Learning for Programs:

Connecting code, statistics, semantics, and language

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

http://bit.ly/sutton-learning4programs

THE UNIVERSITY of EDINBURGH
informatics

The Alan Turing Institute

Microsoft Research

EPSRC
Engineering and Physical Sciences Research Council

Source code is a means of human communication

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

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

Over **20 billion** lines of open source code online

Implicit knowledge about how to write code

- Uses common libraries
- Avoids common bugs
- Easy to read and maintain
- How to write good documentation, tests

Machine learning to find these implicit patterns

Suggest them during development

To automatically repair and test code

To guide program analysis

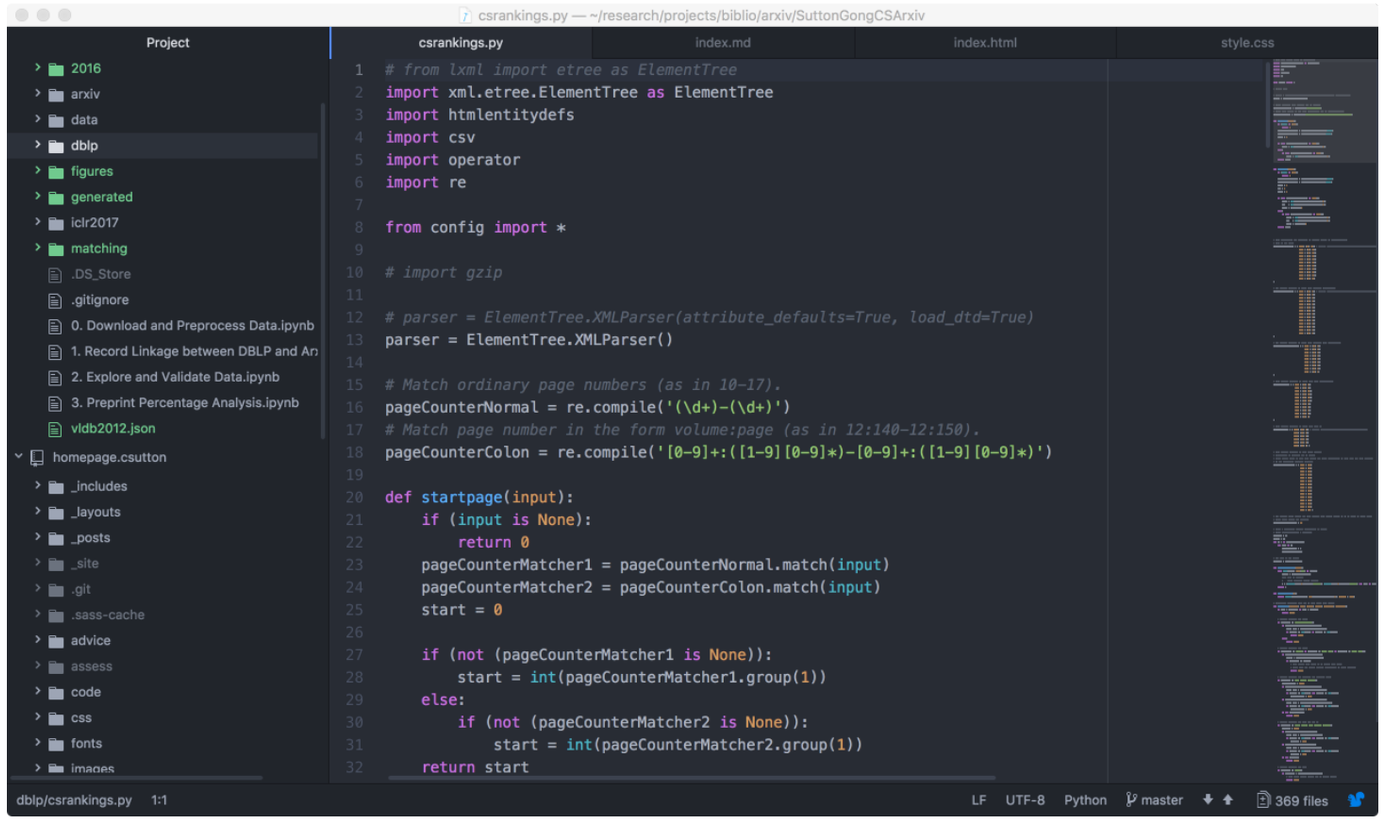
Source: Black Duck

DEI: Development Environment with Intelligence

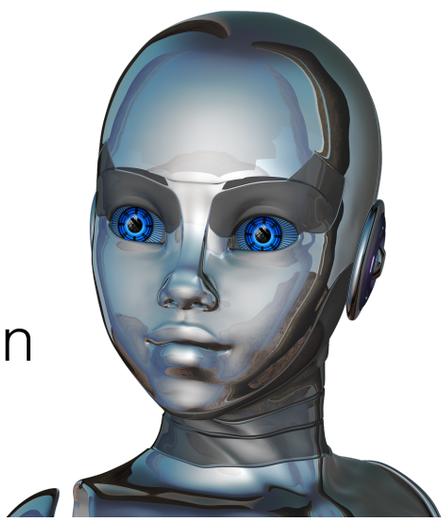
AI support for the full software lifecycle

Cyberpair programming

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



Suggestions on: Coding style
Bug fixes
Documentation
Debugging



Naturalize

[Allamanis, Barr, Bird, Sutton; FSE 2014]

junit/src/test/java/junit/tests/runner/TextRunnerTest.java

```
public class TextRunnerTest extends TestCase {
  void execTest(String testClass, boolean success) throws Exception {
            ...
            InputStream i = p.getInputStream();
            while ((i.read()) != -1);
            ...
        }
        ...
}
```

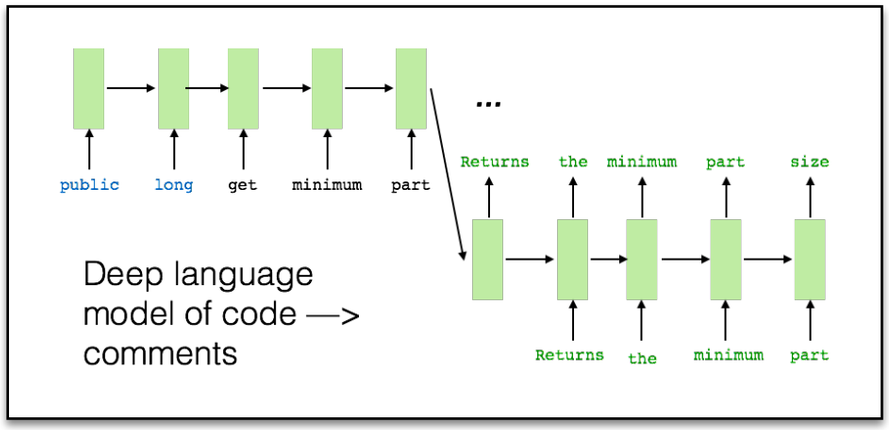
Suggest alternate names

```
input
inputStream
is
stream
```

Score by ngram model and threshold

```
input (81.93%)
```

$$P(t_0 \dots t_M) = \prod_{m=0}^{M} P(t_m | t_{m-1} \dots t_{m-n+1})$$



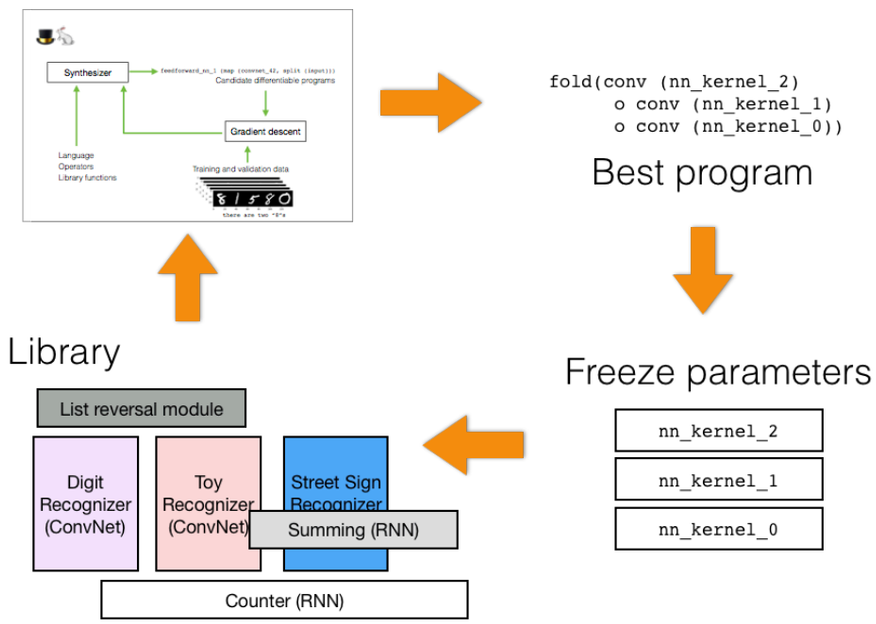
Documentation

```
Item goodItem = null;
foreach (item in collection) {
    if ( SomethingIsTrueOfAll(item.Children) ) {
            goodItem = item;
            break;
        }
}
```

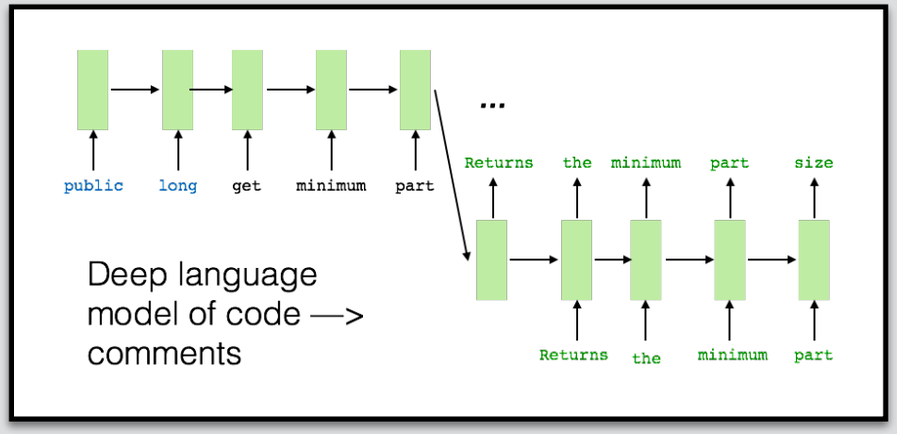
Reads from all items in collection

Writes to unitary variable

Semantic Patterns in Software



Program synthesis to enable high-level transfer



Finding Uninformative Comments

http://bit.ly/sutton-learning4programs

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

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

Let’s discourage repetitive comments!

The Ugly

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

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

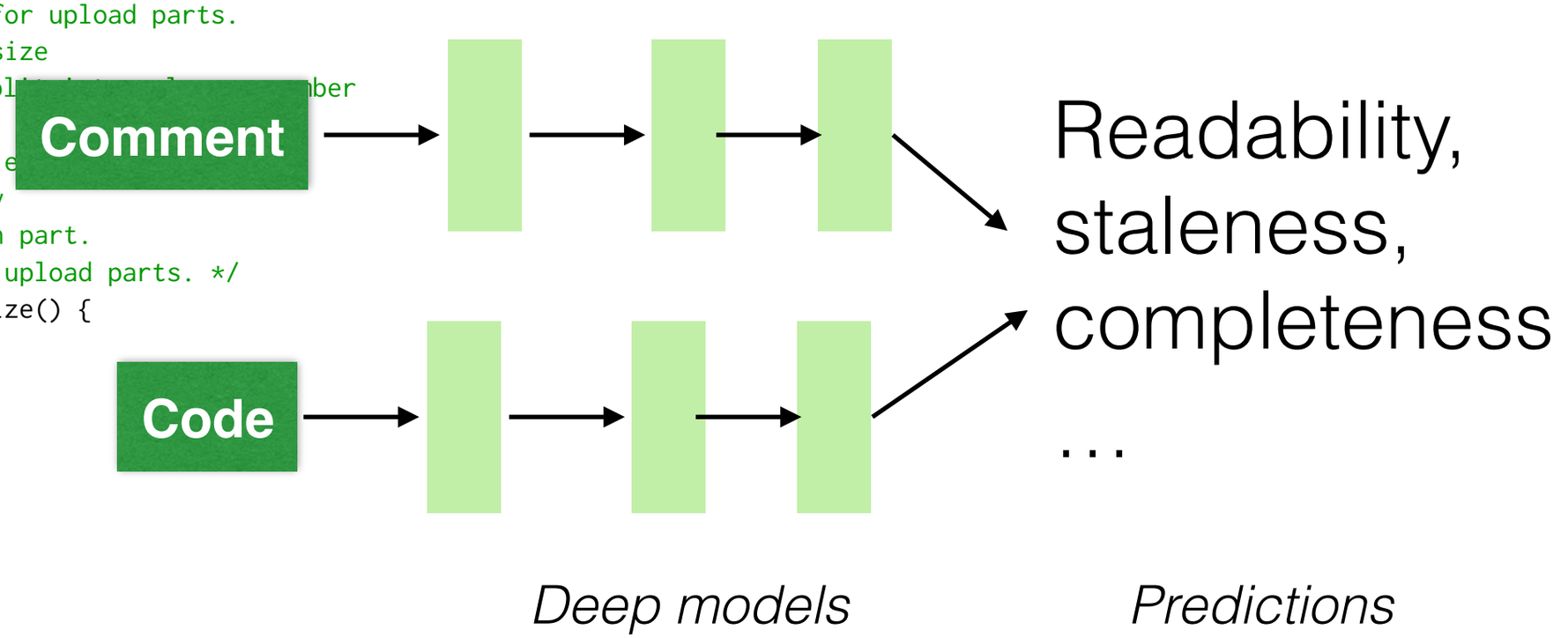
- Bad comments cause bloat
- Good comments take time
- Comments go stale

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                    mber
     of smaller parts. Setting
this value too low has a negative e
     speeds, causing extra latency
and network communication for each part.
@return The minimum part size for upload parts. */
public long getMinimumUploadPartSize() {
  return minimumUploadPartSize;
}
```



Comment entailment problem

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

Comment sentence

```
6  public long getMinimumUploadPartSize() {
7    return minimumUploadPartSize;
8  }
```

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?

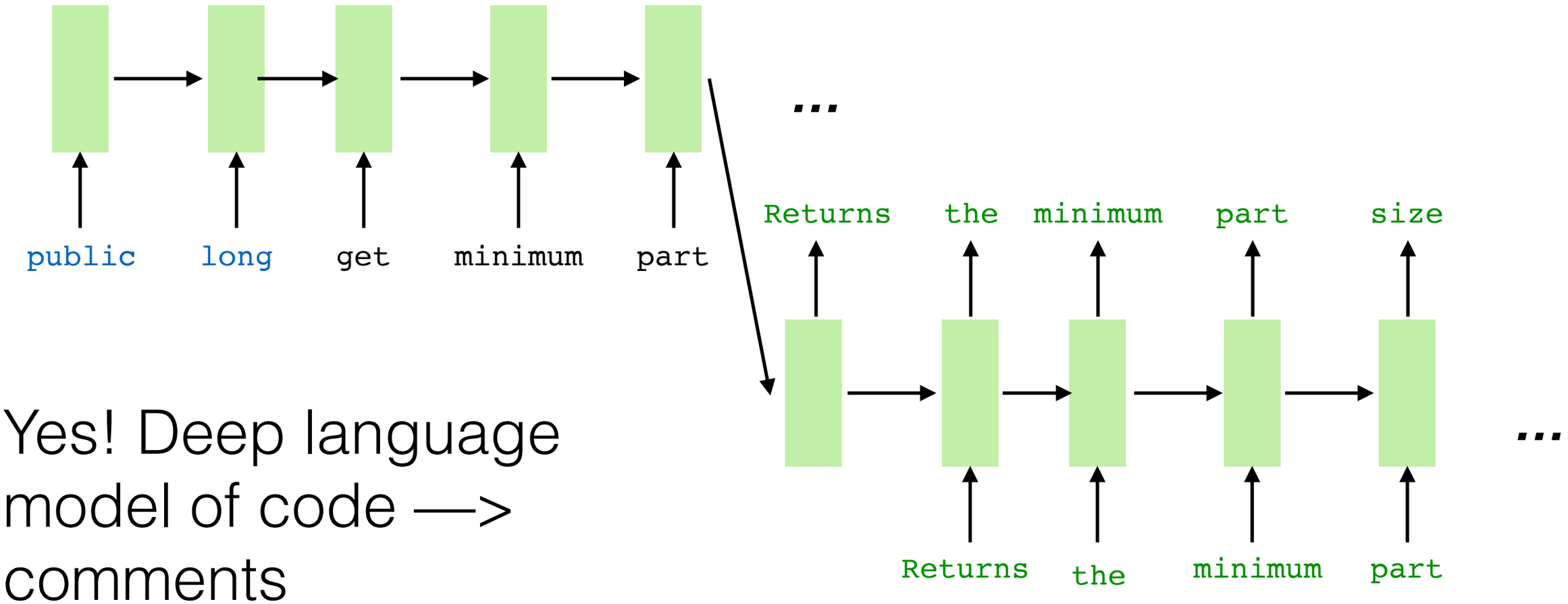
Academic: Entailment is **good** because the point of comments is to explain the code, right?

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

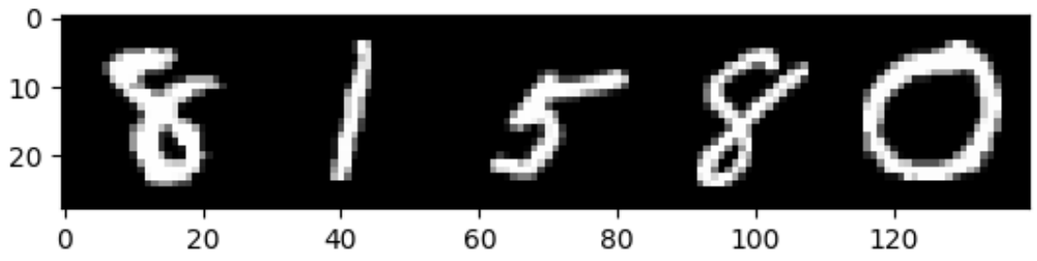
Model	Perplexities		
	Train	**Valid**	**Test**
LM	7.80	10.34	9.87
s2s-signature	5.70	6.90	8.26
s2s-begin-end	3.44	4.18	5.31
s2s-identifier	4.50	5.34	6.00
LM English newswire			58

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

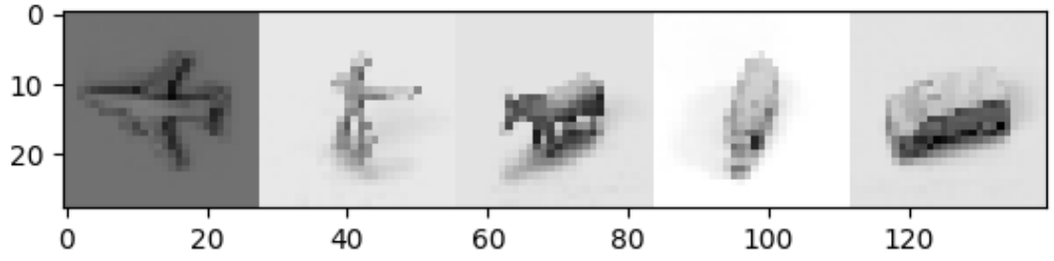
High level transfer

Task 1
Count digits

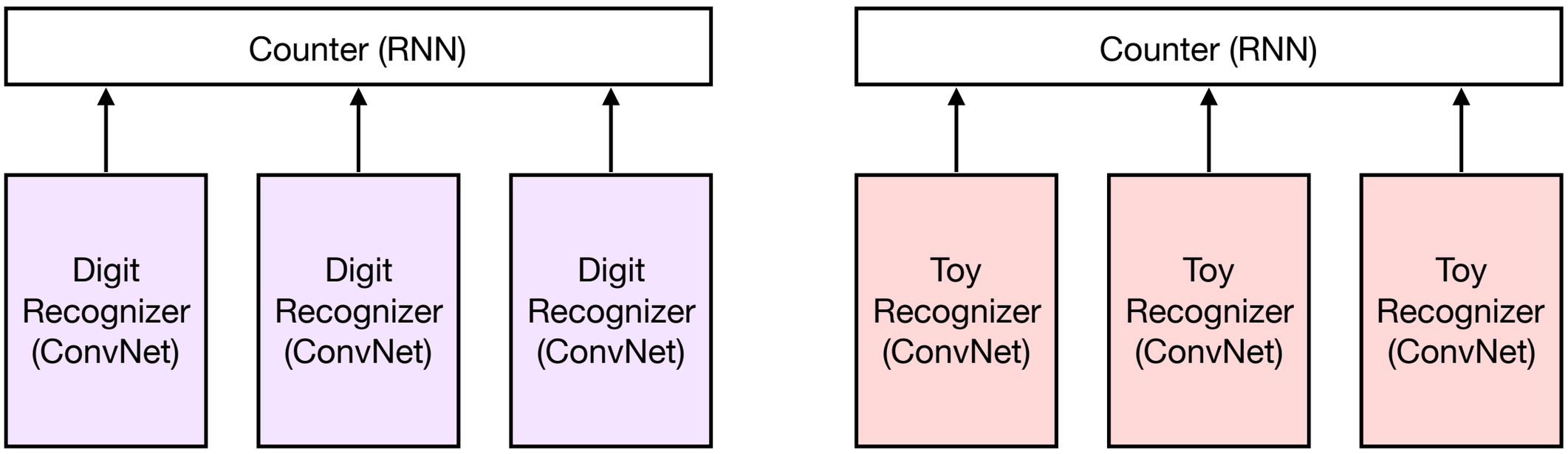


there are two “8”s

Task 2
Count toys



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



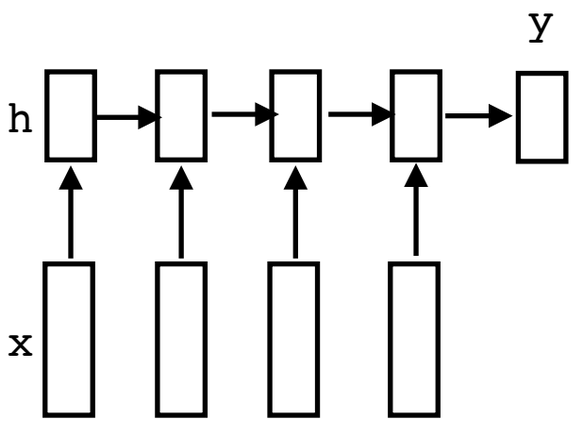
Reusing early layers not sufficient!

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

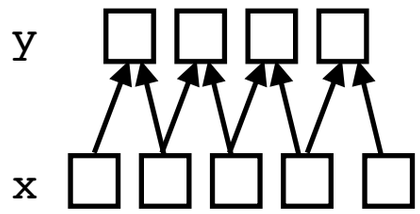
100 neural architectures, 1 weird trick

Functional programming

Recurrent neural network



1-layer ConvNet




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

multiple filters? change this

```
fold (rnn_1step)
```

```
conv (nn_kernel_0)
```

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

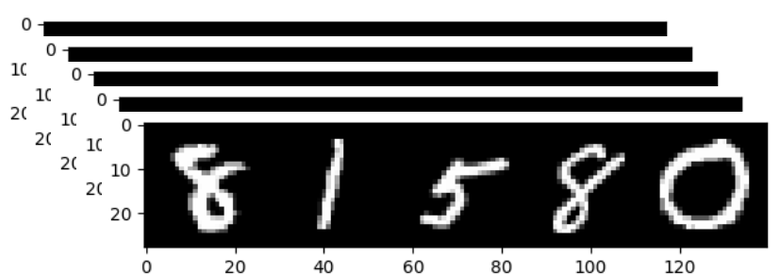
```
feedforward_nn_1 (map (convnet_42, split (input)))
```

Candidate differentiable programs

Gradient descent

Language
Operators
Library functions

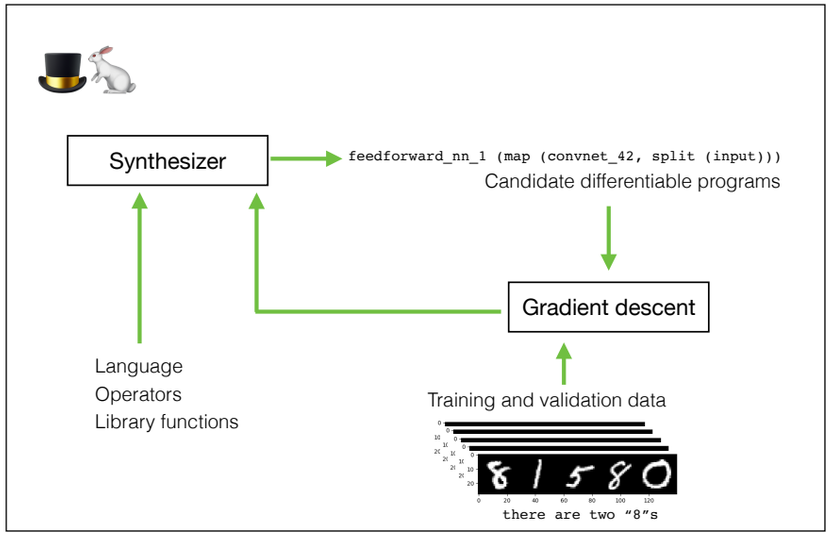
Training and validation data



there are two “8”s

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

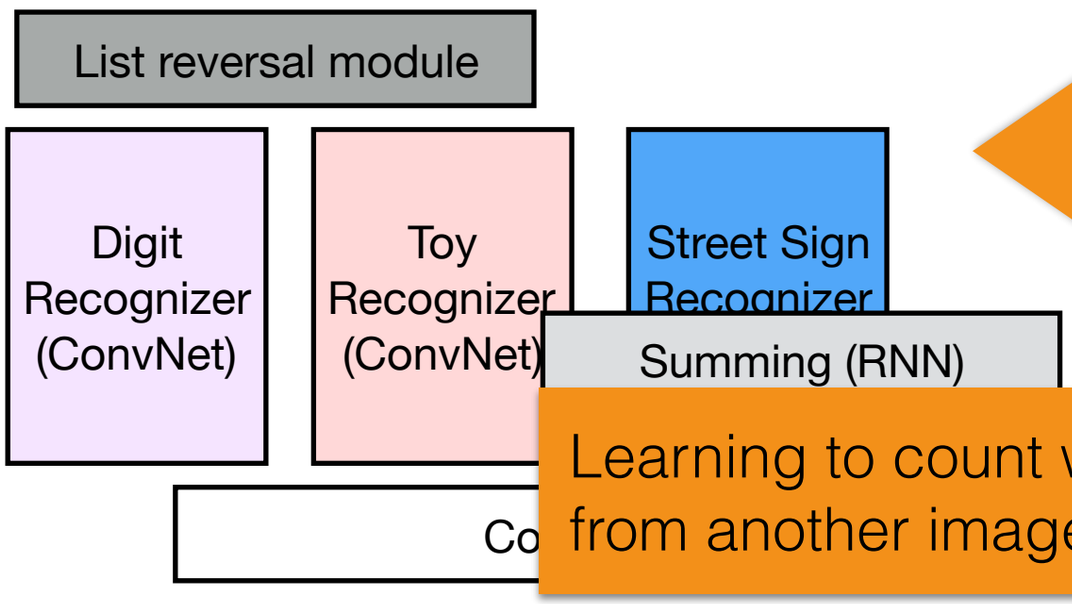
Synthesis for Lifelong Learning



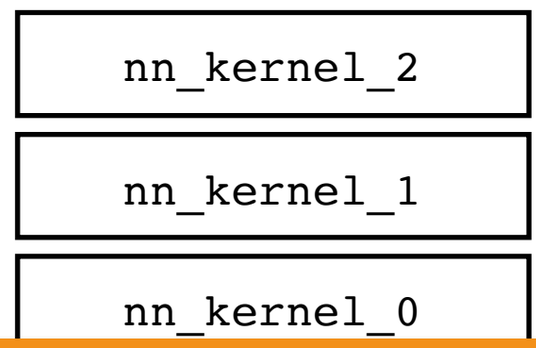

```
fold(conv (nn_kernel_2)
     o conv (nn_kernel_1)
     o conv (nn_kernel_0))
```

Best program

Library



Freeze parameters



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

```
Item goodItem = null;
foreach (item in collection) {
    if ( SomethingIsTrueOfAll(item.Children) ) {
            goodItem = item;
            break;
        }
}
```

Reads from all items in collection

Writes to unitary variable

Finding Semantic Idioms in Code

[Allamanis, Barr, Bird, Devanbu, Marron, Sutton, 2018]

http://bit.ly/sutton-learning4programs

A story in five acts

Stupid

```
Item goodItem = null;
foreach (item in collection) {
    foreach ( child in item.Children ) {
        if (!  SomethingIsTrue (child) ) {
            //  I don't want item
            continue outer_loop;
        }
    }
    goodItem = item; break;
}
```

Idiomatic

```
Item goodItem = null;
foreach (item in collection) {
    if ( SomethingIsTrueOfAll(item.Children) ) {
            goodItem = item;
            break;
        }
}
```

Adventures of LINQ

```
Item goodItem = null;
foreach (item in collection) {
    if (item.Children.All (child => SomethingIsTrue (child) ) {
            goodItem = item;
            break;
        }
}
```

Guru

```
Item goodItem = collection.Where(item => item.Children.All(child => SomethingIsTrue(child))
                          .FirstOrDefault();
```

This is an idiom

Recurring pattern

```
Item goodItem = null;
foreach (item in collection) {
    if ( SomethingIsTrueOfAll(item.Children) ) {
            goodItem = item;
            break;
        }
}
```

Reads from all items in collection

Writes to unitary variable

Semantic information

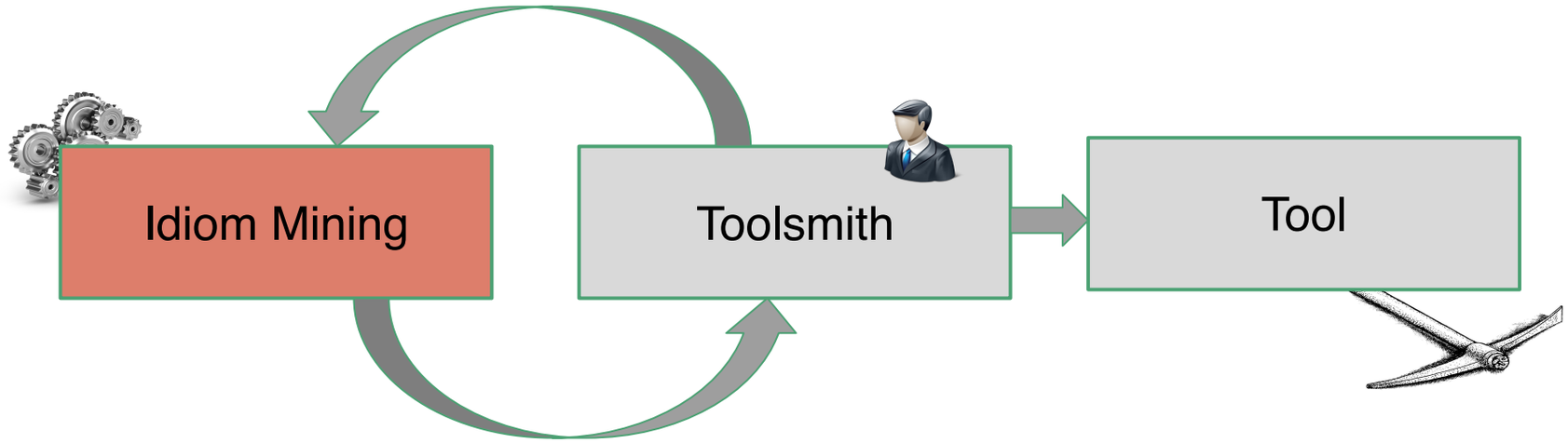
Access patterns, reads / writes to collections

Our goal: Mine these from big code

Semantic idioms

Find patterns not just
in what programmers *say*
but in what they *mean*

Uses of semantic idioms



Mine and **prioritize** patterns

Data-driven design of

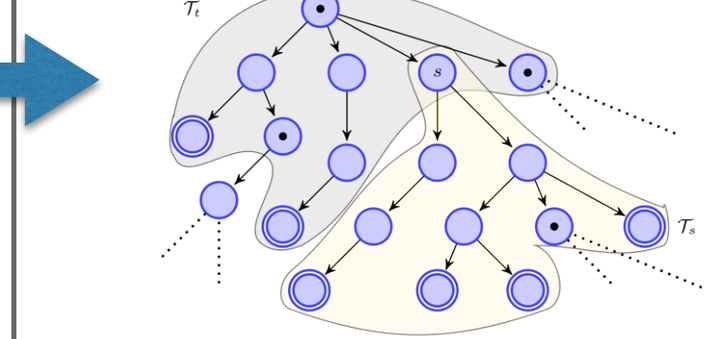
Refactoring tools
Verification engines
Synthesis engines
APIs
Programming languages

Syntactic idioms in code

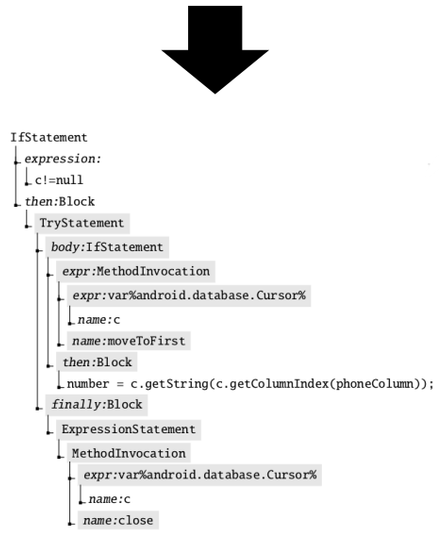
HAGGIS

Holistic, Automatic Gathering of Grammatical Idioms from Software

Corpus of ASTs



Learn pTSG



Extract & Filter Production Rules

Idioms

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

```
Location.distanceBetween(
    $(Location).getLatitude(),
    $(Location).getLongitude(),
    $...);
```

```
Document doc=Jsoup.connect(URL).
    userAgent("Mozilla").
    header("Accept","text/html").
    get();
```

```
Toast.makeText(this,
    $stringLit,Toast.LENGTH_SHORT)
      .show()
```

```
while (($(String) = $(BufferedReader).
    readLine()) != null) {
  $BODY$
}
```

[Allamanis and Sutton, FSE 2014]

Mined Idioms (General Java)

Iterate through the elements of an `Iterator`

```
for (Iterator iter=$methodInvoc;
    iter.hasNext(); )
  {$BODY$}
```

Looping through lines from a `BufferedReader`

```
while (($(String) = $(BufferedReader).
    readLine()) != null) {
  $BODY$
}
```

Creating a logger for a class

```
private final static Log $name=
  LogFactory.getLog($type.class);
```

Defining a String constant

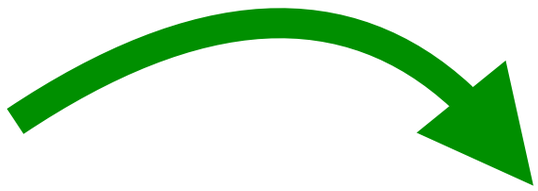
```
public static final
  String $name = $StringLit;
```

[Allamanis and Sutton, FSE 2014]

Loop coiling

Semantic abstraction

```
foreach(int x in arr) {
  var d = new Obj(x);
  lst.add(d);
  if (x > 0) {
        count += 1;
  }
}
```



```
foreach(int x in arr) {
  write collection lst and read x
  if ( read x ) {
    write count 1;
  }
}
```

1. Straight line code —> uninterpreted functions
2. Abstract names, retaining internode references
3. Annotate with purity information

Sample idioms and concrete loops

Idiom

```
for(int 0=EXPR; 0<EXPR; INC(0))
  $REGION[UR(0);C^{S,E}R(1); URW(2)]
```

Concrete Loop

```
for (int i⁰ = 0; i⁰ < length; i⁰++)
  charsNeeded² += components¹[i⁰].Length;
```

Reduce with for loop

coverage: 11%

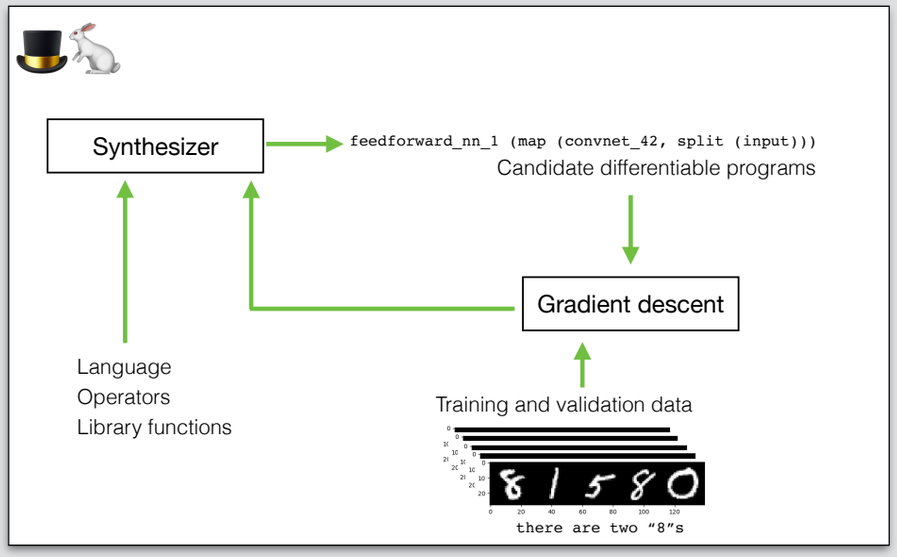
Sample idioms and concrete loops

Idiom

```
foreach (var $0 in EXPR) {
  $REGION[UR($0, $1); C^S RW($2); C^E RW($2) ]
}
```

Reduce with for-each and fixed input

coverage: 2%

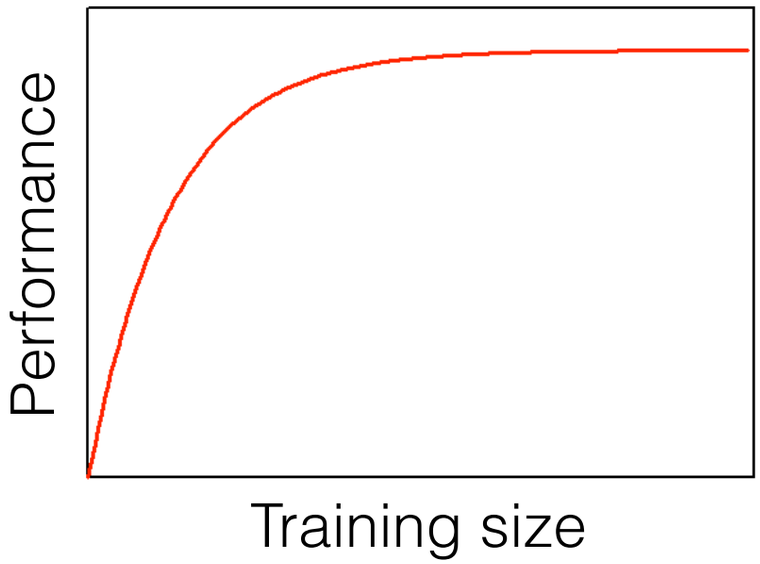


Synthesis of Differentiable Functional Programs for Lifelong Learning

http://bit.ly/sutton-learning4programs

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

The Problem with Learning



All learning curves stop.

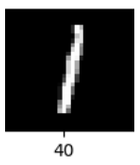
Why “data hungriness”?

Learning systems never get better at learning itself

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

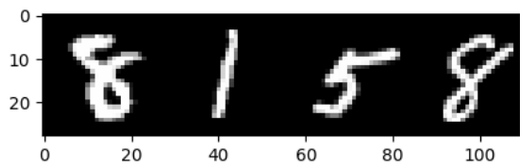
On series of tasks, accelerate learning by re-using learned structure

Task 1
Recognition



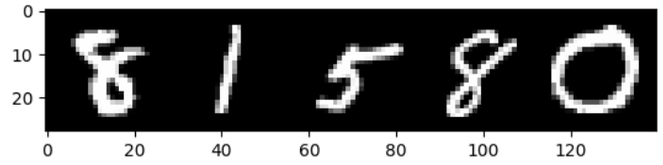
`that's a “one”`

Task 2
Counting



`I count two “8's”`

Task 3
Arithmetic



`The sum is 23`

For image sources, see: http://bit.ly/adventures-neurosymbolic

Neural libraries

[Gaunt, Brockschmidt, Kushman, Tarlow ICML 2017]

Digit Recognizer (ConvNet)

Toy Recognizer (ConvNet)

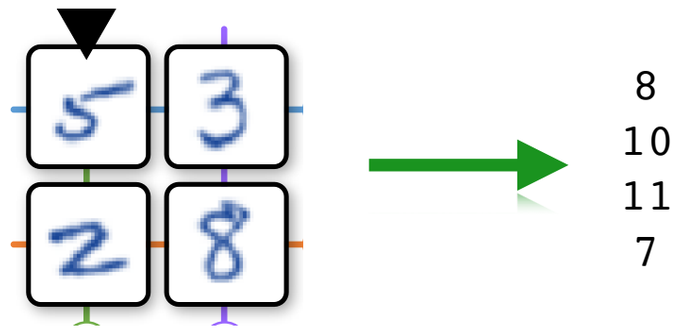
Street Sign Recognizer (ConvNet)

Networks from old tasks

Controller in differentiable PL

Differentiable interpreter

New task



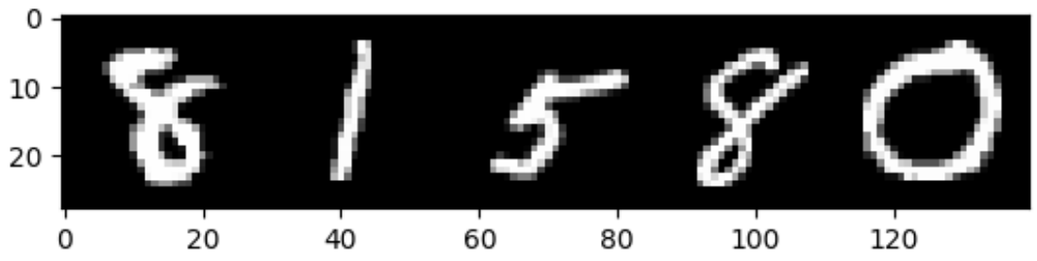
Learn programs plus perceptual networks

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

end-to-end gradient descent

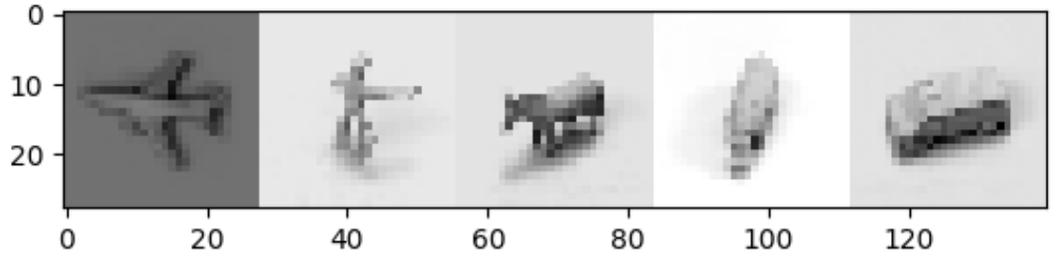
High level transfer

Task 1
Count digits

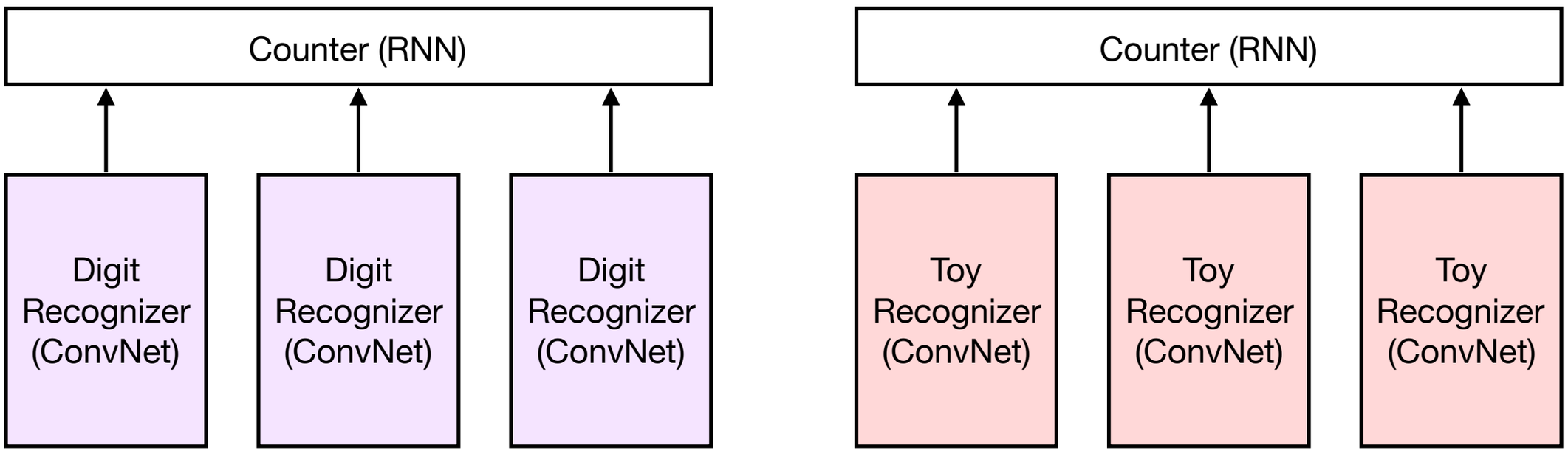


there are two “8”s

Task 2
Count toys



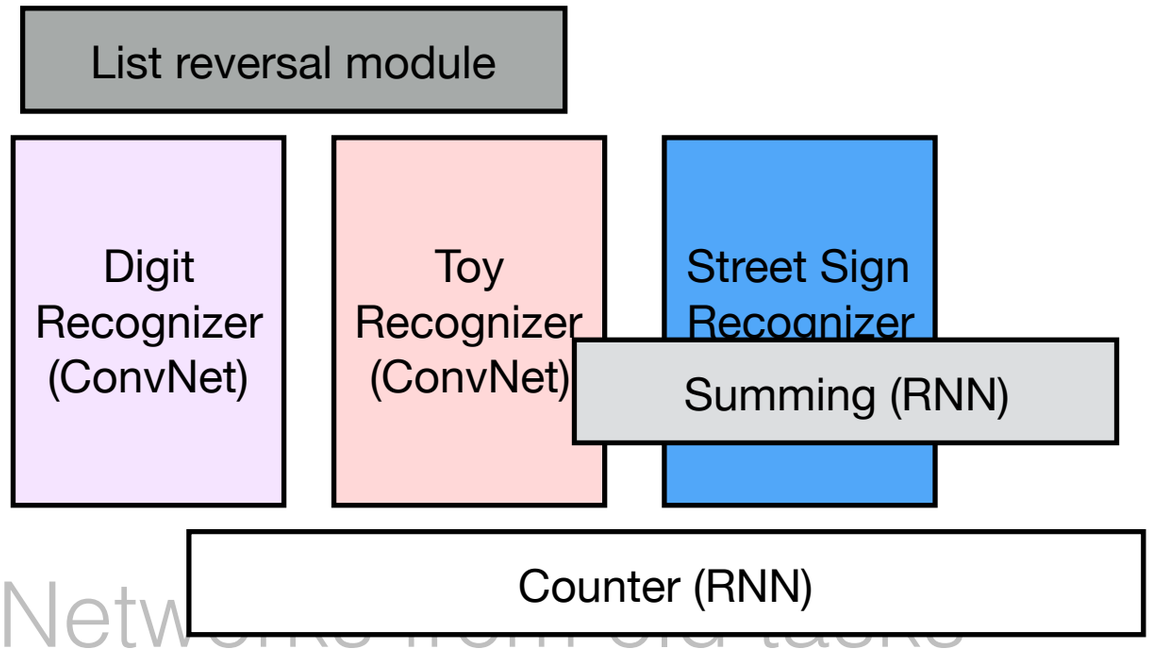
there is one “toy airplane”
(and why don’t I have two?)



Reusing early layers not sufficient!

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

High-level neural libraries



"High-level modules"

Other networks as input

Controller in differentiable PL

Differentiable interpreter

Problem: How to combine?

Learn programs plus perceptual networks

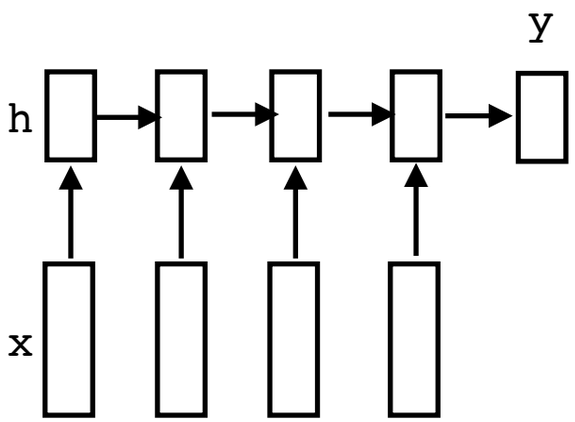
```
# initialization:
R0 = READ
# program:
R1 = MOVE_EAST
R2 = MOVE_SOUTH
R3 = SUM(R0, R1)
R4 = NOOP
return R3
```

end-to-end gradient descent

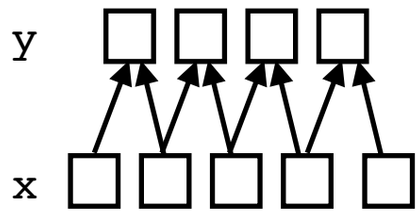
100 neural architectures, 1 weird trick

Functional programming

Recurrent neural network



1-layer ConvNet




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

multiple filters? change this

```
fold (rnn_1step)
```

```
conv (nn_kernel_0)
```

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

Synthesizer

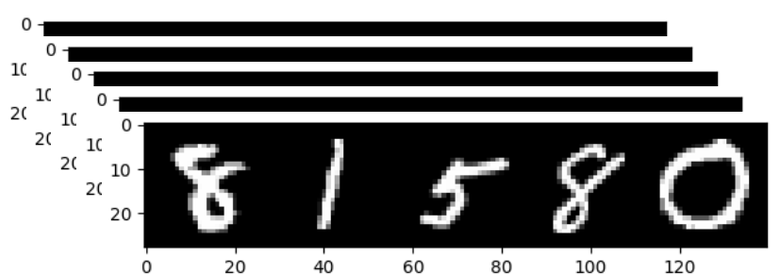
```
feedforward_nn_1 (map (convnet_42, split (input)))
```

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

Language

Types

guides the search space

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

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

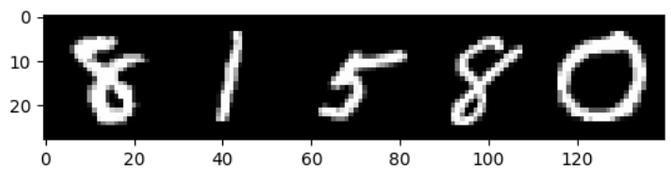
“neural functions”, e.g., two-layer ff network

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  >>
```

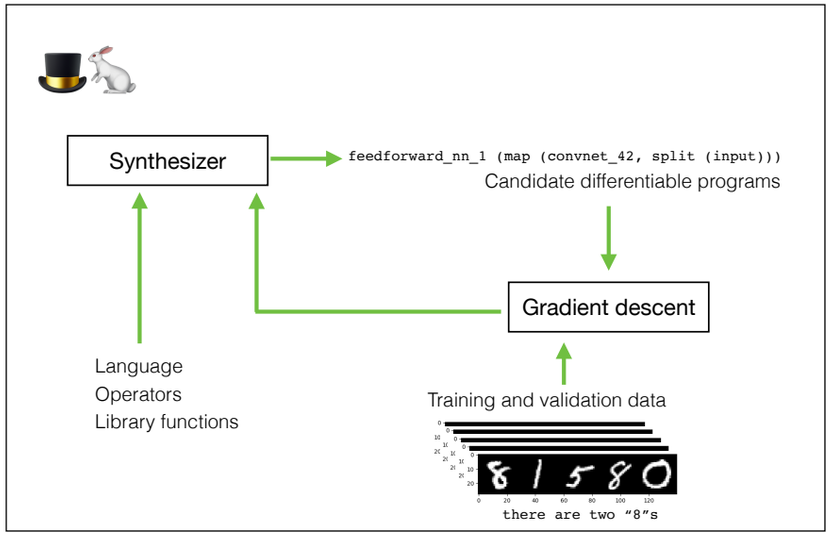
Partial program with type

Expand using grammar

`map` ✗

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

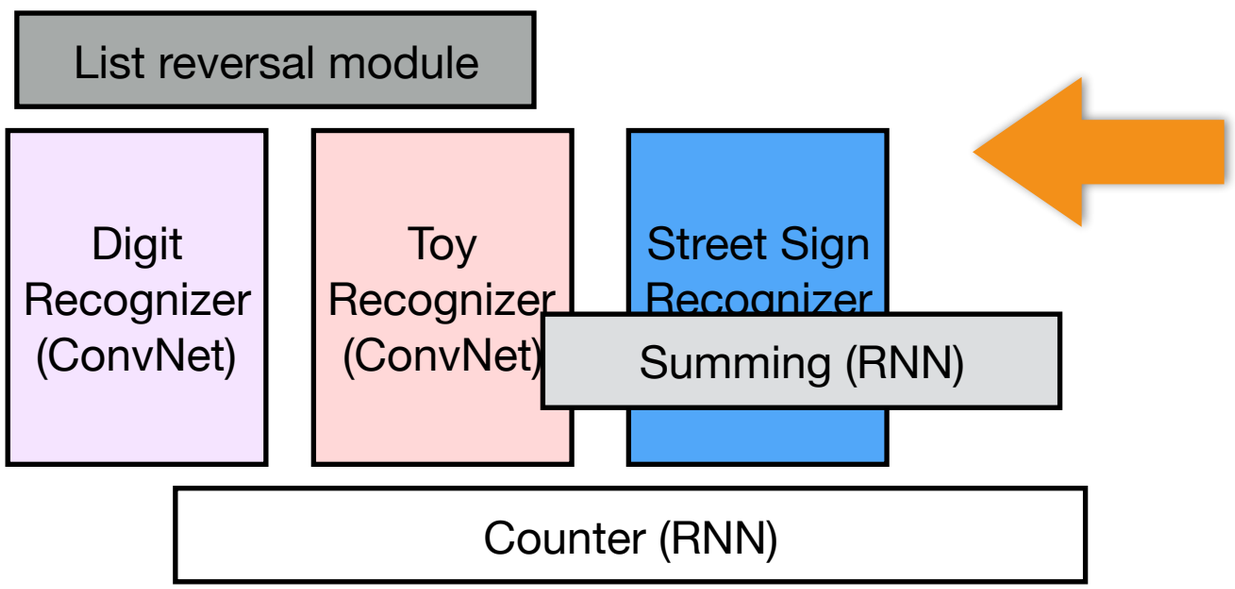
Synthesis for Lifelong Learning




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

Best program

Library

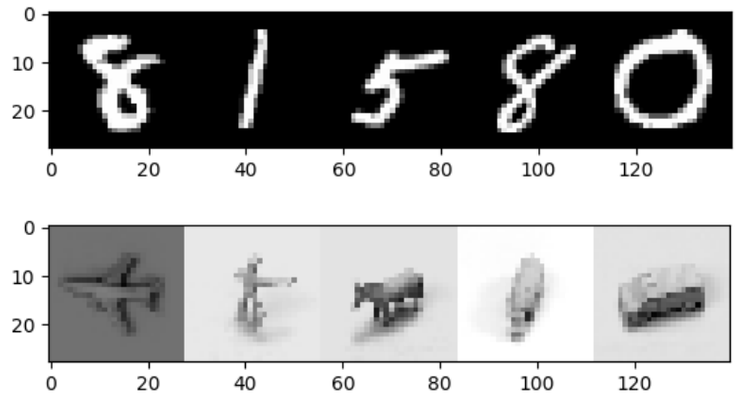


Freeze parameters

```
nn_kernel_2
nn_kernel_1
nn_kernel_0
```

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

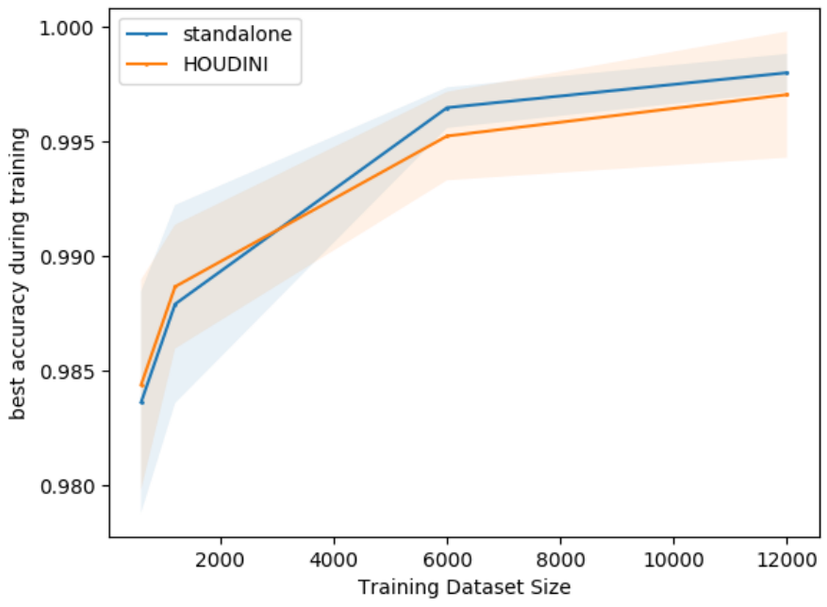
$$(\mathbf{conv}^{i}_{\mathtt{graph}\langle\mathtt{Tensor}[2]\rangle}\ nn\_relax) \circ (\mathbf{map}_{\mathtt{graph}\langle\mathtt{Tensor}[32][32][3]\rangle}\ perceive)$$

Classify(sign), Shortest_path(sign)

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

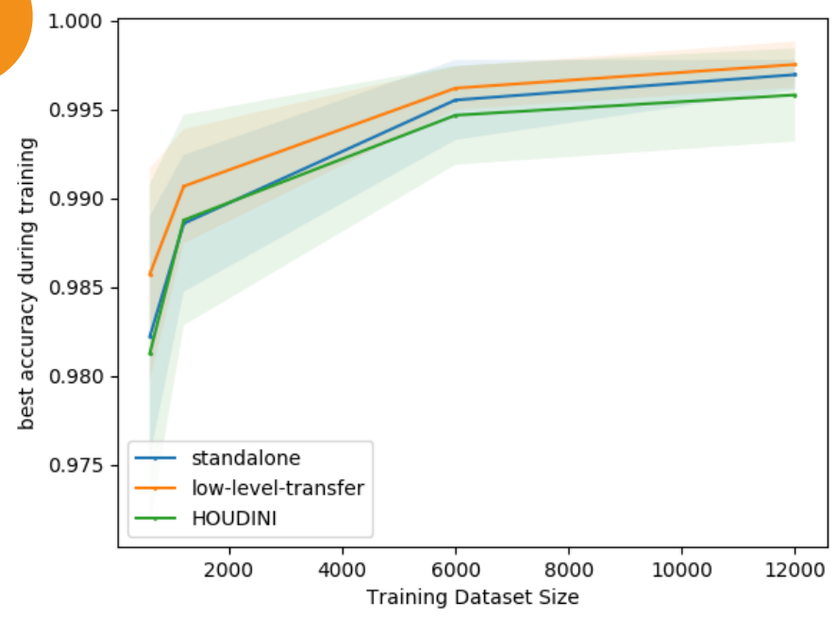
Results: Learning to Count

1



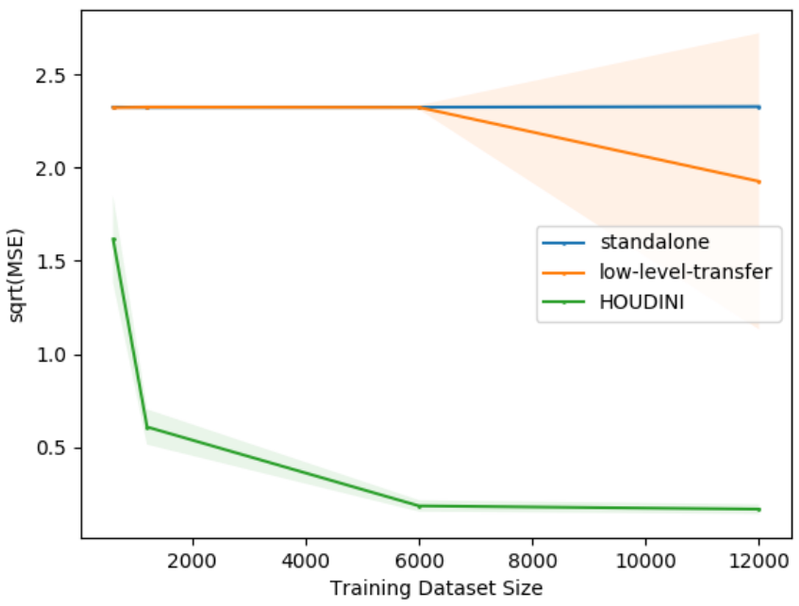
Recognise digit d_1
(binary classification)

2



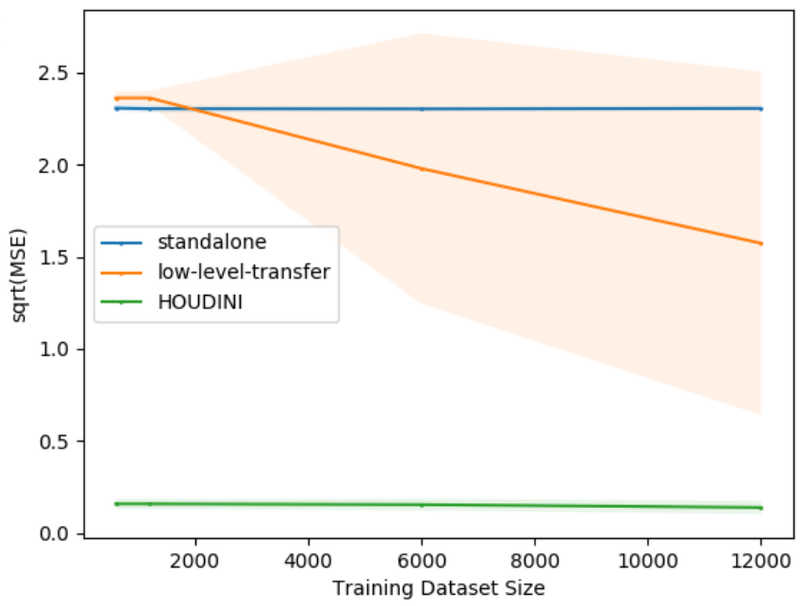
Recognise digit d_2
(binary classification)

3



Count digits d_1

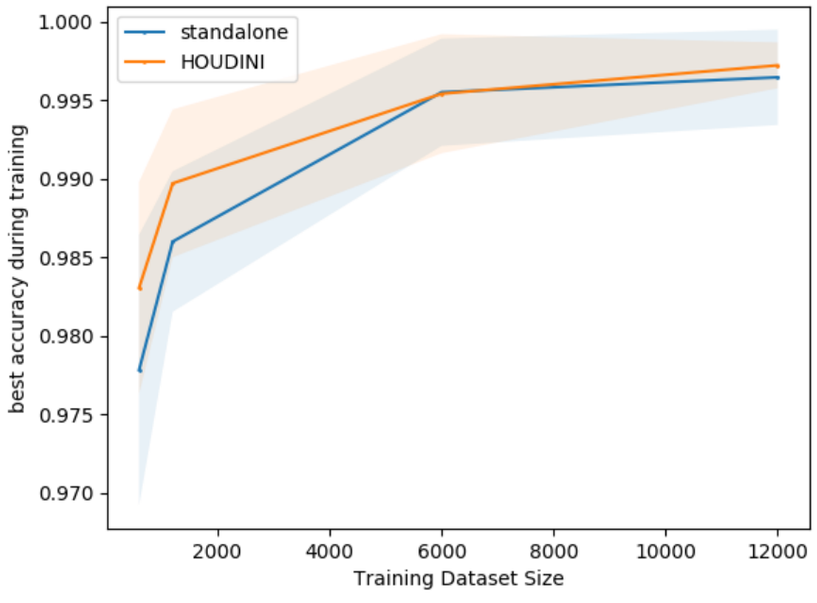
4



Count digits d_2

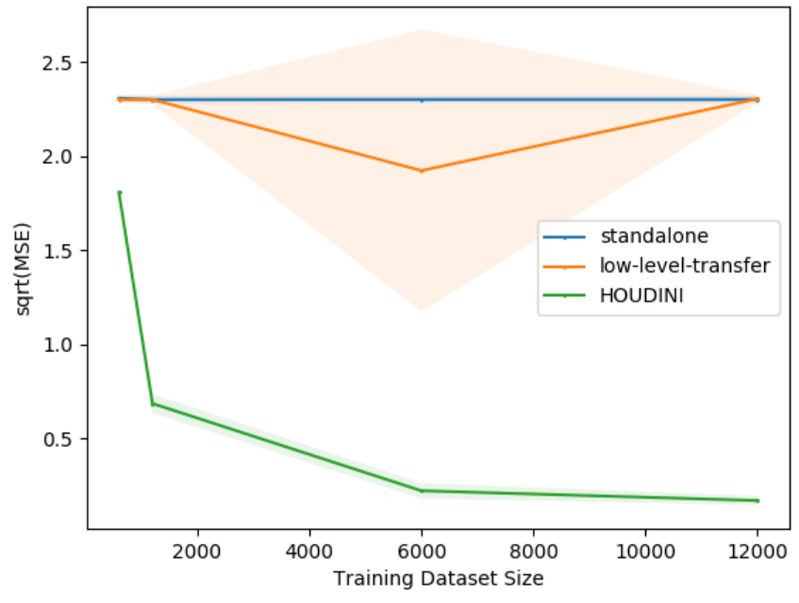
Results: Counting Toys

1



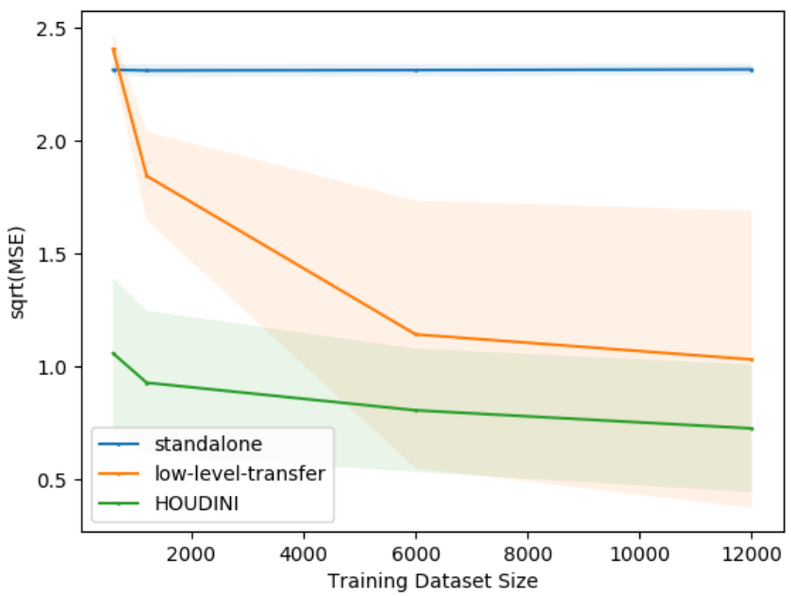
Recognise digit d_1

2



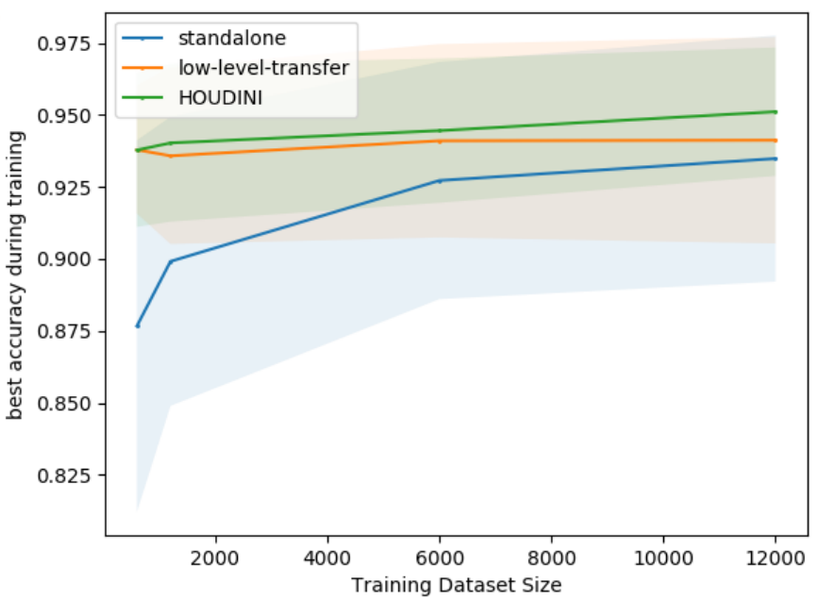
Count digit d_1

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

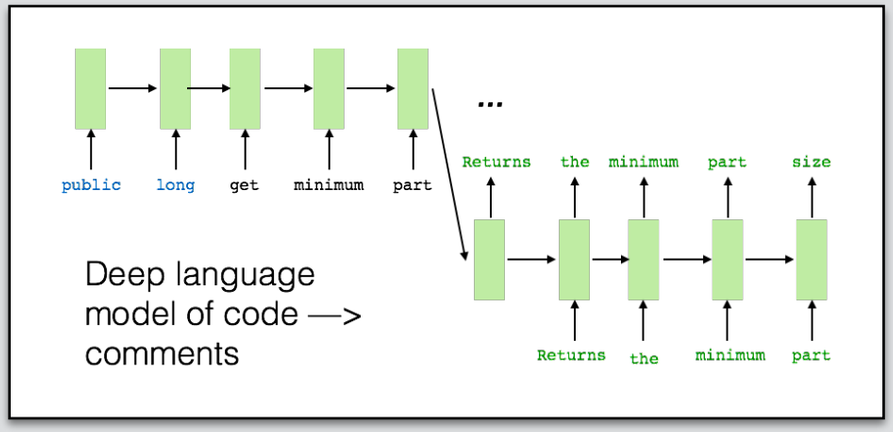
Types of transfer

- 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

Learning for Programs:

Connecting code, statistics, semantics, and language

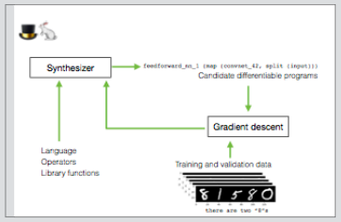
http://bit.ly/sutton-learning4programs




```
Item goodItem = null;
foreach (item in collection) {
    if ( SomethingIsTrueOfAll(item.Children) ) {
            goodItem = item;
            break;
        }
}
```

Reads from all items in collection

Writes to unitary variable




```
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)

Text informs ML program analysis

Program semantics informs ML

Programs a symbolic representation for ML

Miltiadis Allamanis
Annie Louis
Lazar Valkov
Akash Srivastava
Earl Barr
Dipak Chaudhari
Swarat Chaudhuri
Santanu Dash

Thanks!

EPSRC
Engineering and Physical Sciences Research Council

Microsoft Research