.31
s2s-identifier	4.50	5.34	6.00
LM English newswire			58

Human judgements

category	**count**	**avg**	**stdev**	**median**
entails	237	9.50	33.23	2.30
partly entailed	12	14.77	17.00	7.35
not entailed	39	115.73	266.65	13.35
unrelated	4	1069.73	676.71	1206.36

VEEGAN: Reducing Mode Collapse in Generative Adversarial Learning

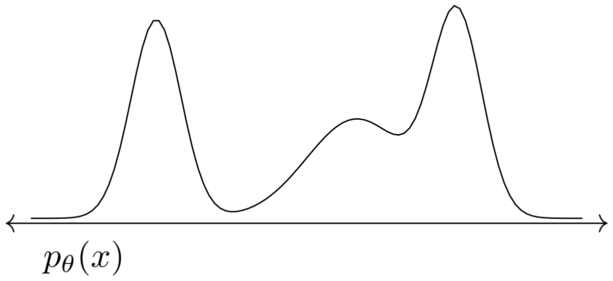


[Srivastava, Valkov, Russell, Gutmann, Sutton, NIPS 2017]

Generative Adversarial Networks

[Goodfellow et al, 2014]

Classical probabilistic modelling



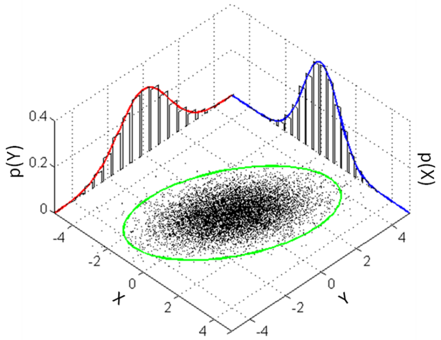
0.00003

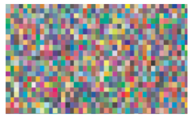
Input

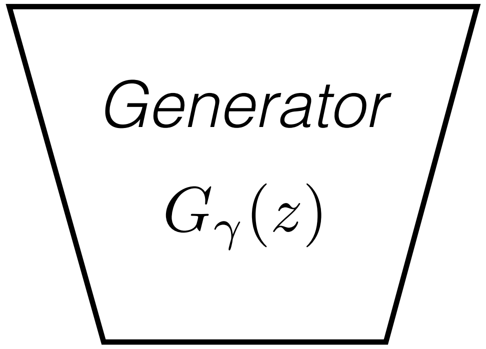
Explicit model

Density value

Implicit probabilistic modelling







Generator

$G_\gamma(z)$

Gaussian
$p(z)$

Representation
z

Generator network
$G_\gamma(z)$

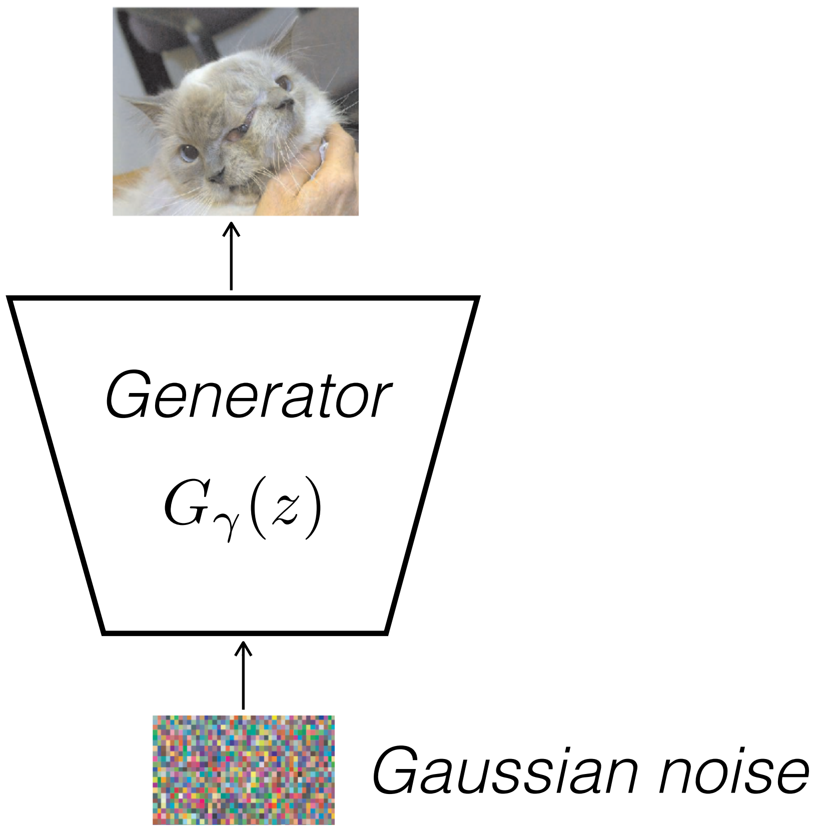
Image
x

Sampling procedure for $p_\theta(x)$

How to train?

Can’t use maximum likelihood. There is no likelihood!

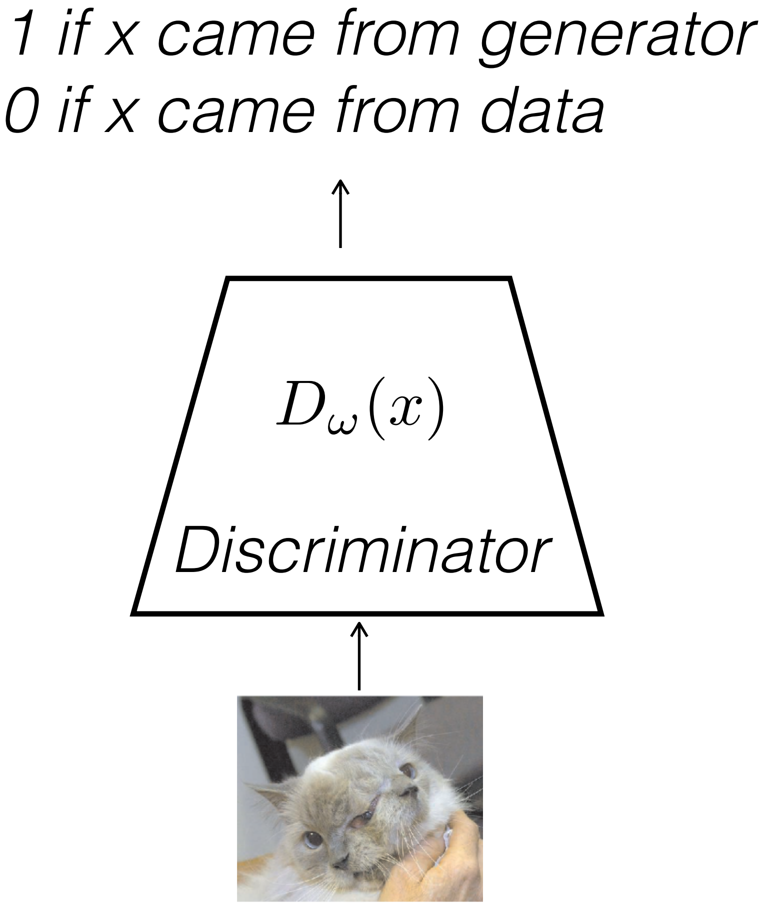
Instead define a game



Generator

$G_\gamma(z)$

Gaussian noise



1 if x came from generator
0 if x came from data

$D_\omega(x)$

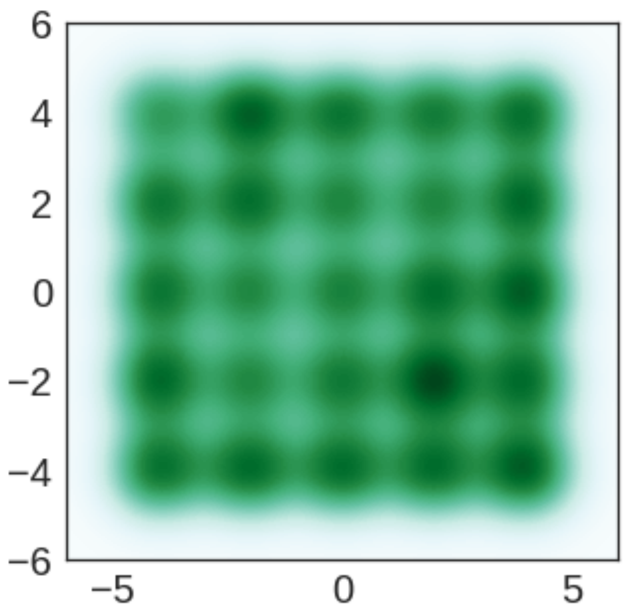
Discriminator

Optimize

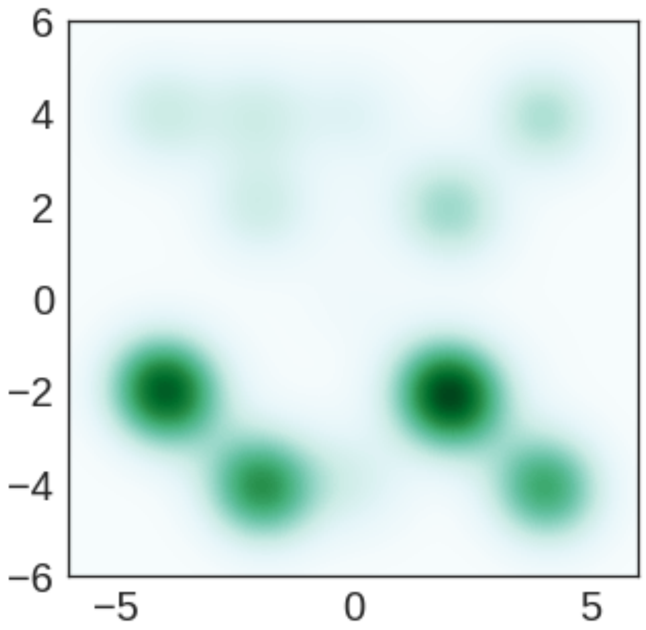
$$\max_{\omega} \min_{\gamma} \mathcal{O}_{\text{GAN}}(\omega, \gamma) := E_z\left[\log D_\omega(G_\gamma(z))\right] + E_x\left[\log\left(1 - D_\omega(x)\right)\right]$$

Mode Collapse

Example from 2D mixture of Gaussians



True data



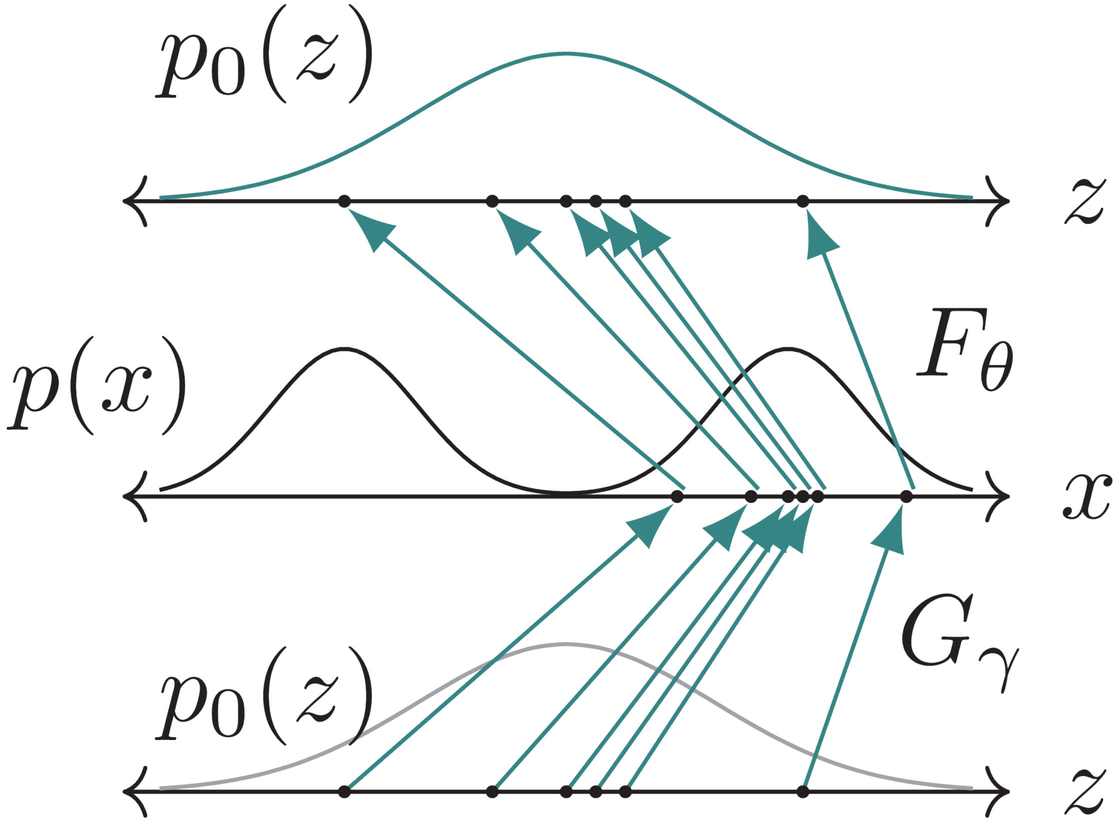
Samples from GAN

VEEGAN: Detecting collapse

Train F_θ to: 1. map true data to Gaussian

2. approximately invert the generator

Then it can help detect mode collapse:



Output of reconstructor

True data distribution

Generator input

VEEGAN: A Variational Encoder Enhancement to Generative Adversarial Nets

VEEGAN: Autoencoding Noise

Alternate:

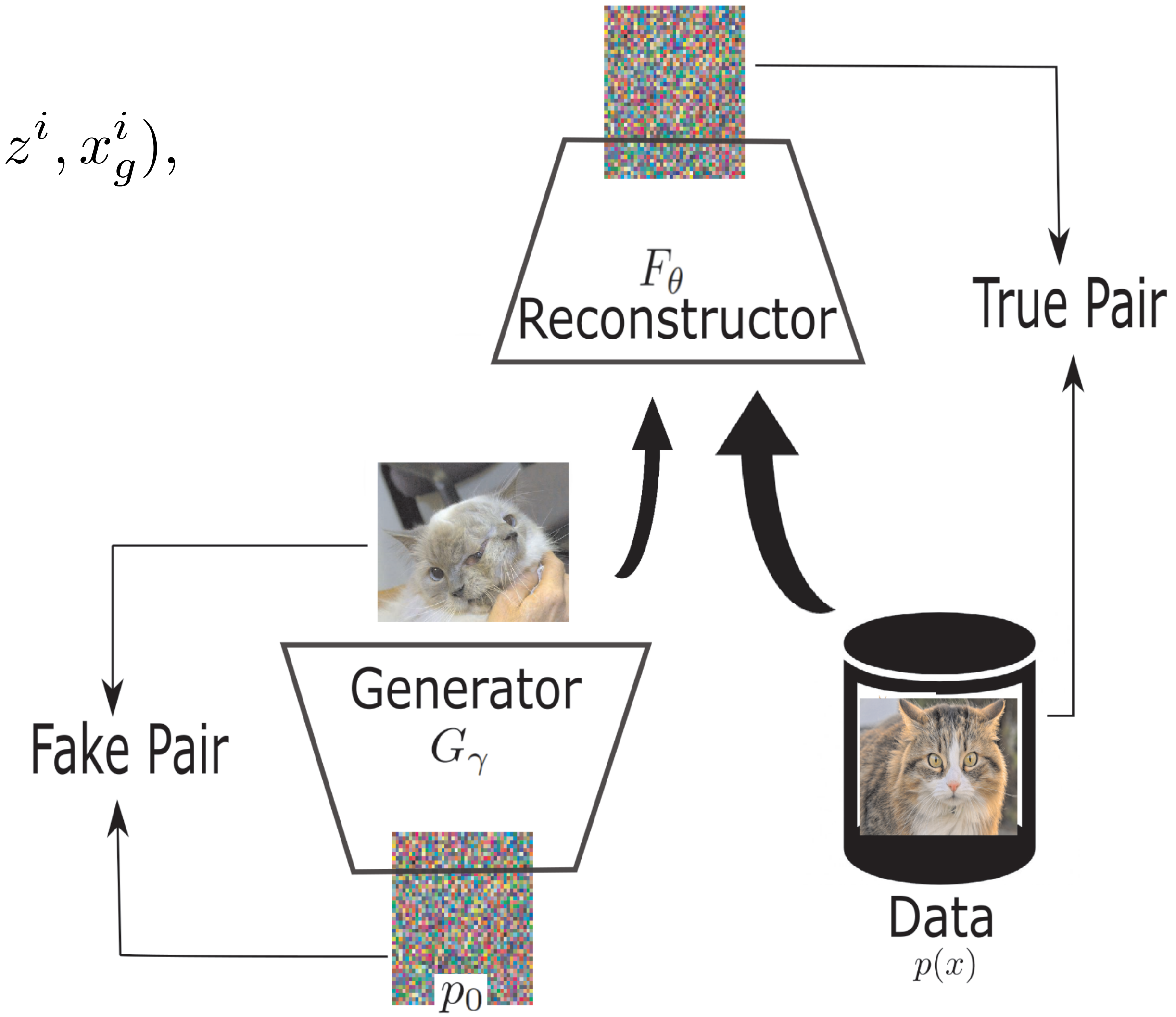
Train discriminator

$$O_{\mathrm{LR}}(\omega, \gamma, \theta) = -\mathbb{E}_{\gamma}[\log(\sigma(D_{\omega}(z_T, x_G)))] - \mathbb{E}_{\theta}[\log(1 - \sigma(D_{\omega}(z_F, x_T)))]$$

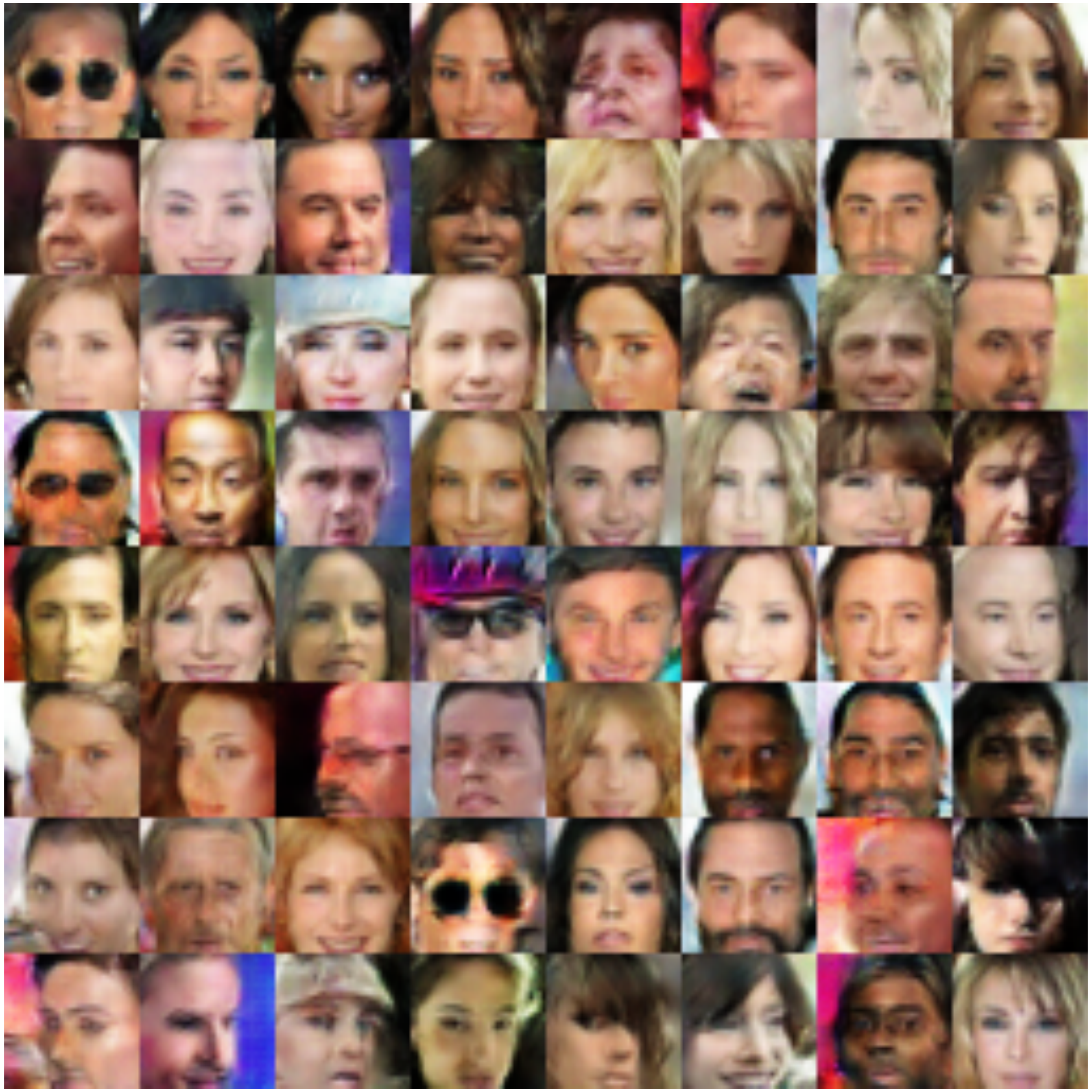
Train generator and reconstructor

$$\hat{\mathcal{O}}(\omega, \gamma, \theta) = \frac{1}{N}\sum_{i=1}^{N} \mathcal{D}_{\omega}(z^i, x_g^i) + \frac{1}{N}\sum_{i=1}^{N} d(z^i, x_g^i),$$

Much less susceptible to mode collapse than other competing methods

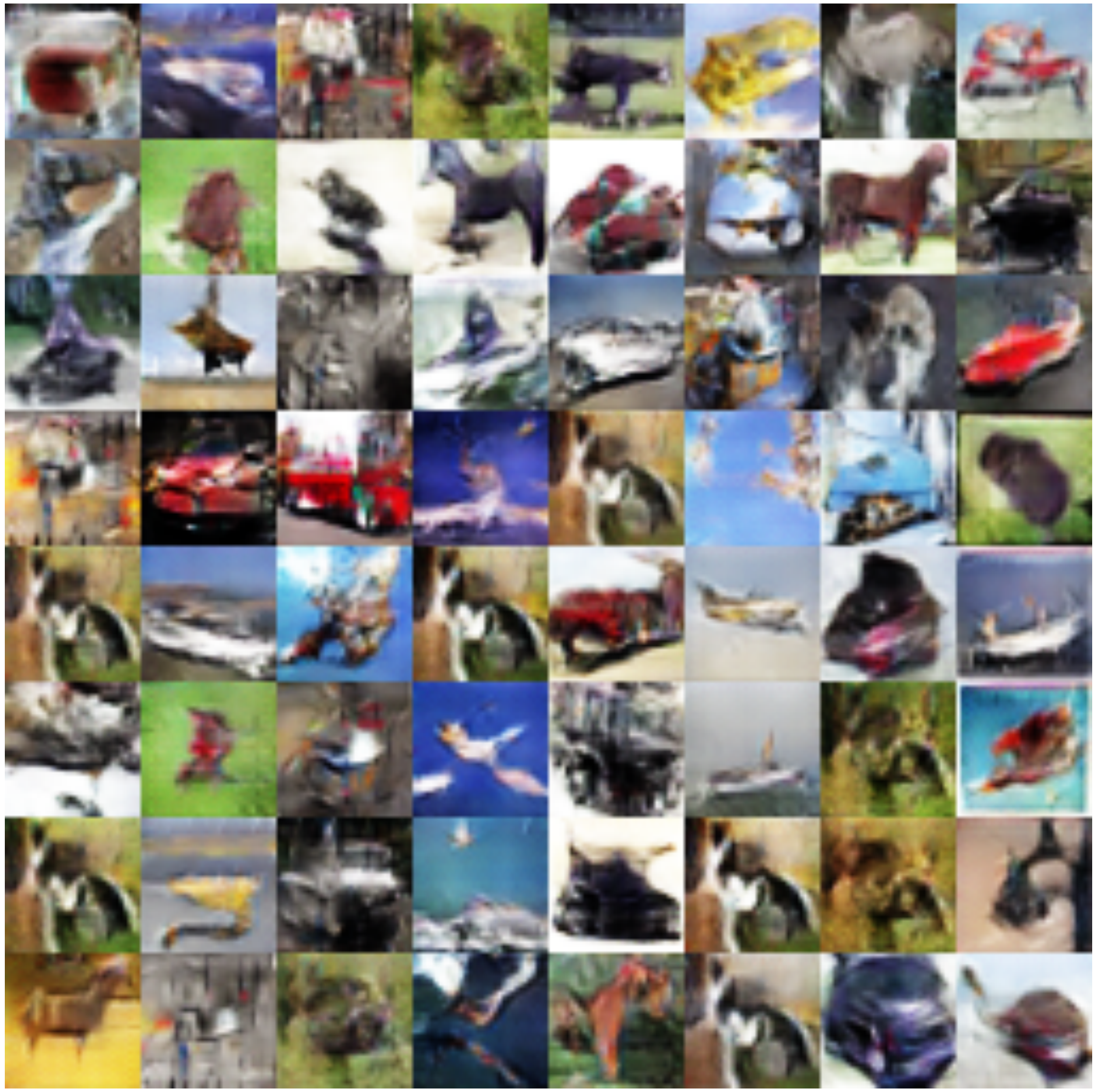


Examples of generated images

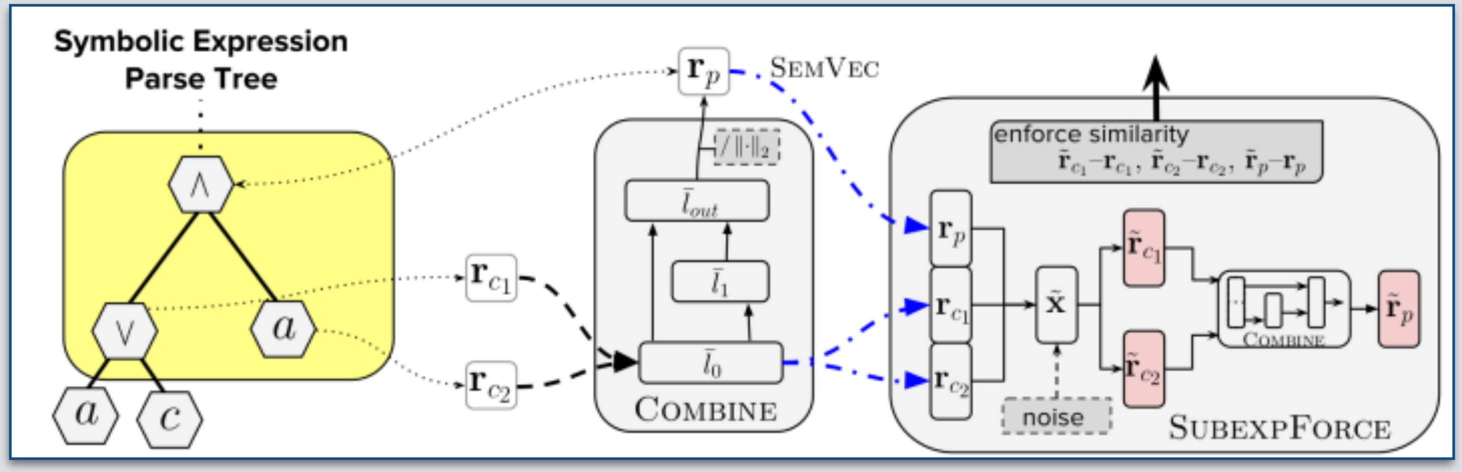


Celebrity faces

Examples of generated images



CIFAR-10 natural images

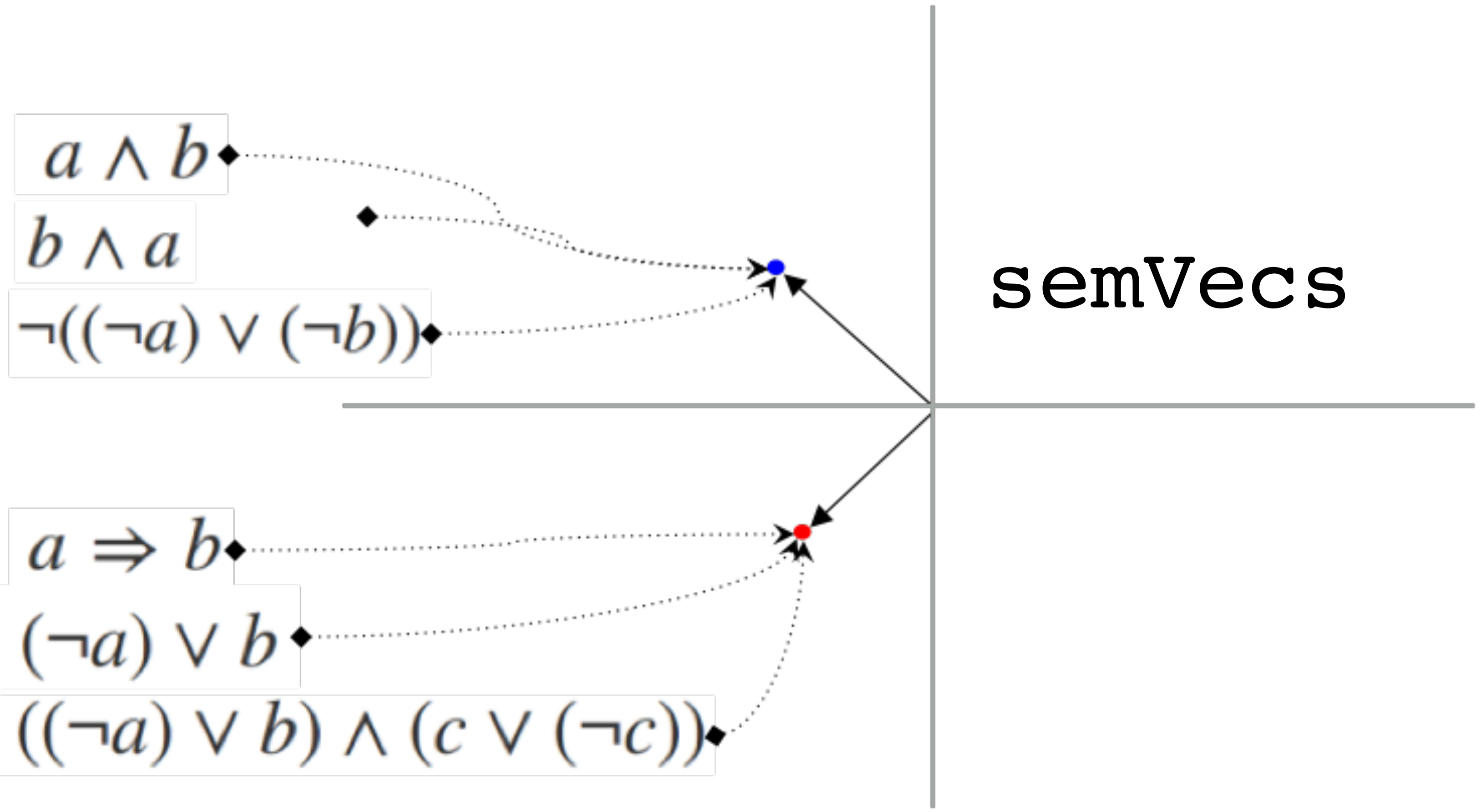


Continuous Representations of Symbolic Expressions

http://bit.ly/sutton-dllc

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

Can vectors help symbols?

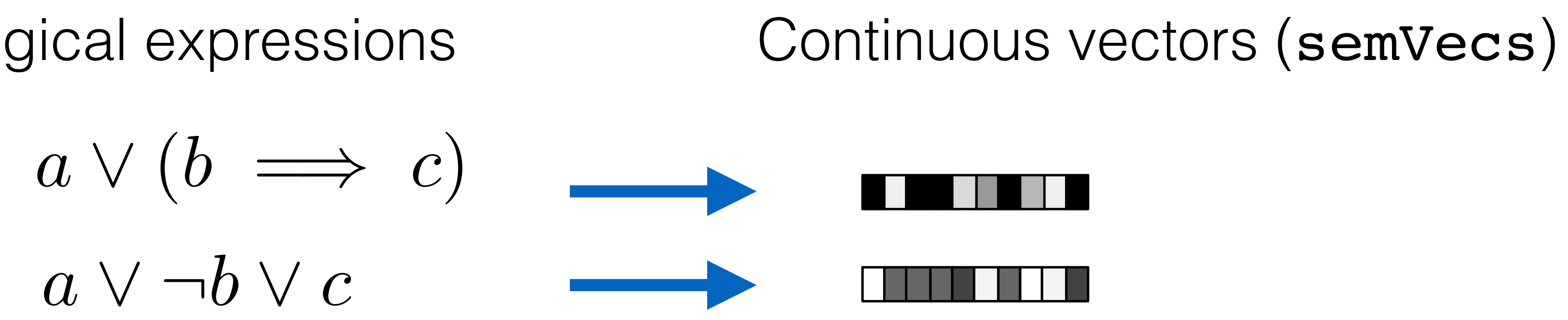

$a \wedge b$
$b \wedge a$
$\neg((\neg a) \vee (\neg b))$
semVecs
$a \Rightarrow b$
$(\neg a) \vee b$
$((\neg a) \vee b) \wedge (c \vee (\neg c))$

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

Continuous vectors (`semVecs`)

$a \lor (b \implies c)$

$a \lor \neg b \lor c$

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

Theorem Proving

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

Program Synthesis

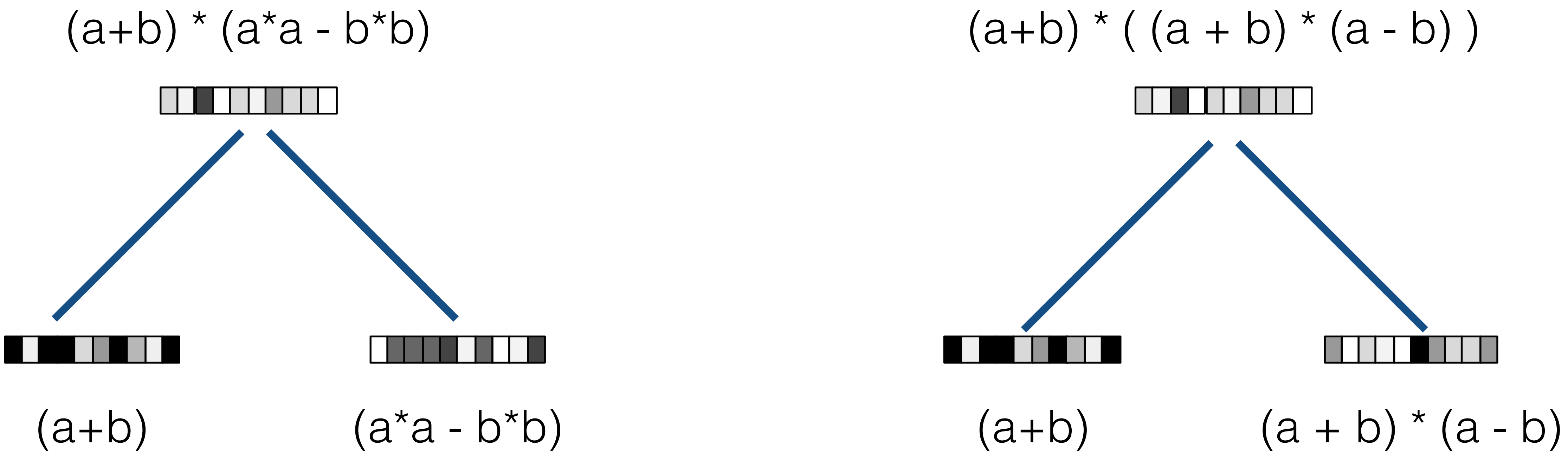
[Gulwani et al, CACM 2015]

Inductive Logic Programming

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

Transfer Learning

Desiderata

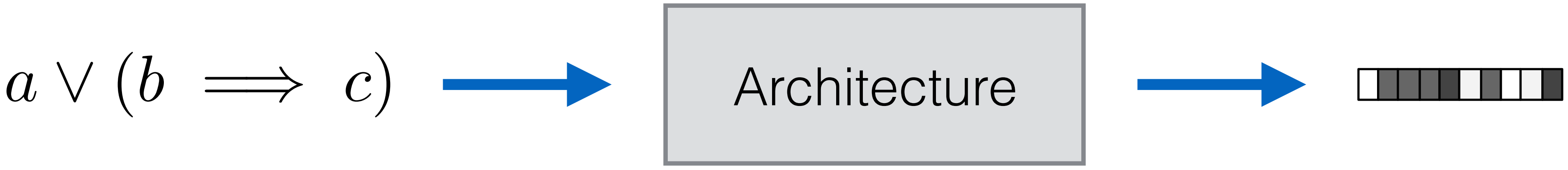


Syntax directed: Semantics is compositional

Not too much: Small syntax change —> big semantics

“man bites dog” problem

Computing semVecs

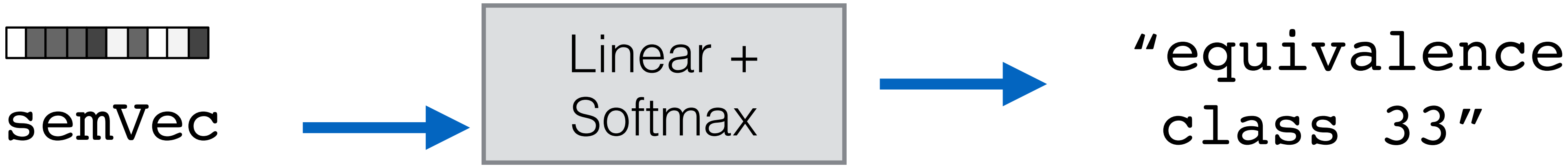


$a \vee (b \implies c)$

Architecture

Training

Partition training expressions into equivalence classes



`semVec`

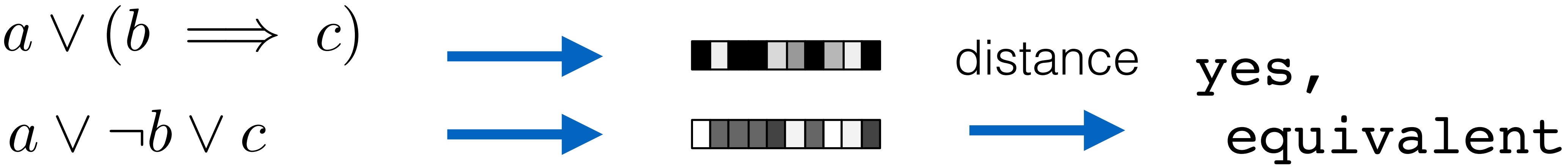
Linear +
Softmax

`“equivalence class 33”`

Use a supervised max-margin loss

Testing

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



$a \vee (b \implies c)$

$a \vee \neg b \vee c$

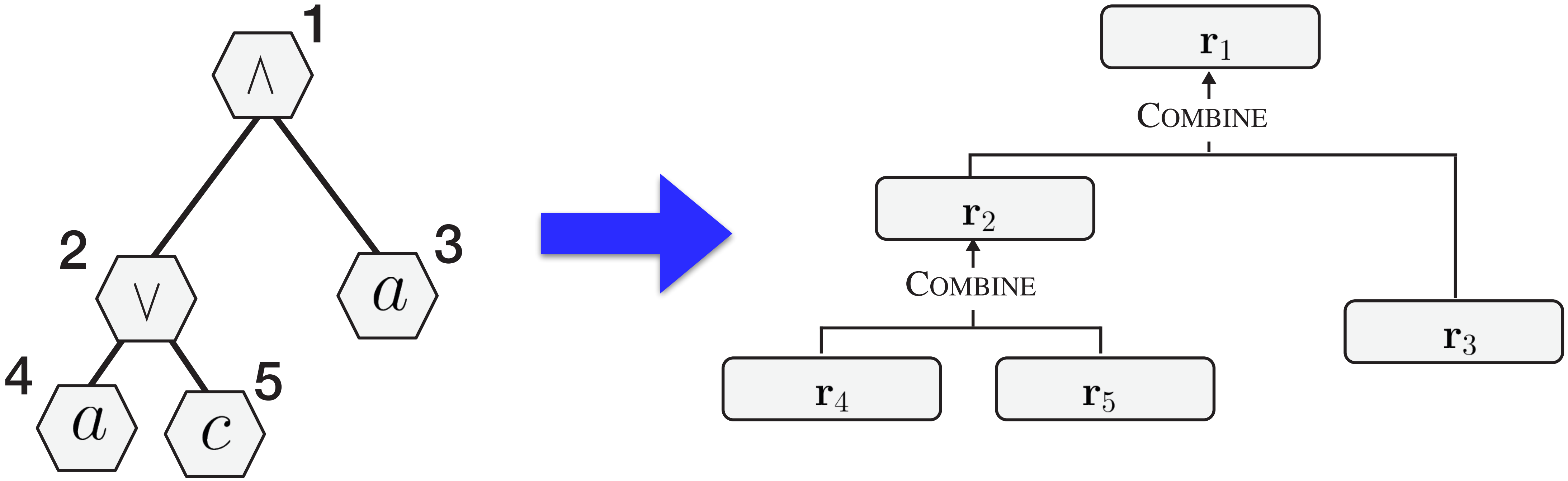
distance

`yes, equivalent`

Allows zero-shot learning on equivalence classes.

Recursive NN (TreeNN)

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



Syntax tree

Network architecture

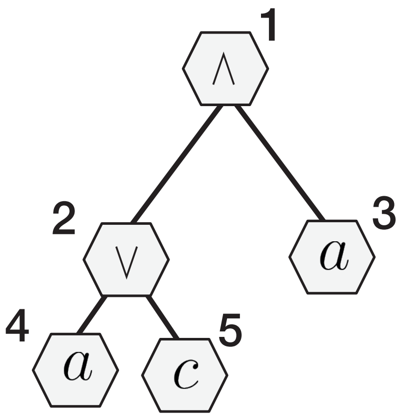
Problem: Representations mostly syntactic. Too much syntax!

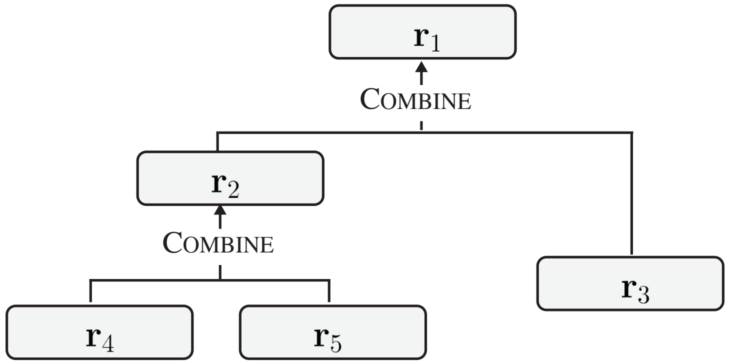
[Socher et al, 2011, 2013]

EqNet

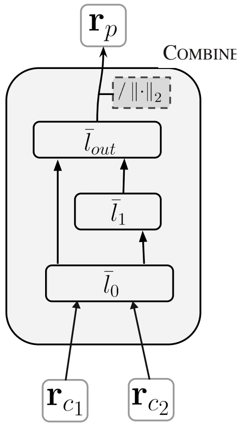
Start with TreeNNs

$(a \vee c) \wedge a$





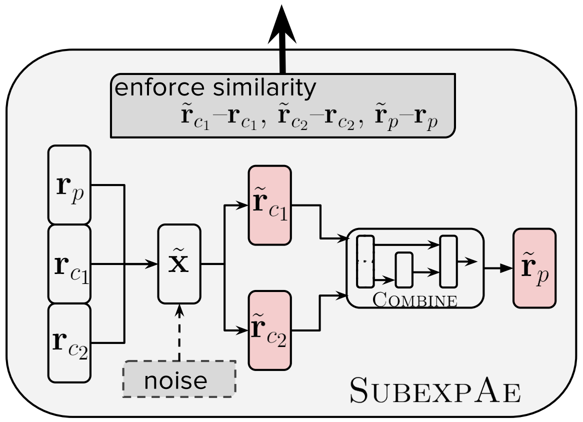
Add:



Moar! Layers!

$$\|\cdot\|_2$$

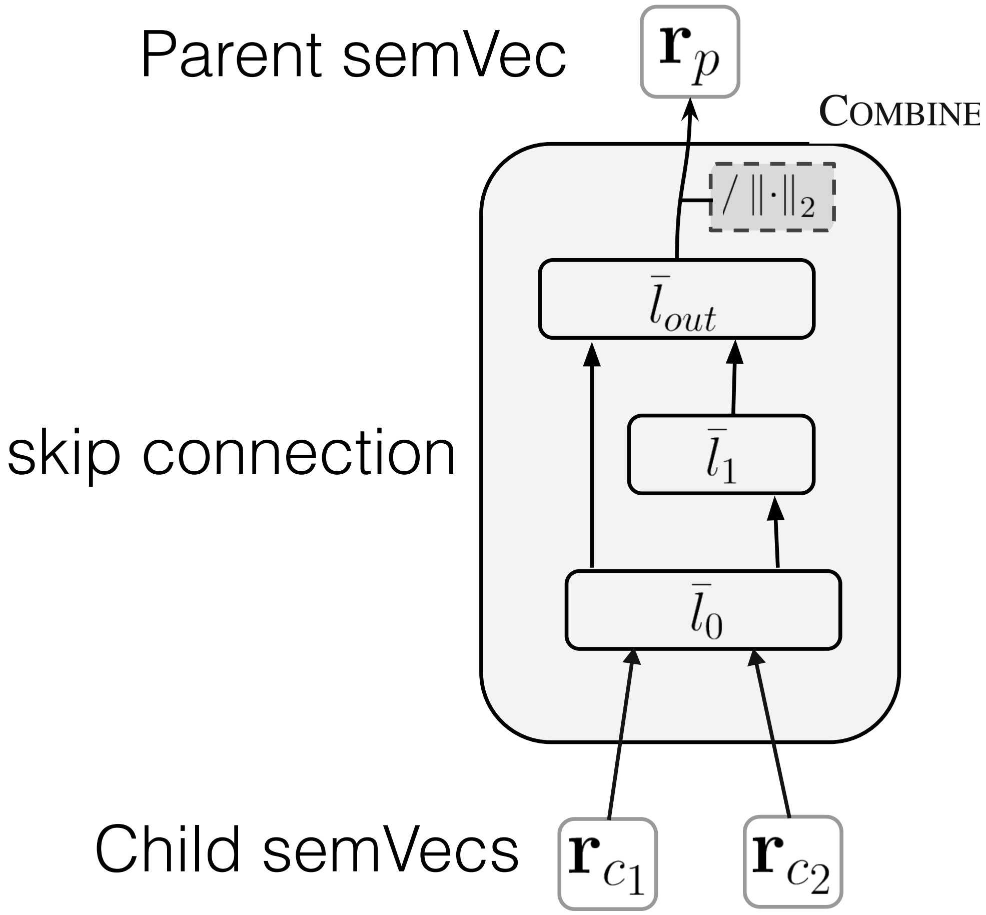
Normalization


enforce similarity
$\tilde{\mathbf{r}}_{c_1}$–$\mathbf{r}_{c_1}$, $\tilde{\mathbf{r}}_{c_2}$–$\mathbf{r}_{c_2}$, $\tilde{\mathbf{r}}_p$–$\mathbf{r}_p$
$\mathbf{r}_p$
$\mathbf{r}_{c_1}$
$\mathbf{r}_{c_2}$
$\tilde{\mathbf{x}}$
noise
$\tilde{\mathbf{r}}_{c_1}$
$\tilde{\mathbf{r}}_{c_2}$
COMBINE
$\tilde{\mathbf{r}}_p$
SUBEXPAE

Subexpression AE

Layers and Normalization

For one syntactic parent-child



```
COMBINE $(\mathbf{r}_{c_0}, \ldots, \mathbf{r}_{c_k}, \tau_p)$
    $\bar{l}_0 \leftarrow [\mathbf{r}_{c_0}, \ldots, \mathbf{r}_{c_k}]$
    $\bar{l}_1 \leftarrow \sigma\left(W_{i,\tau_p} \cdot \bar{l}_0\right)$
    $\bar{l}_{out} \leftarrow W_{o0,\tau_p} \cdot \bar{l}_0 + W_{o1,\tau_p} \cdot \bar{l}_1$
    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**

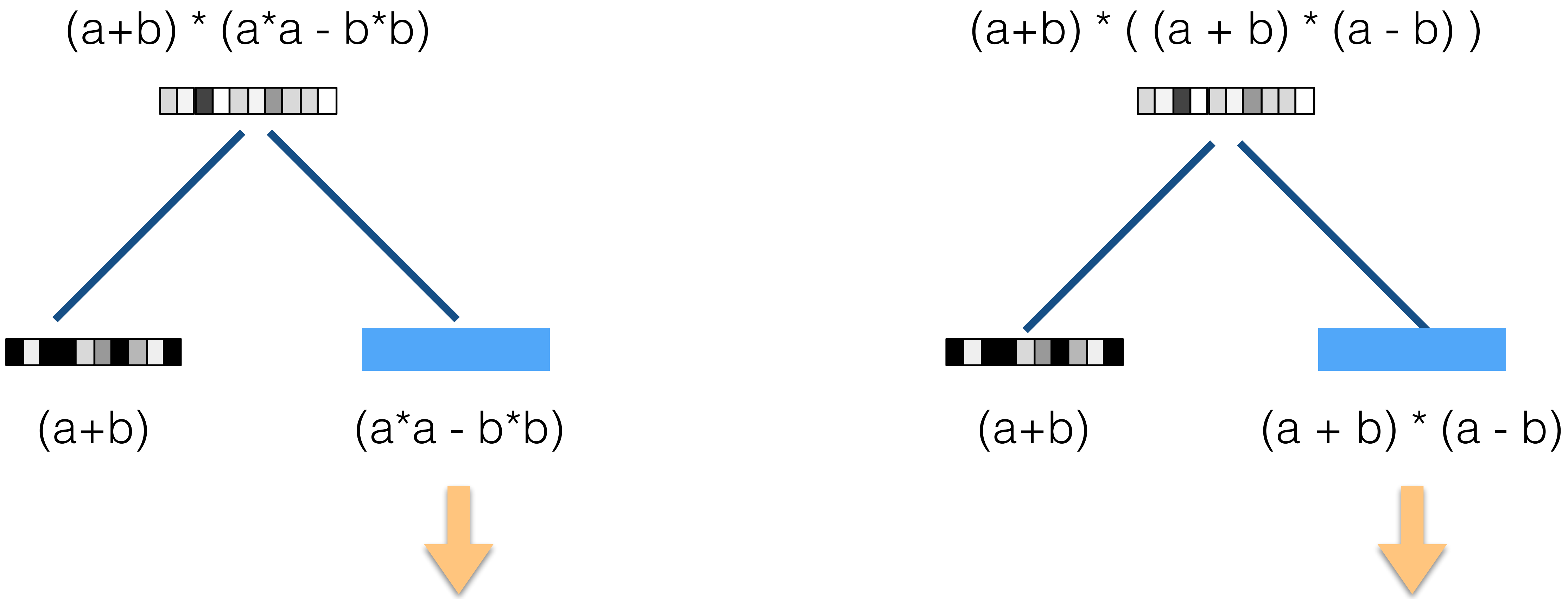
But **parents** provide info re **children**

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

Unification propagates this info automatically

How to map to continuous space?

SubexprAE Motivation



(a+b) * (a*a - b*b)

(a+b)

(a*a - b*b)

(a+b) * ((a + b) * (a - b))

(a+b)

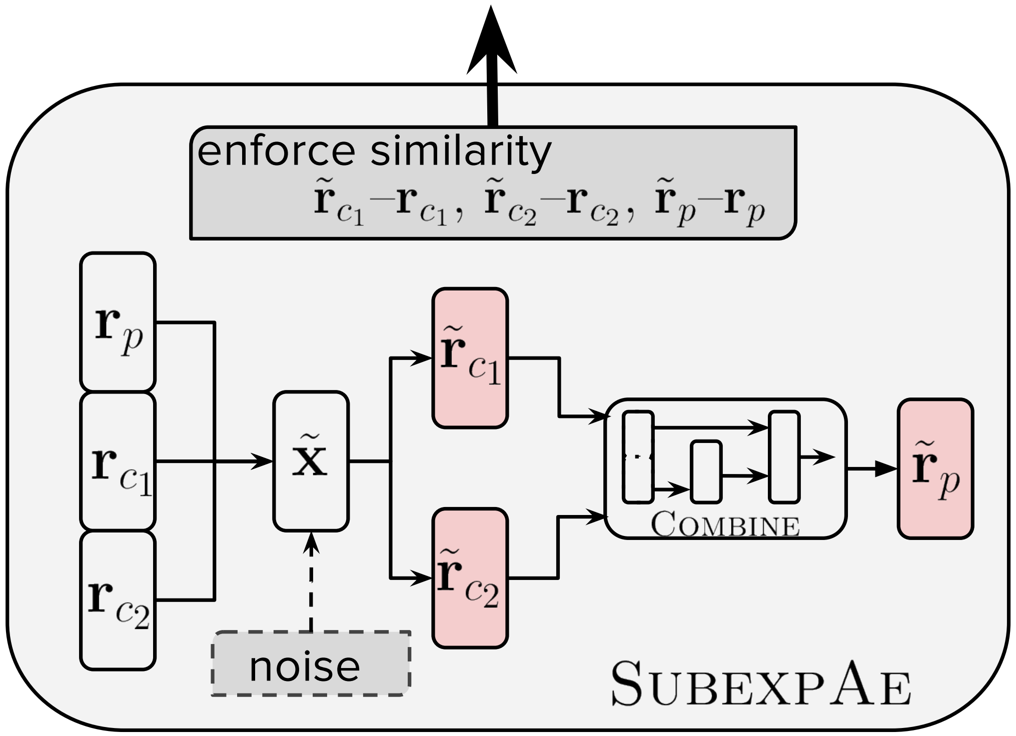
(a + b) * (a - b)

ensure this prediction problem is “easy”

semantic classes will be clustered together

Subexpression Autoencoder

For every node in syntax tree, add regularisation


enforce similarity
$\tilde{\mathbf{r}}_{c_1} - \mathbf{r}_{c_1}$, $\tilde{\mathbf{r}}_{c_2} - \mathbf{r}_{c_2}$, $\tilde{\mathbf{r}}_p - \mathbf{r}_p$
$\mathbf{r}_p$
$\mathbf{r}_{c_1}$
$\mathbf{r}_{c_2}$
$\tilde{\mathbf{x}}$
noise
$\tilde{\mathbf{r}}_{c_1}$
$\tilde{\mathbf{r}}_{c_2}$
COMBINE
$\tilde{\mathbf{r}}_p$
SUBEXPAE

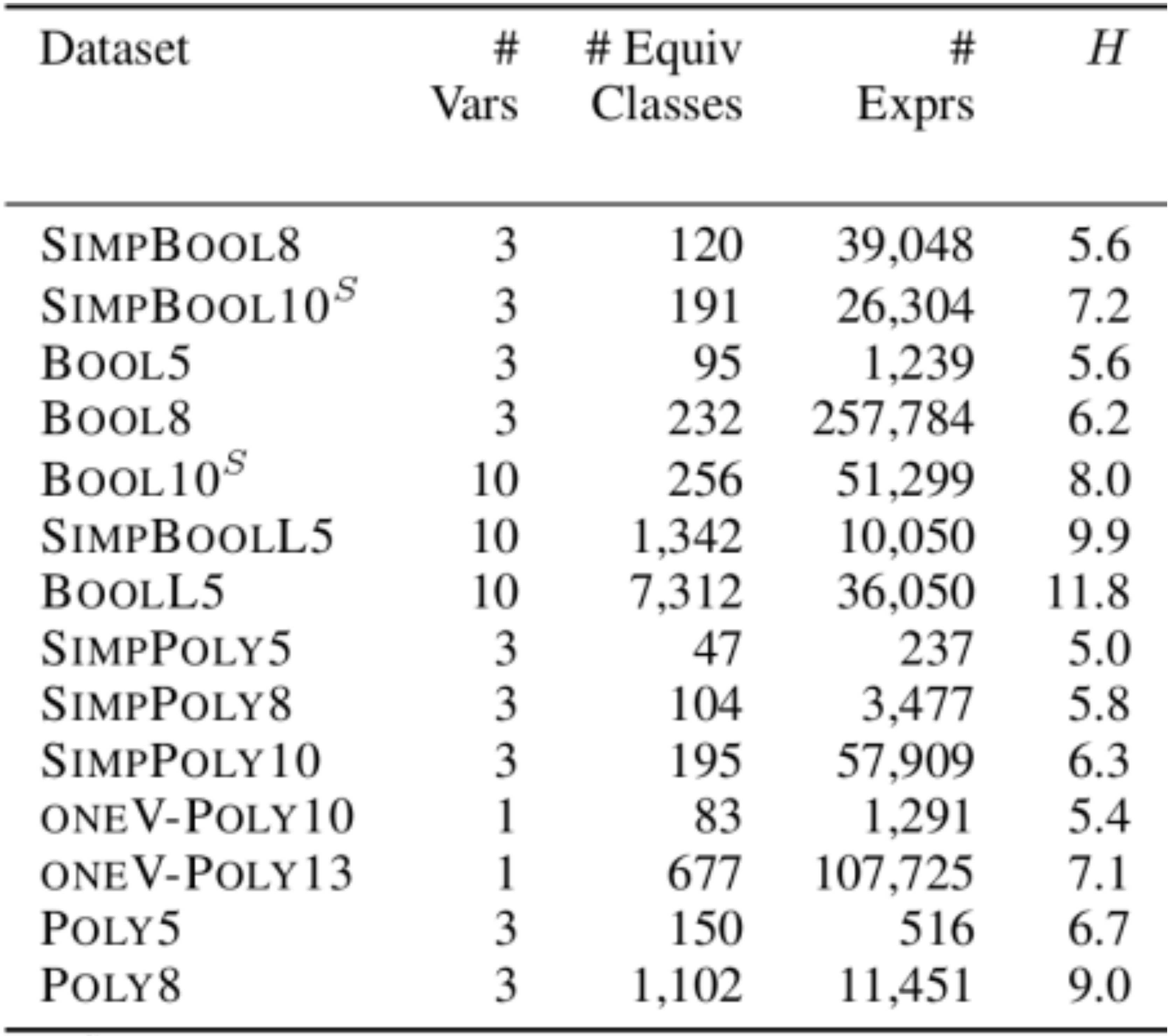
Denoising autoencoder plus bottleneck on (parent, child1, child2) semVecs

Intention is

Bottleneck → Abstraction

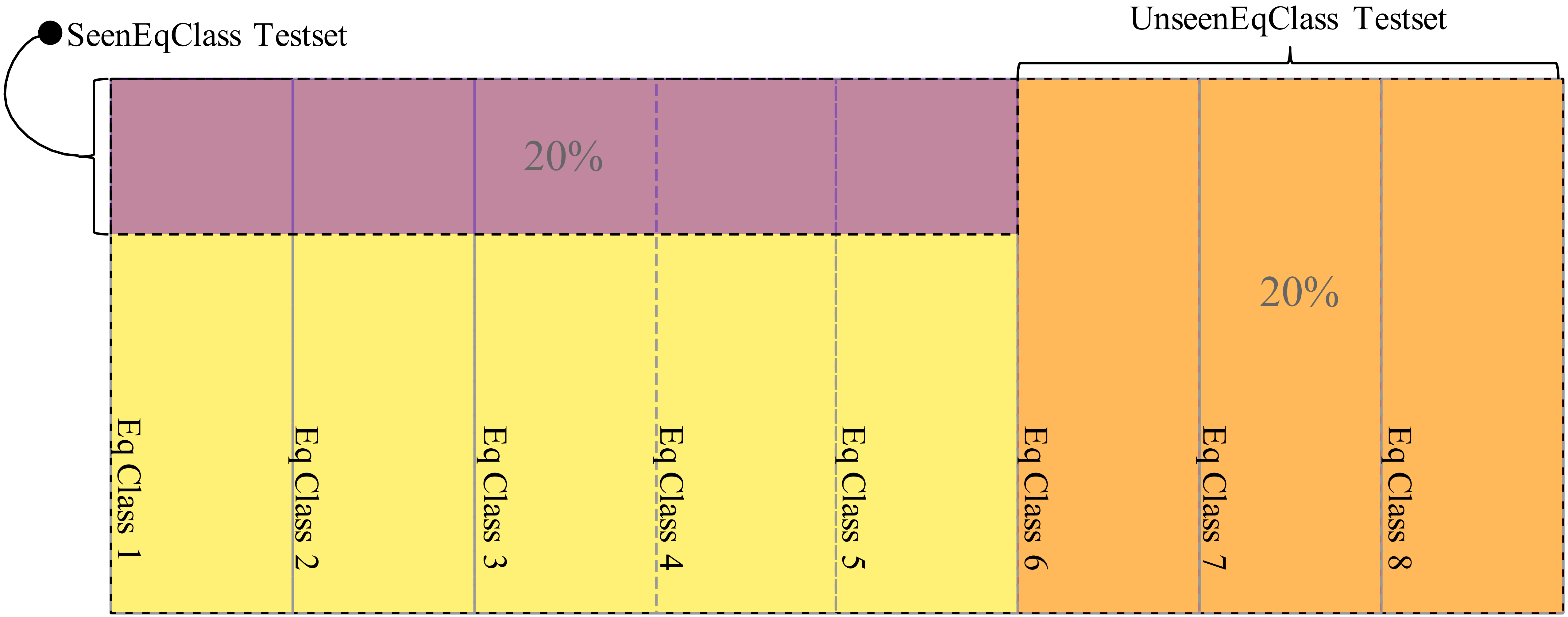
Denoising → Reversibility

Evaluation

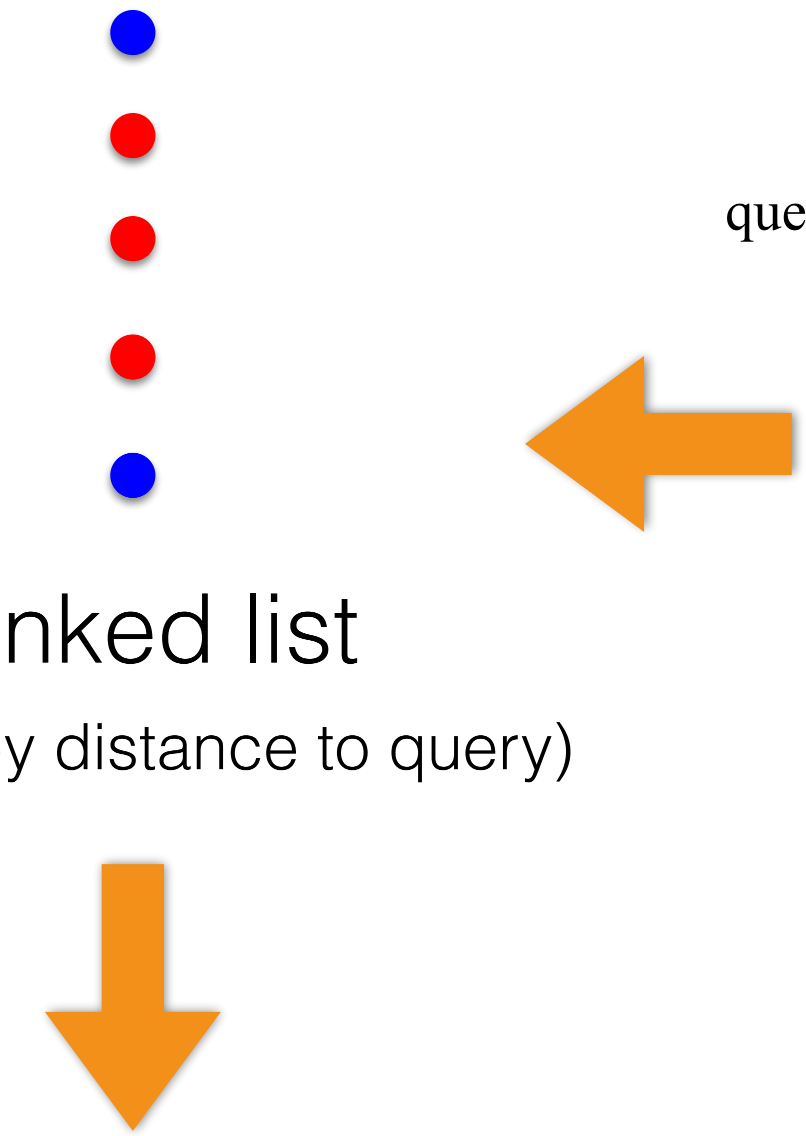


Dataset	# Vars	# Equiv Classes	# Exprs	H
SimpBool8	3	120	39,048	5.6
SimpBool10^S	3	191	26,304	7.2
Bool5	3	95	1,239	5.6
Bool8	3	232	257,784	6.2
Bool10^S	10	256	51,299	8.0
SimpBoolL5	10	1,342	10,050	9.9
BoolL5	10	7,312	36,050	11.8
SimpPoly5	3	47	237	5.0
SimpPoly8	3	104	3,477	5.8
SimpPoly10	3	195	57,909	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

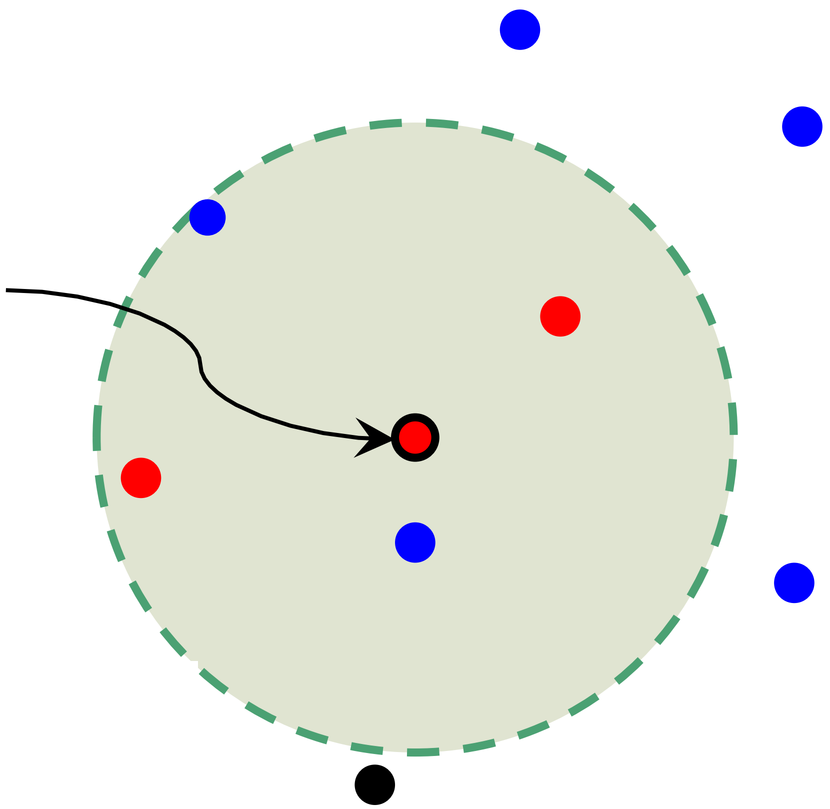


Ranked list

(by distance to query)

Precision
and recall

query point

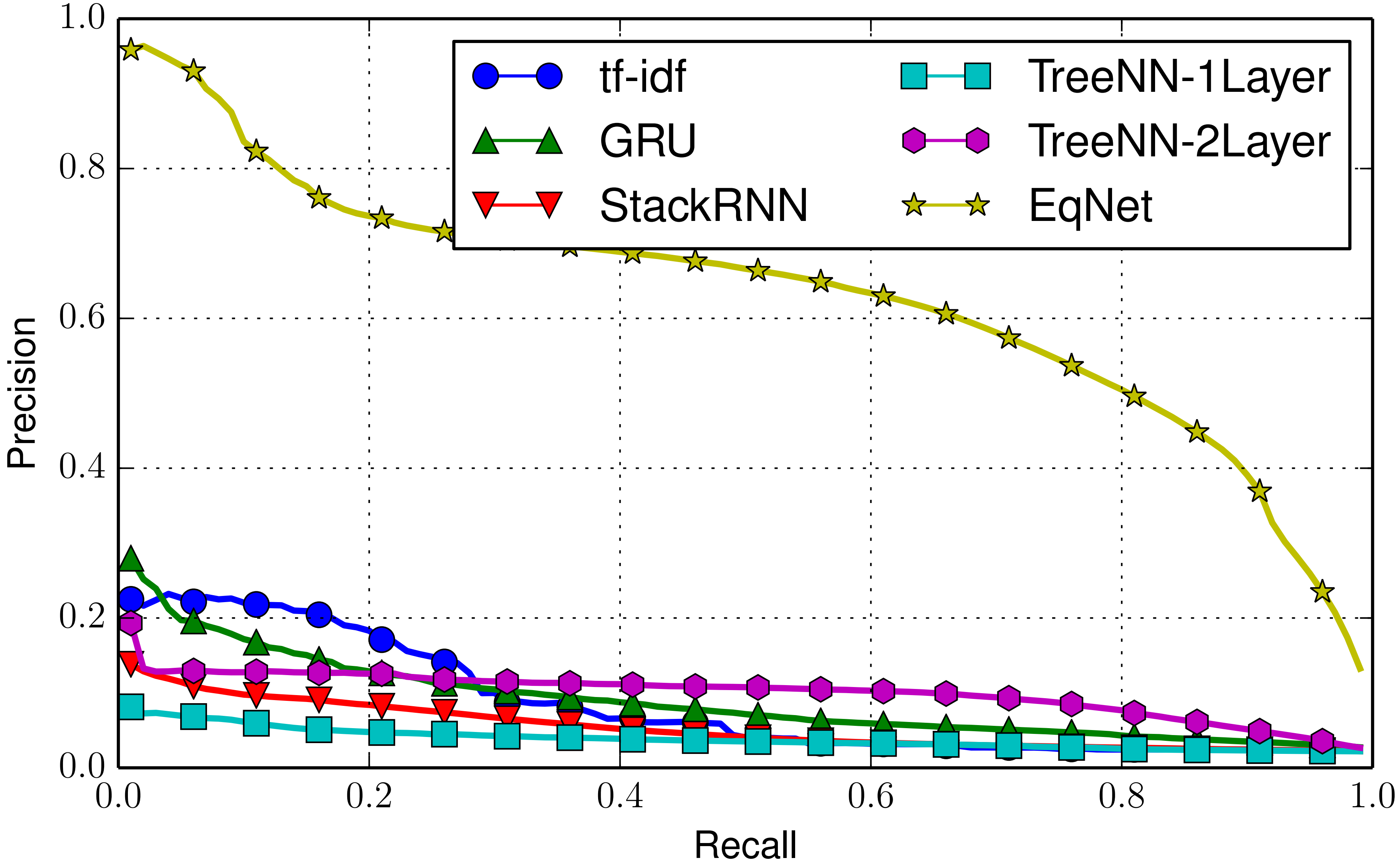


semantic space

Same equivalence class to query
Different equivalence class to query

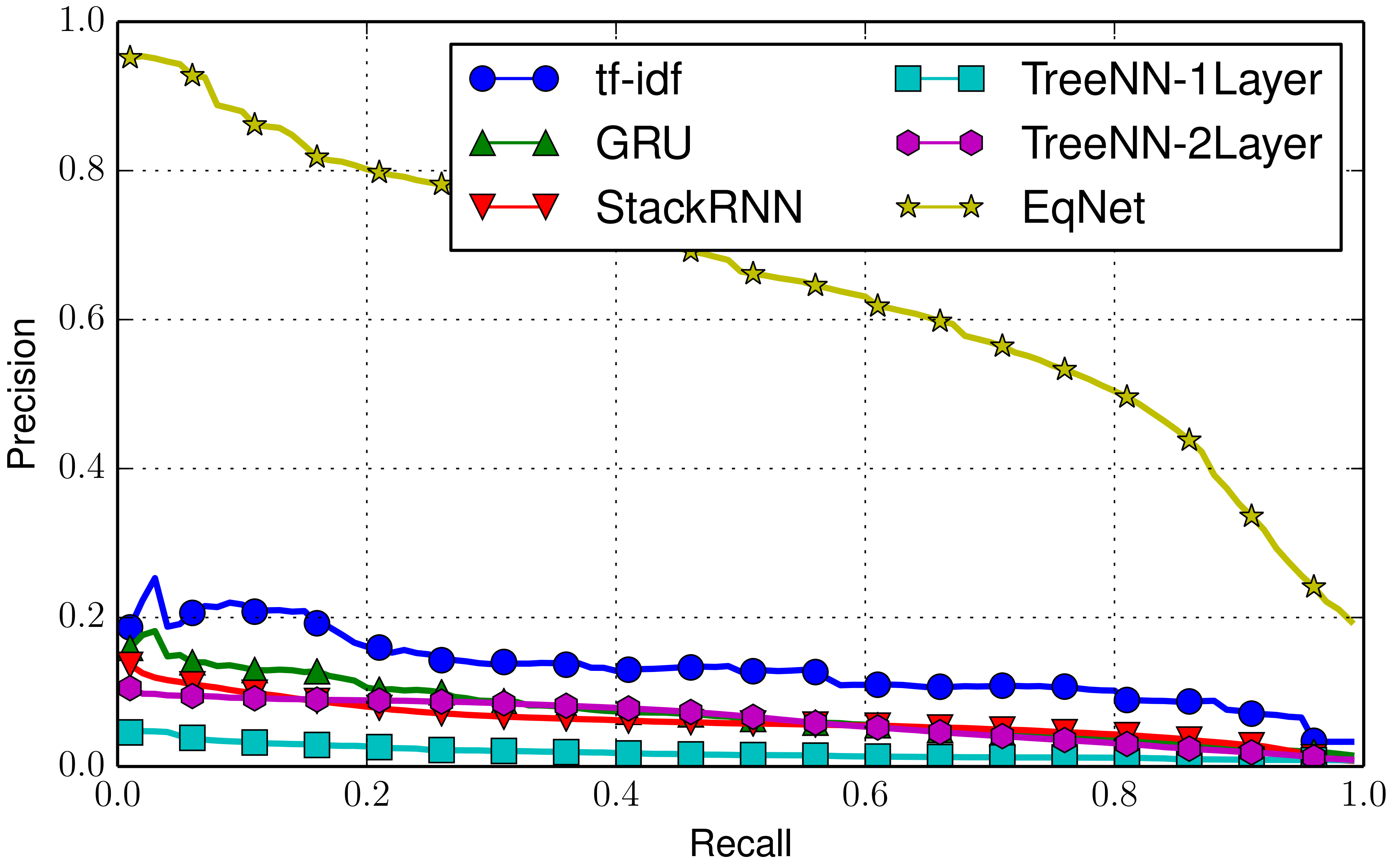
Seen equivalence classes

Equivalent expressions to the queries were in training set



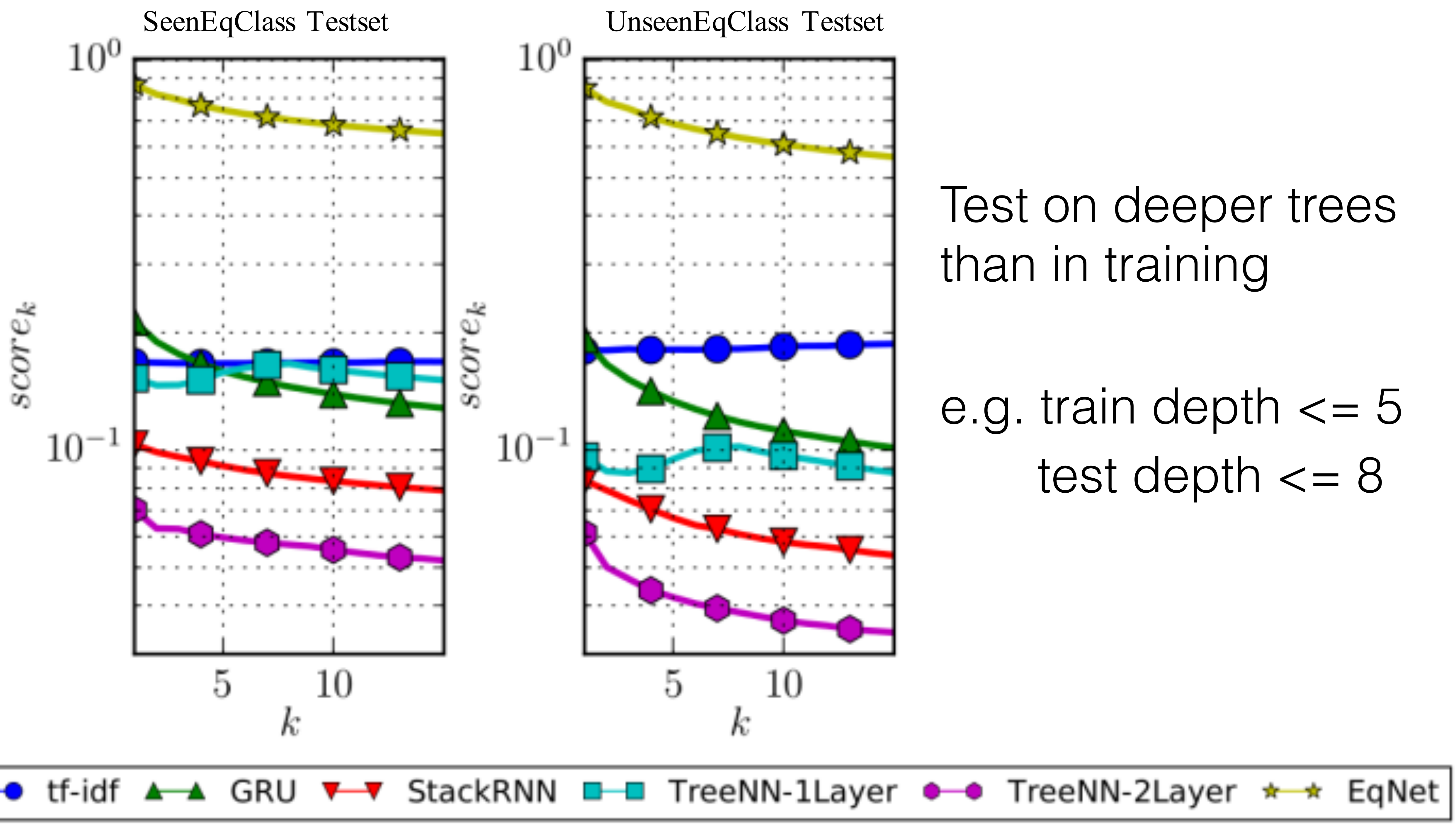
Unseen equivalence classes

Zero shot learning. No training examples of equivalent expressions.



EqNet performance on seen and unseen is similar!

Learned compositionality?

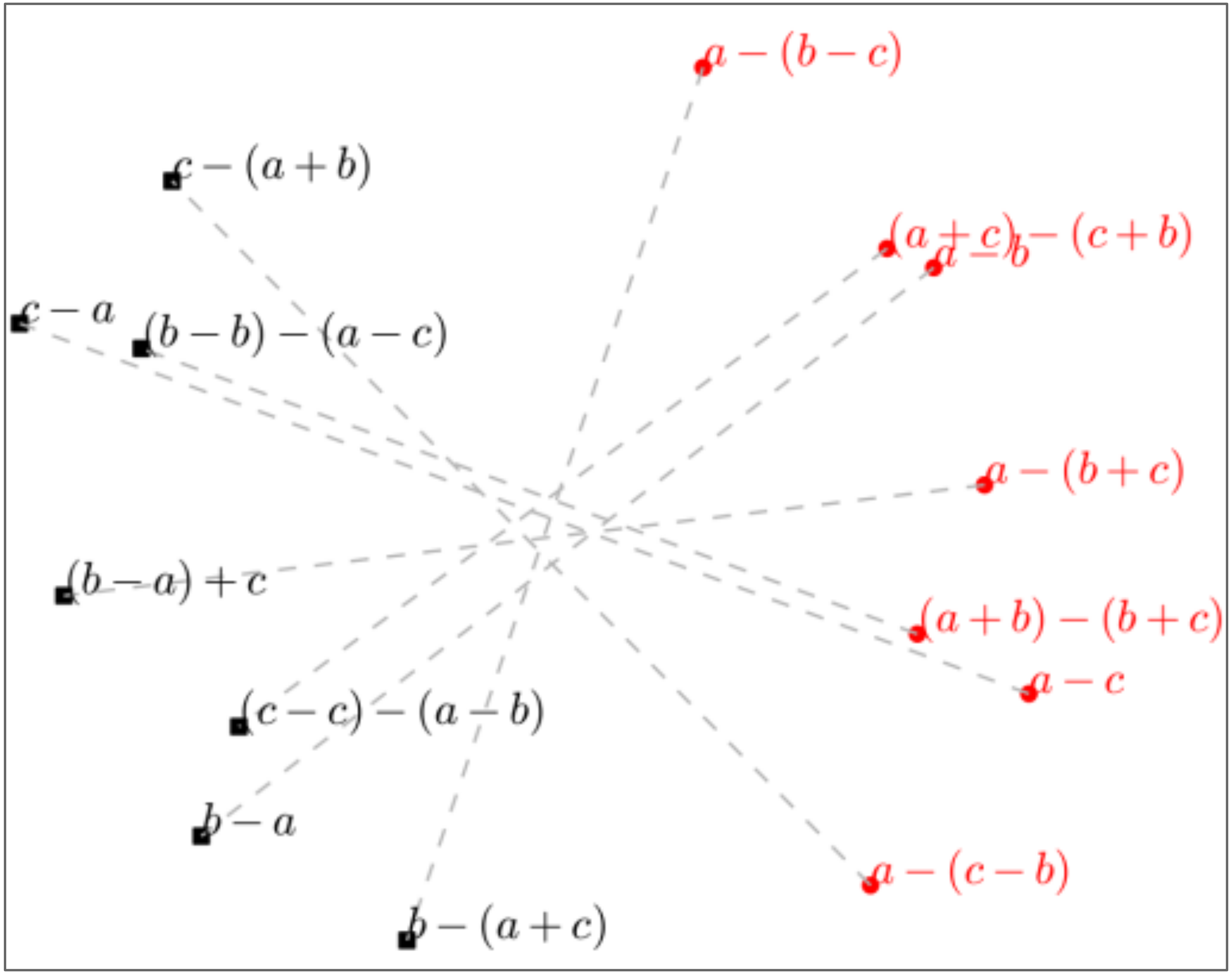


Test on deeper trees than in training

e.g. train depth <= 5
test depth <= 8

Visualizing polynomials

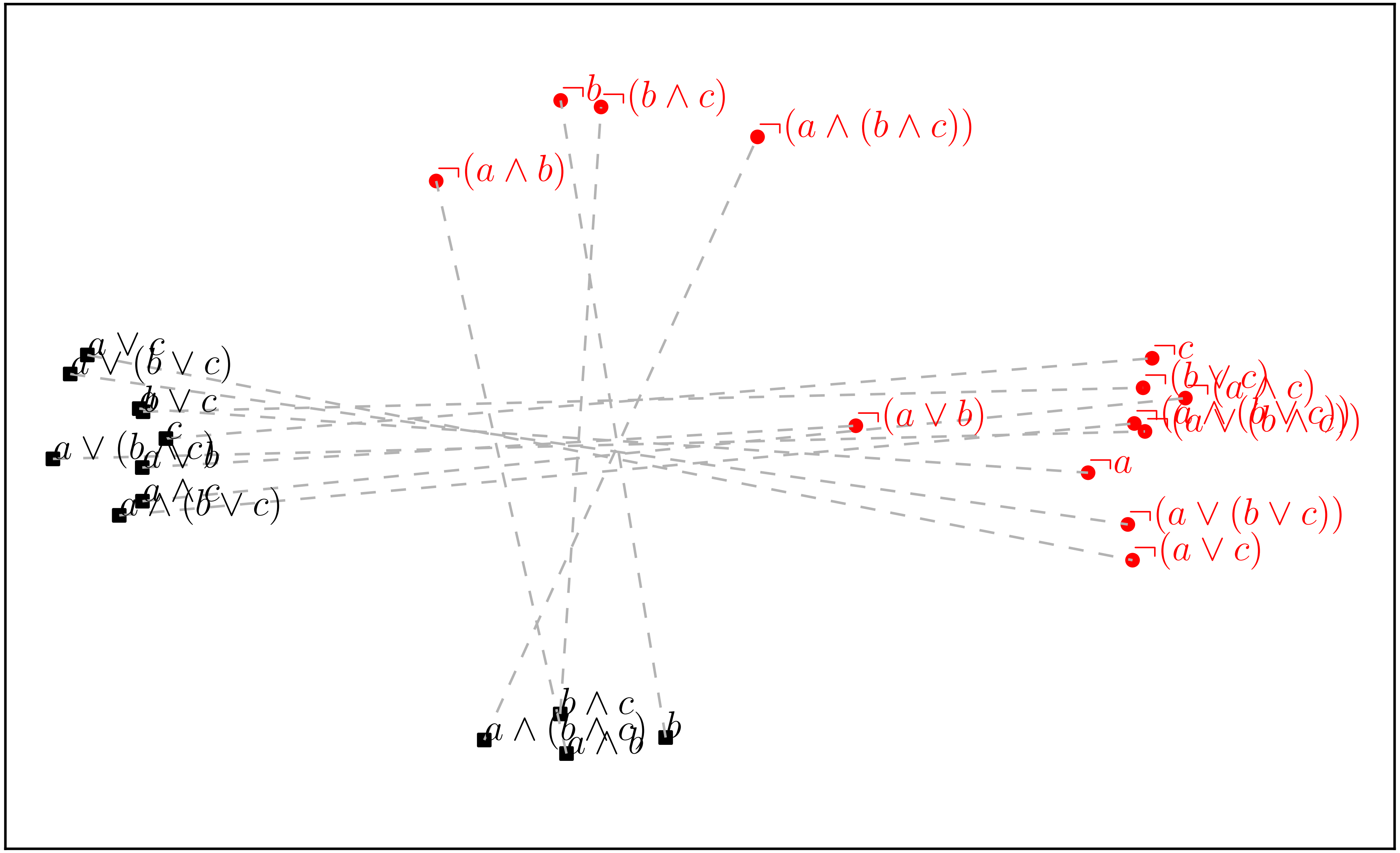
`multivariatePolynomial2vec?`



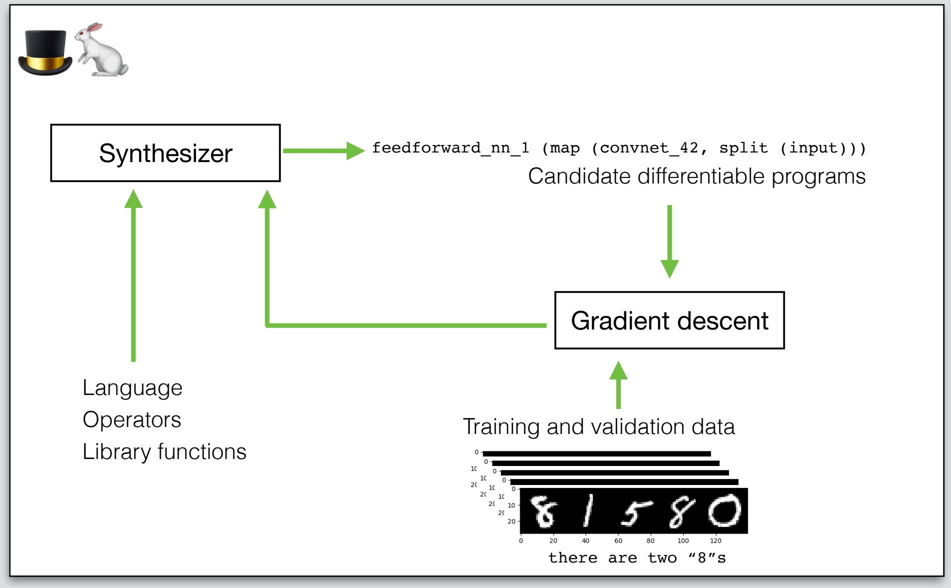
PCA visualization of semVecs

Visualizing boolean expression

`booleanExpression2vec?`



PCA visualization of semVecs



Synthesizer

```
feedforward_nn_1 (map (convnet_42, split (input)))
```

Candidate differentiable programs

Gradient descent

Language
Operators
Library functions

Training and validation data

```
there are two “8”s
```

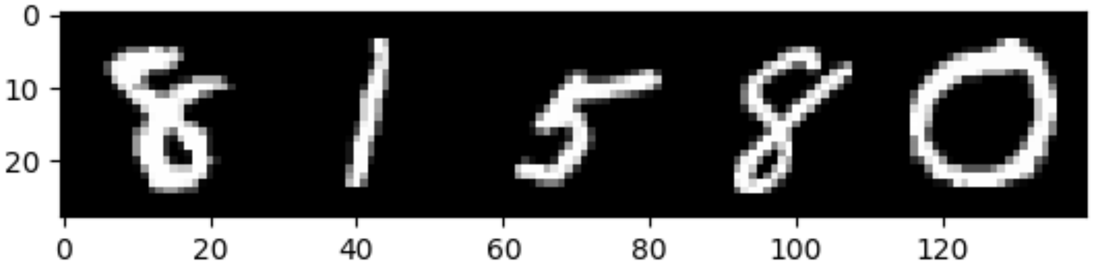
Synthesis of Differentiable Functional Programs for Lifelong Learning

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

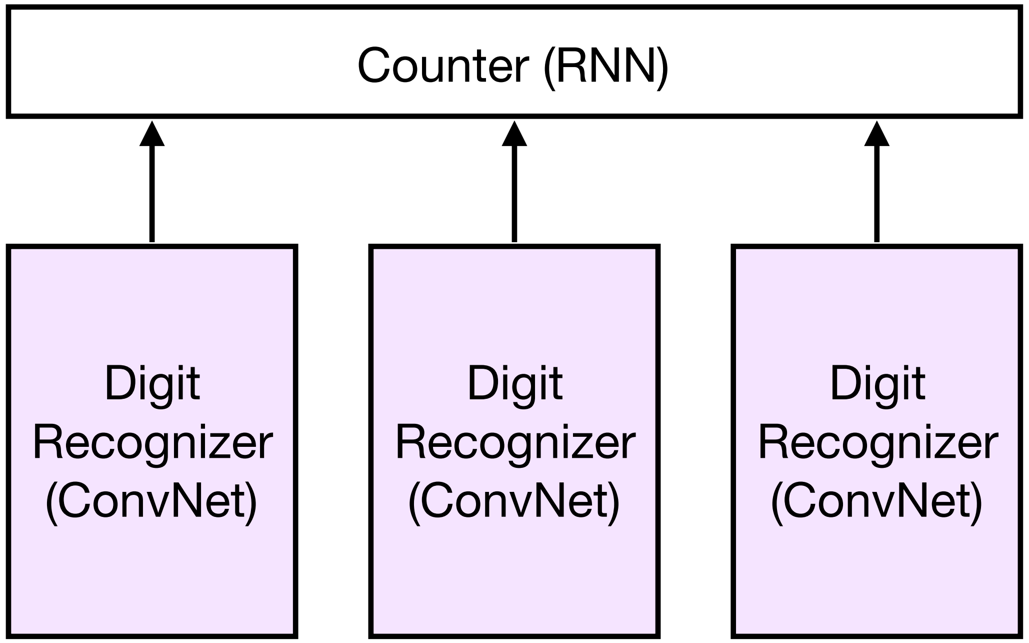
http://bit.ly/sutton-dllc

High level transfer

Task 1
Count digits



`there are two “8”s`



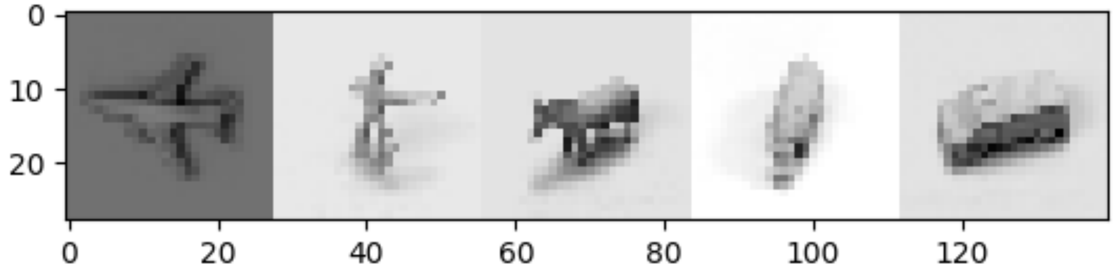
Counter (RNN)

Digit Recognizer (ConvNet)

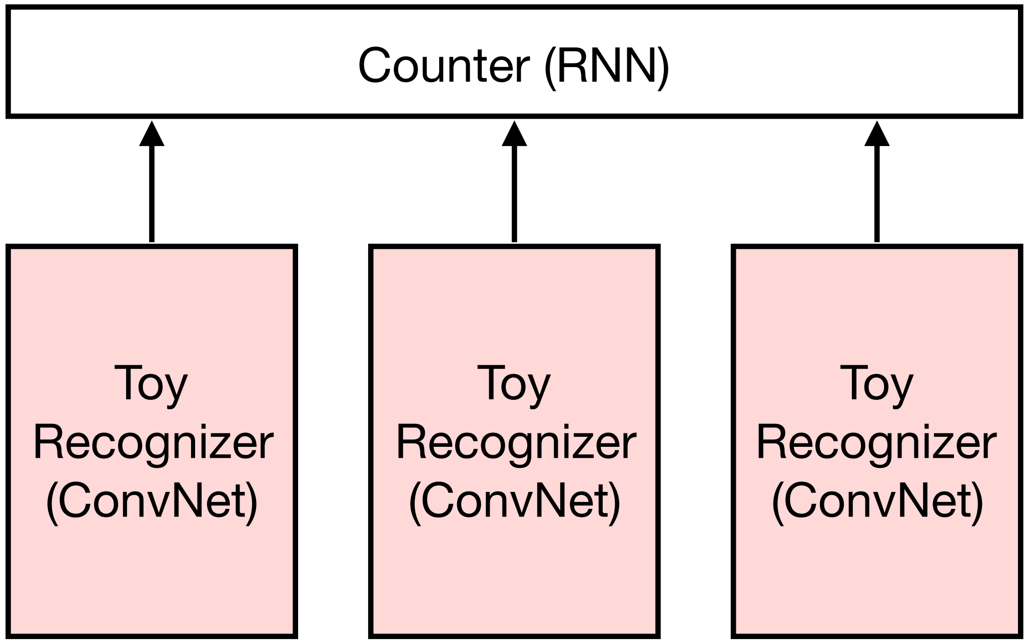
Digit Recognizer (ConvNet)

Digit Recognizer (ConvNet)

Task 2
Count toys



`there is one “toy airplane”`
`(and why don’t I have two?)`



Counter (RNN)

Toy Recognizer (ConvNet)

Toy Recognizer (ConvNet)

Toy Recognizer (ConvNet)

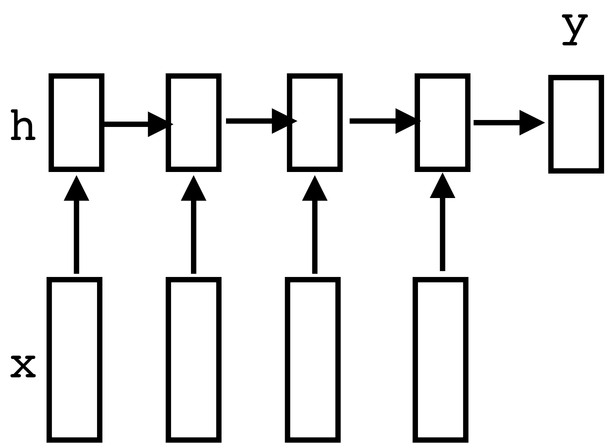
Reusing early layers not sufficient!

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

100 neural architectures, 1 weird trick

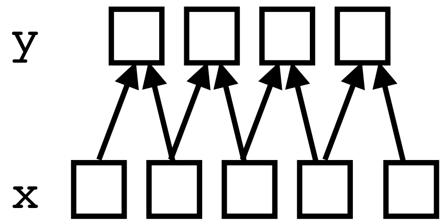
Functional programming

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)
```

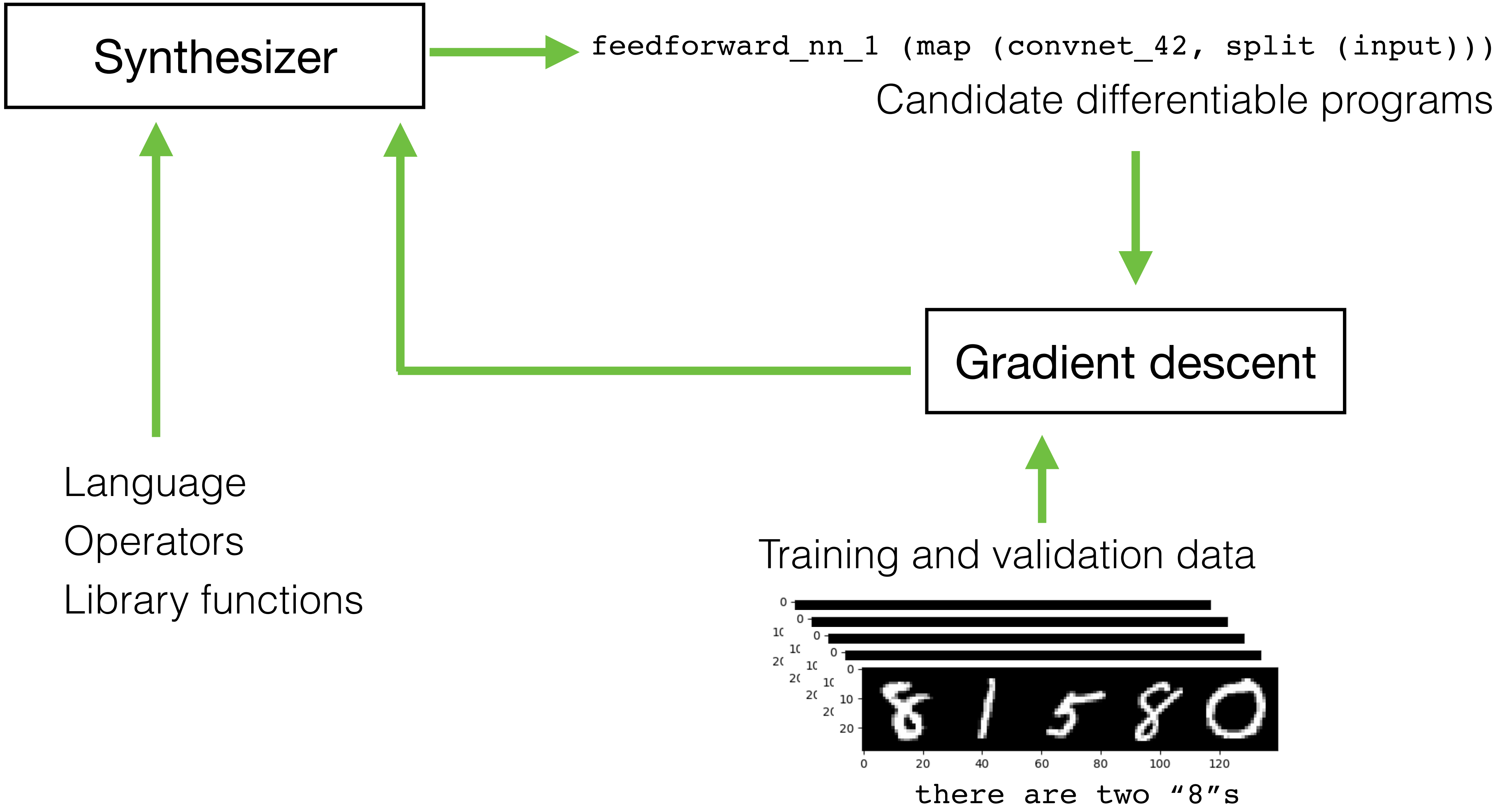
Graph convolutions

```
gconv (nn_kernel_0)
```

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

Houdini 🎩🐇

Synthesis of Differentiable Functional Programs



Synthesizer

```
feedforward_nn_1 (map (convnet_42, split (input)))
```

Candidate differentiable programs

Gradient descent

Language

Operators

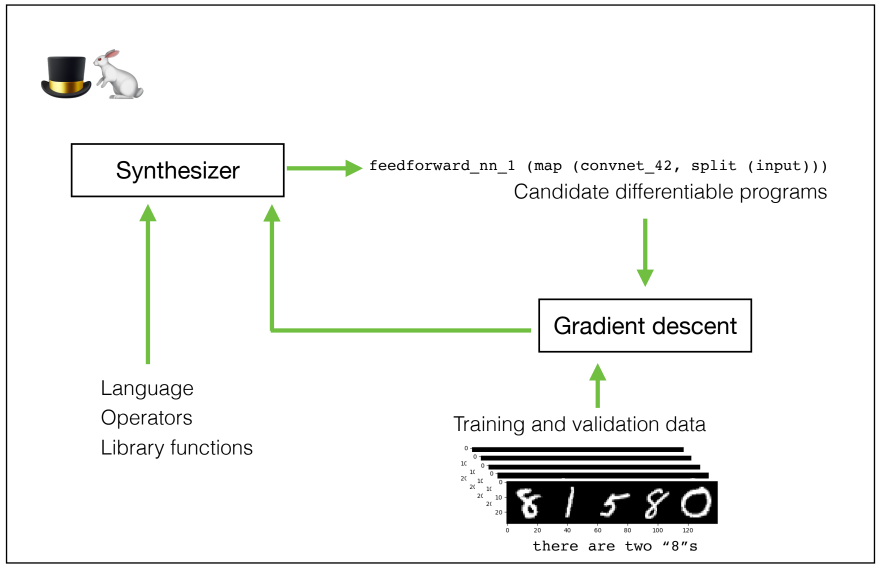
Library functions

Training and validation data

`there are two "8"s`

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

Synthesis for Lifelong Learning



Synthesizer

```
feedforward_nn_1 (map (convnet_42, split (input)))
```

Candidate differentiable programs

Gradient descent

Language
Operators
Library functions

Training and validation data

there are two “8”s

```
fold(conv (nn_kernel_2)
       o conv (nn_kernel_1)
       o conv (nn_kernel_0))
```

Best program

Freeze parameters

nn_kernel_2

nn_kernel_1

nn_kernel_0

Library

List reversal module

Digit Recognizer (ConvNet)

Toy Recognizer (ConvNet)

Street Sign Recognizer

Summing (RNN)

Co

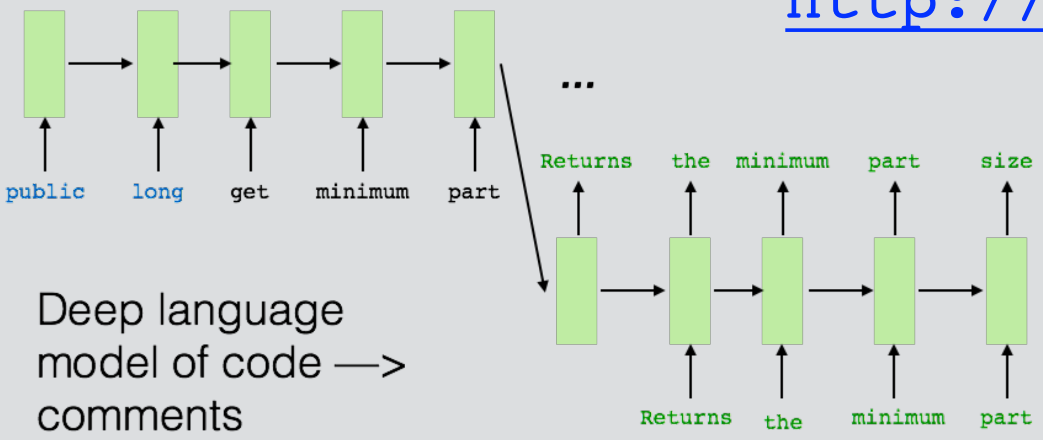
Learning to count with 10% of the data by **high-level transfer** from another image domain

Deep Learning, Language, and Code:

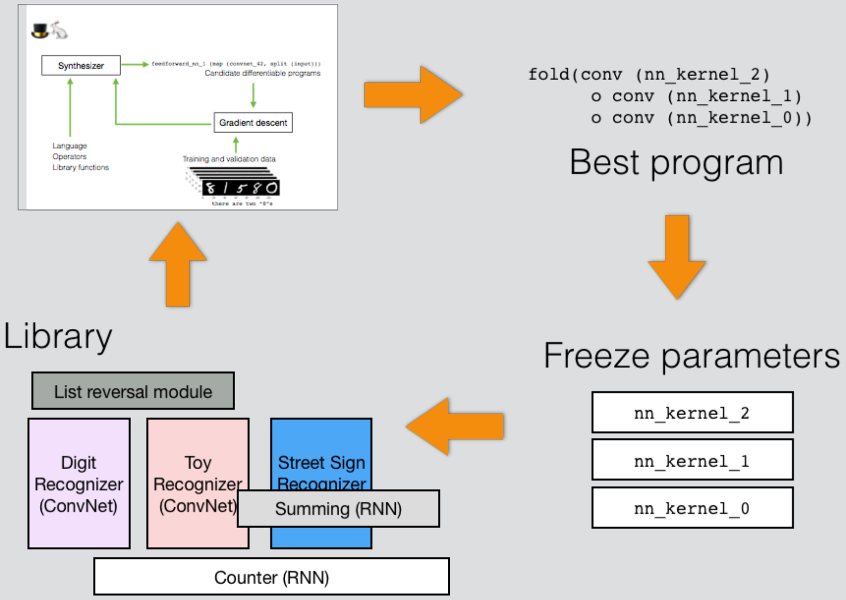
From Methodology to Applications and Back

Charles Sutton, University of Edinburgh

http://bit.ly/sutton-dllc



Deep language model of code —> comments



Synthesizer
Candidate differentiable programs
Gradient descent
Language
Operators
Library functions
Training and validation data

```
fold(conv (nn_kernel_2)
      o conv (nn_kernel_1)
      o conv (nn_kernel_0))
```

Best program

Freeze parameters

nn_kernel_2

nn_kernel_1

nn_kernel_0

Library

List reversal module

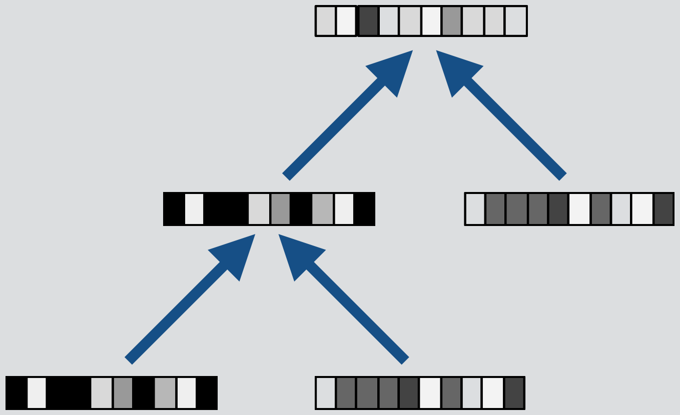
Digit Recognizer (ConvNet)

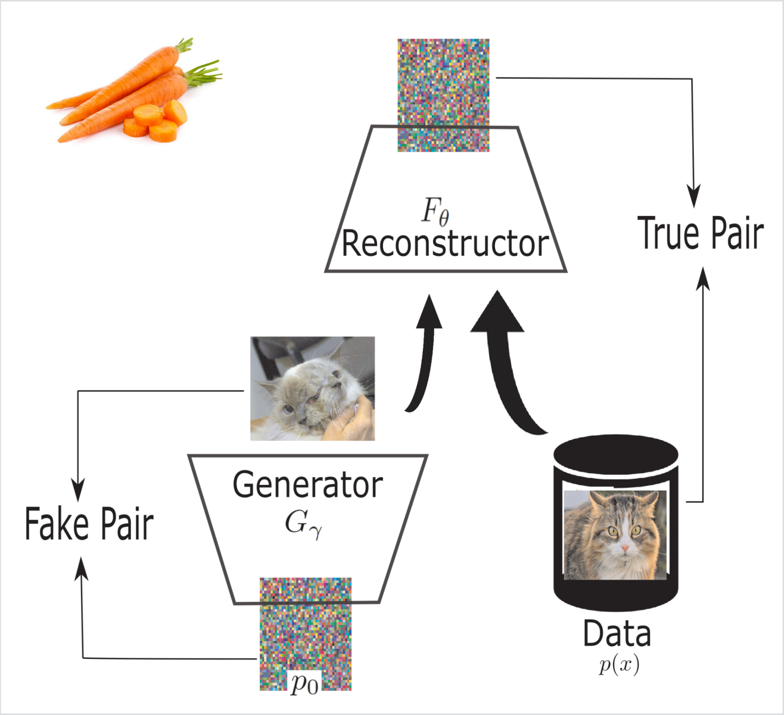
Toy Recognizer (ConvNet)

Street Sign Recognizer

Summing (RNN)

Counter (RNN)





Miltiadis Allamanis

Annie Louis

Lazar Valkov

Akash Srivastava

Santanu Dash

Pushmeet Kohli

Pankajan Chantirasagaran

Earl Barr

Dipak Chaudhari

Swarat Chaudhuri

Michael Gutmann

Chris Russell

Thanks!

Microsoft® Research

EPSRC
Engineering and Physical Sciences Research Council

The Alan Turing Institute