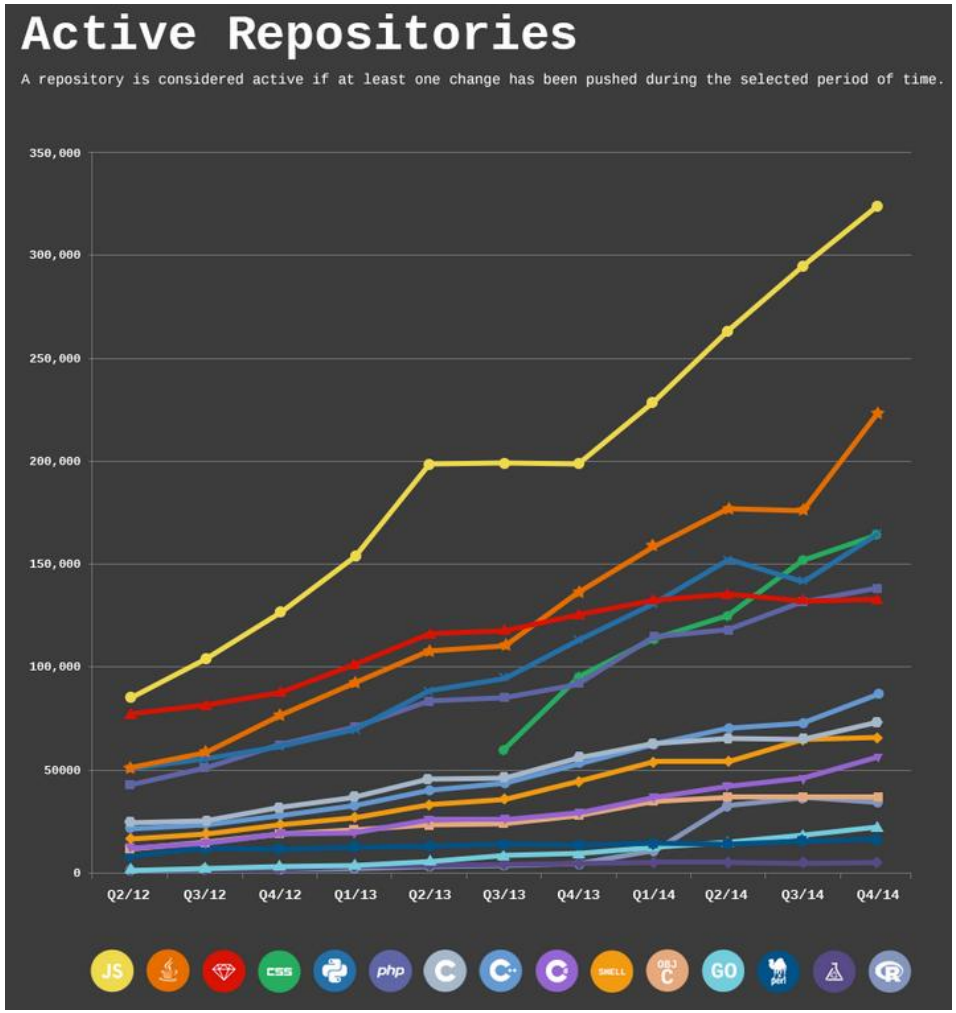
# Solving Software Engineering Problems using Neural Networks

An Overview

Carol Alexandru
s.e.a.l. Softalk
21.01.2015

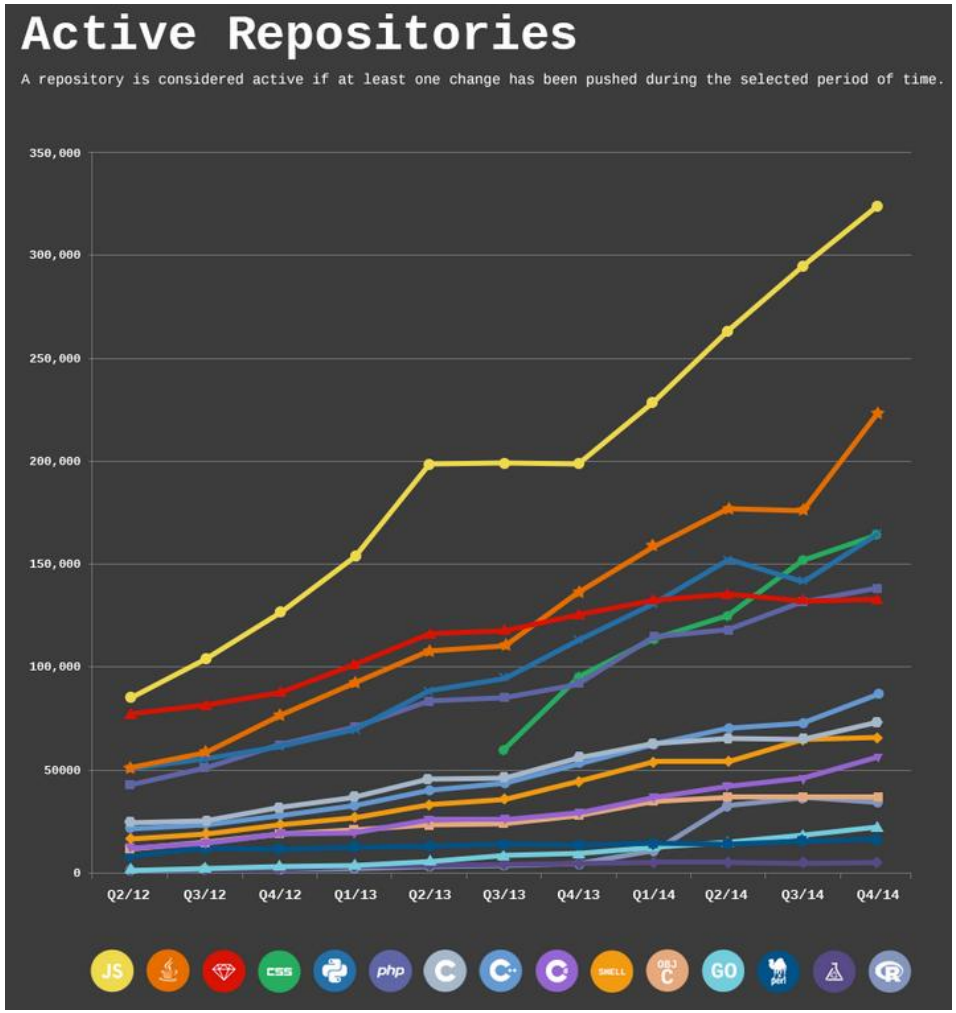
# Motivation




Enormous corpus of source code online

How can we leverage this source code?

# Motivation

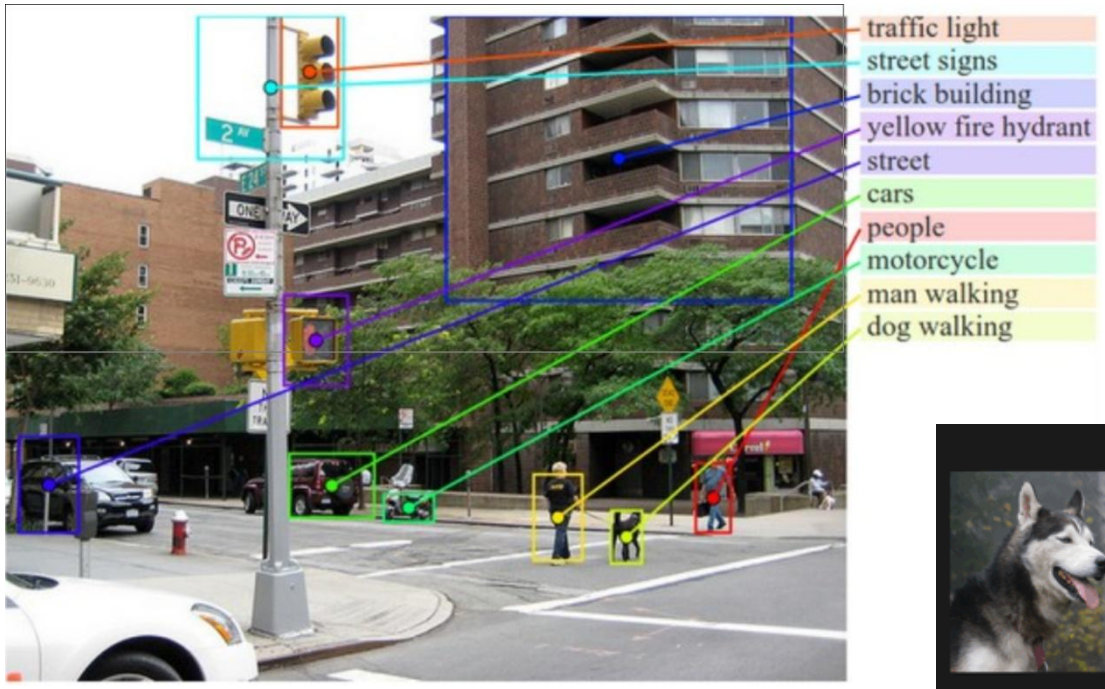


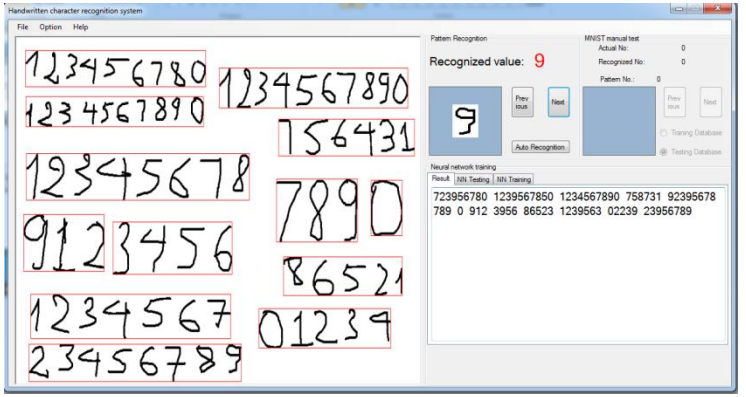

Enormous corpus of source code online

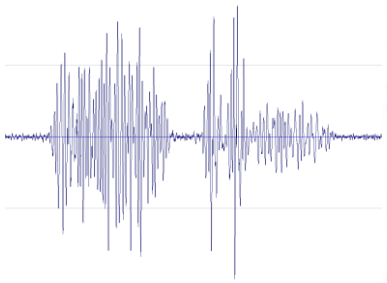
How can we leverage this source code?

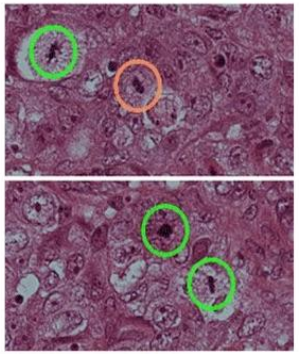
**Can we create new programs from existing code?**

# Neural Networks can recognize...

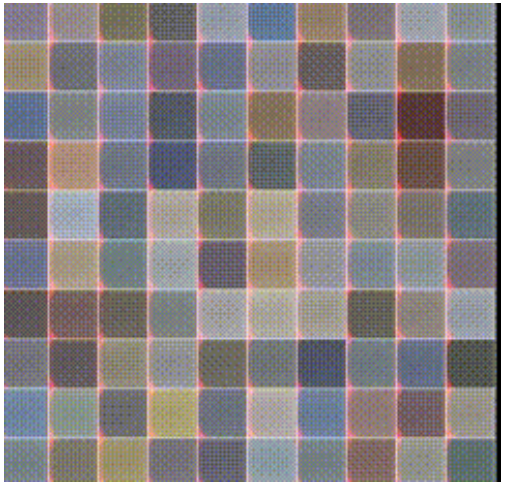


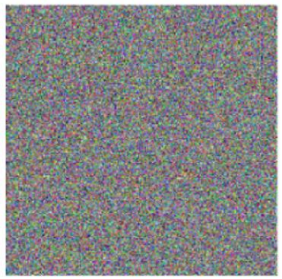



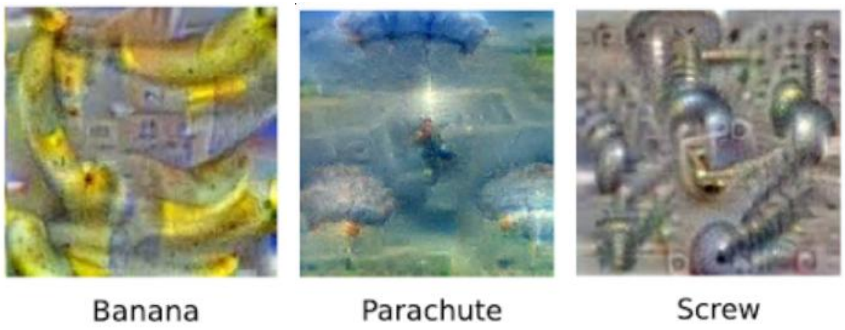





# ...but they can also synthesize!



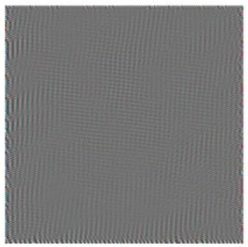




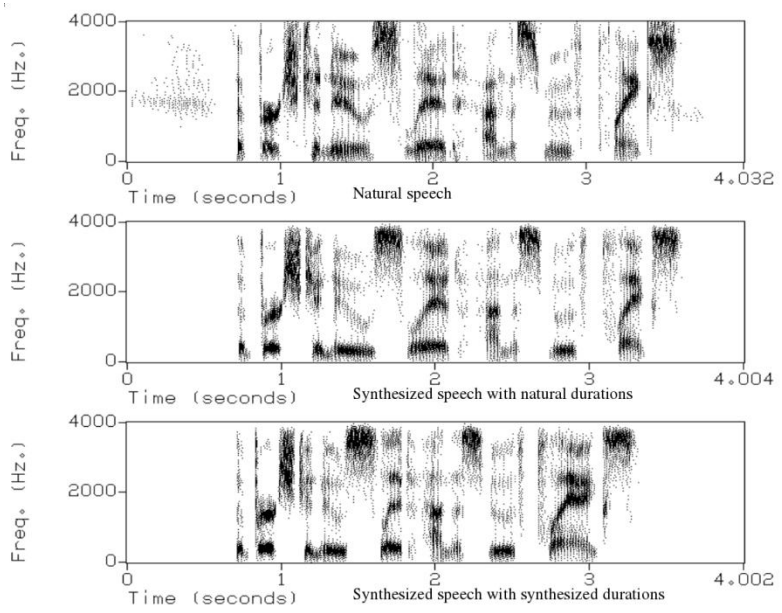

Solving Software Engineering Problems using Neural Networks

Good everybody. Thank you very much. God bless the United States of America, and has already began with the world's gathering their health insurance. It's about hard-earned for our efforts that are not continued.

We are all the assumptionion to the streets of the Americas that we are still for everybody and destruction. We are doing a lot of this. I know that someone would be prefered to their children to take a million insurance company. We're watching their people and continued to find ourselves with Republicans — to give up on these challenges and despite the challenges of our country. In the last two years, we must recognise that our borders have access from the world.



original



# Outline

## Part 1

Quick Intro to Artificial Neural Networks (ANN)

## Part 2

Related Work applying ANN to SE Problems

## Part 3

Current Work & Avenues for further research

# Part 1

# Quick Intro to Artificial Neural Networks (ANN)

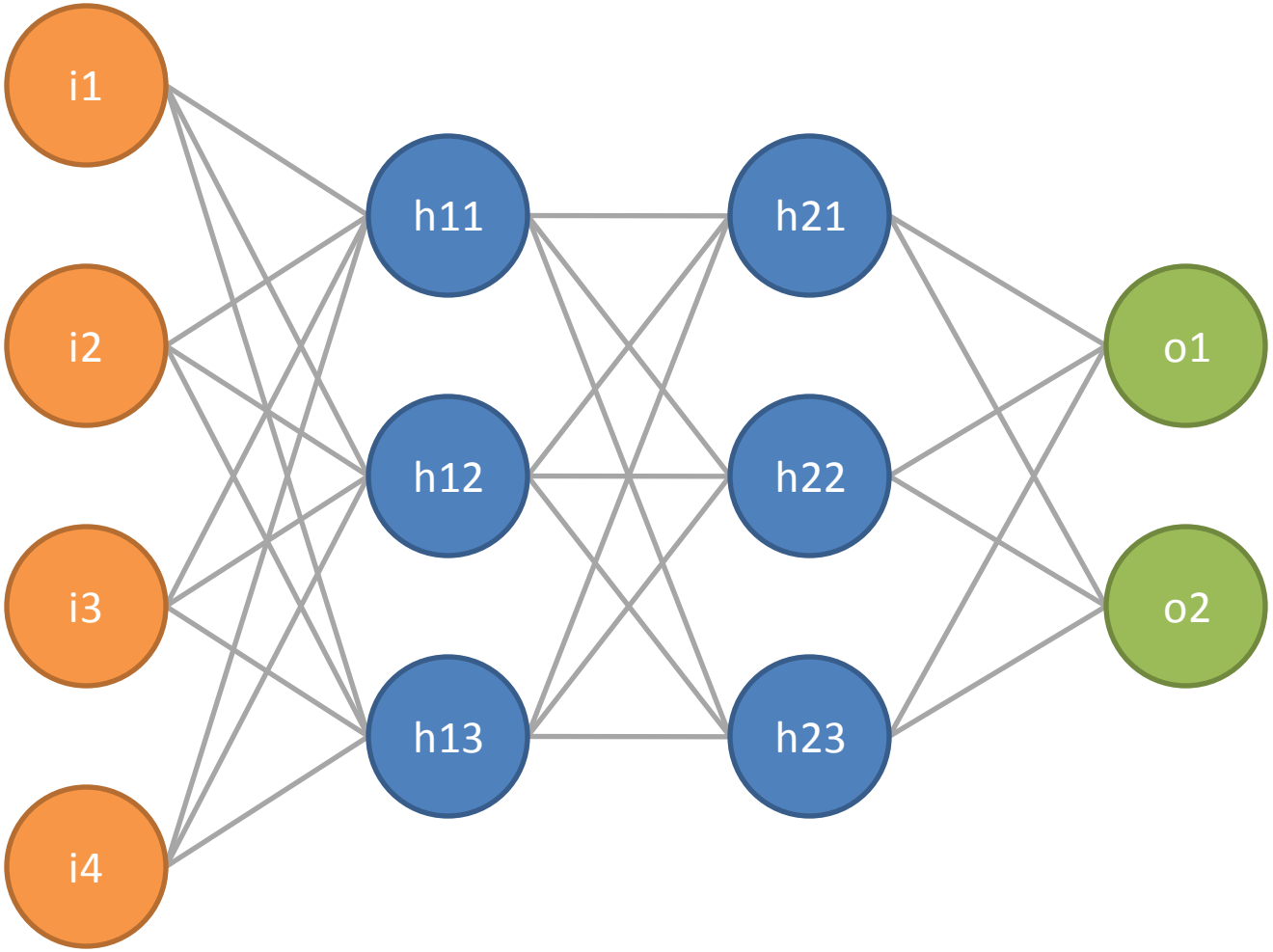
# Neural Networks

- A type of machine learning
- Around since the 1950s
- Gained traction in the late 2000s thanks to higher availability of computational resources

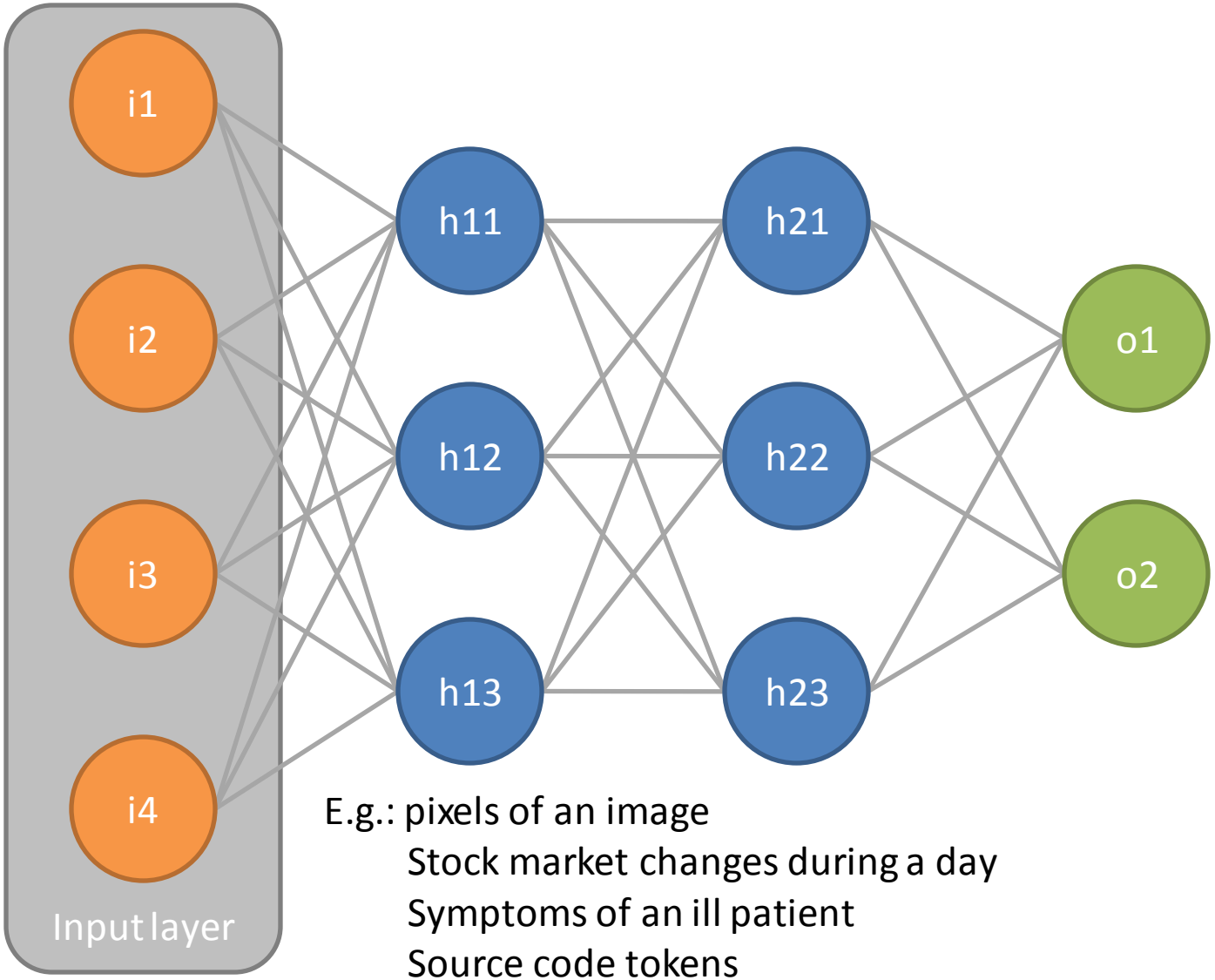
# Neural Networks

- A type of machine learning
- Around since the 1950s
- Gained traction in the late 2000s thanks to higher availability of computational resources
- Good at recognizing complex patterns and dependencies in *raw* data
- „The“ solution for hard classification and recognition problems

# Neural Networks - Overview

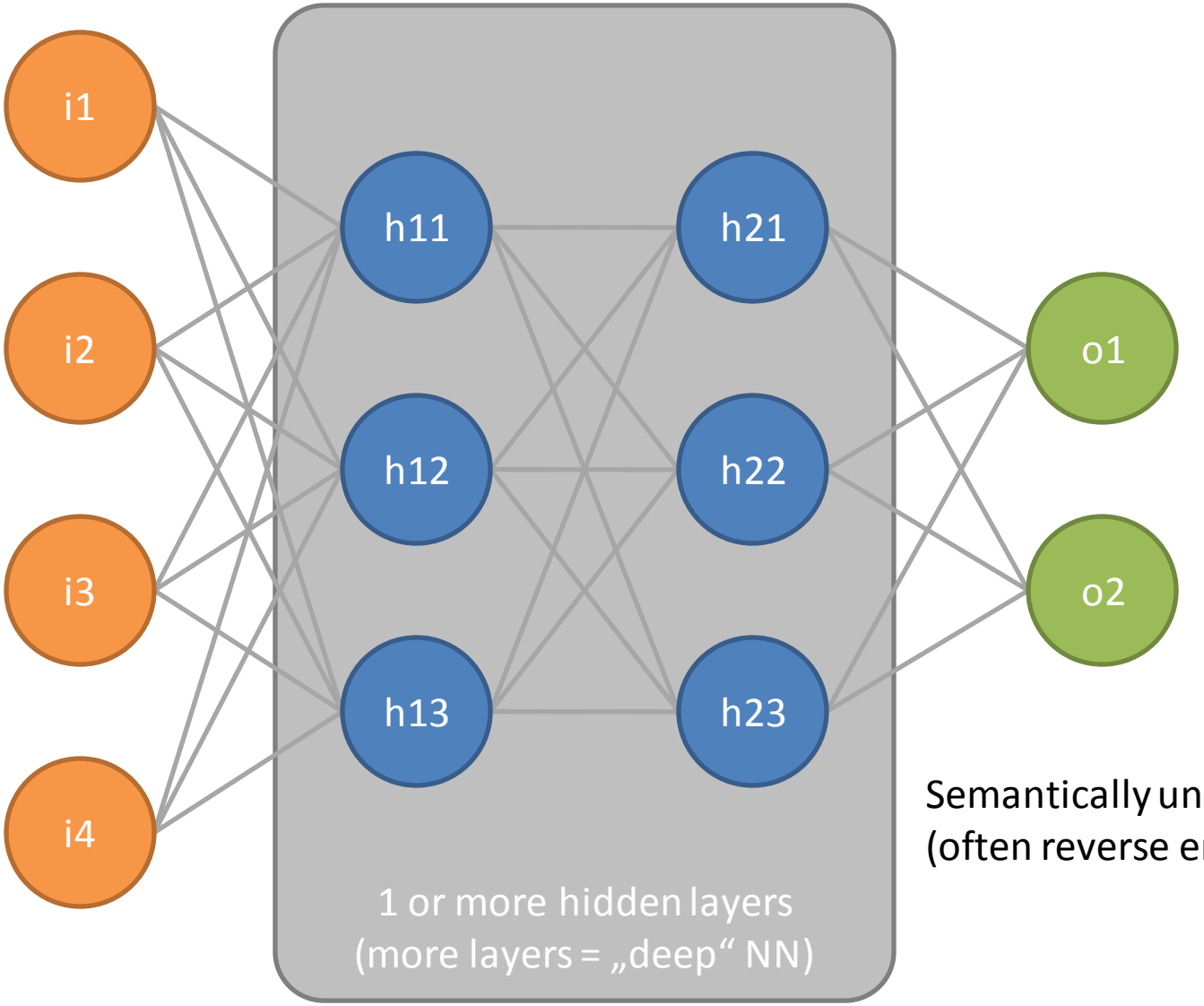


# Neural Networks – Input Layer



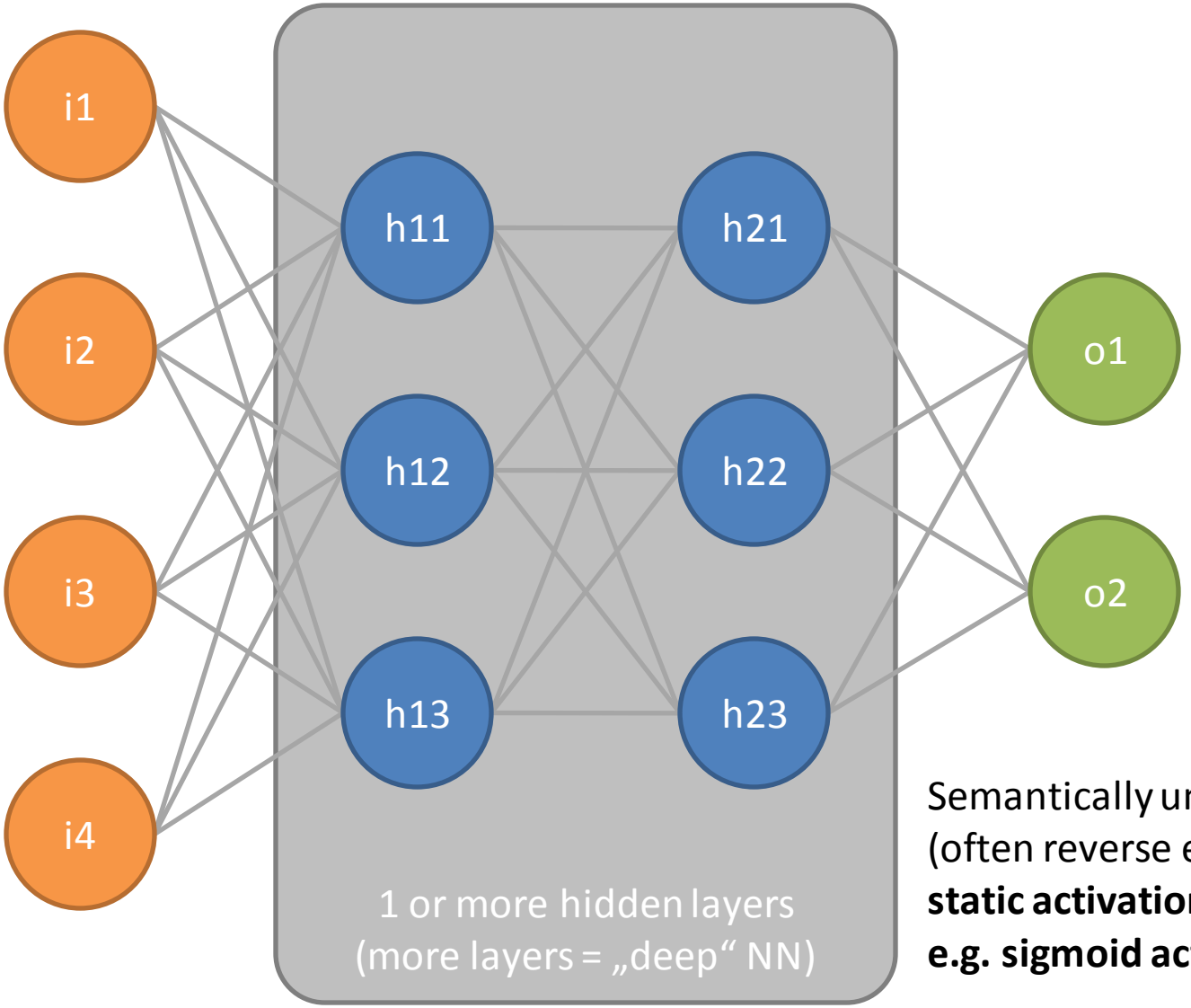

E.g.: pixels of an image
Stock market changes during a day
Symptoms of an ill patient
Source code tokens

# Neural Networks - Hidden Layers



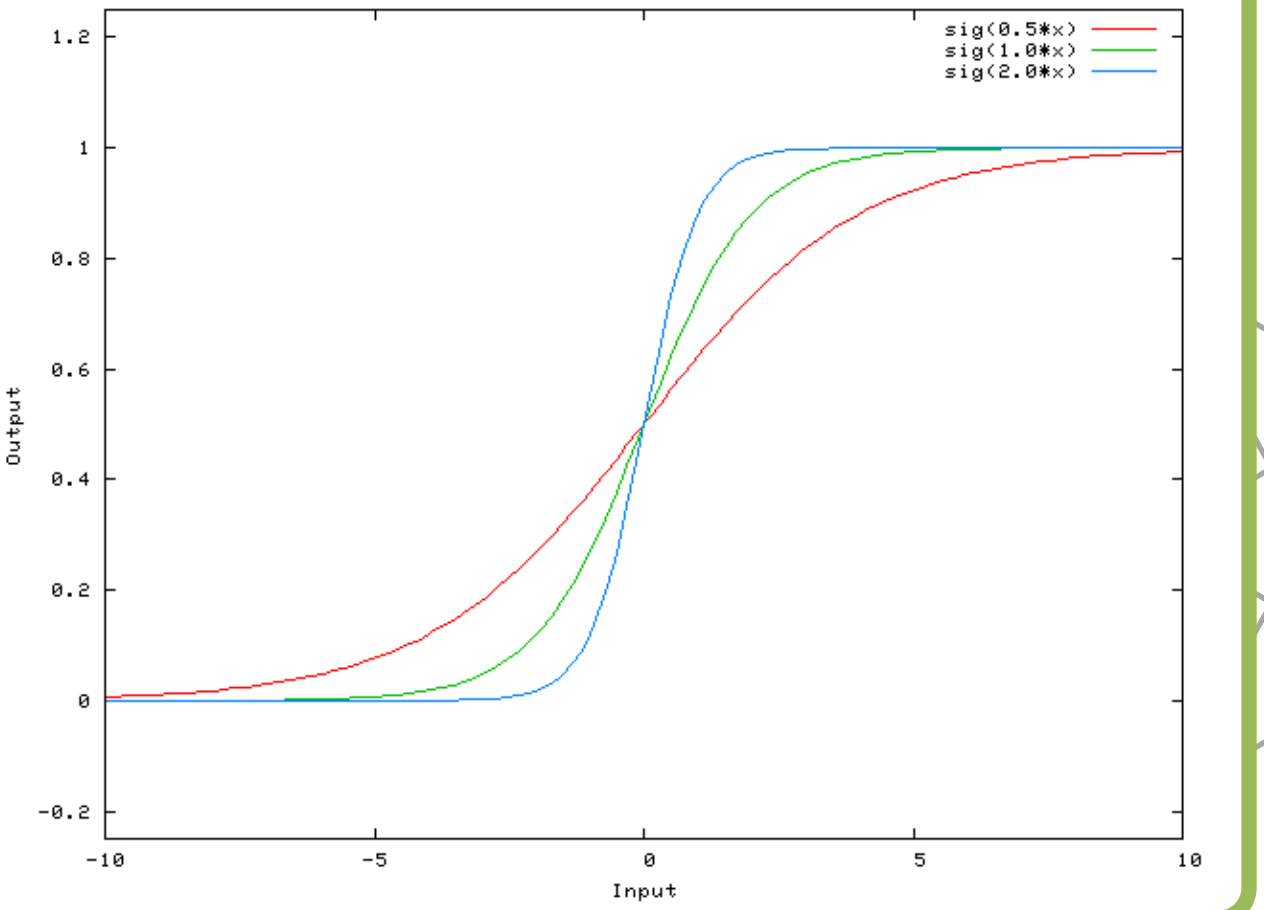

Semantically undetermined
(often reverse engineered)

# Neural Networks - Hidden Layers




Semantically undetermined (often reverse engineered) **static activation function**, **e.g. sigmoid activation**

# Neural Networks - Hidden Layers




o1

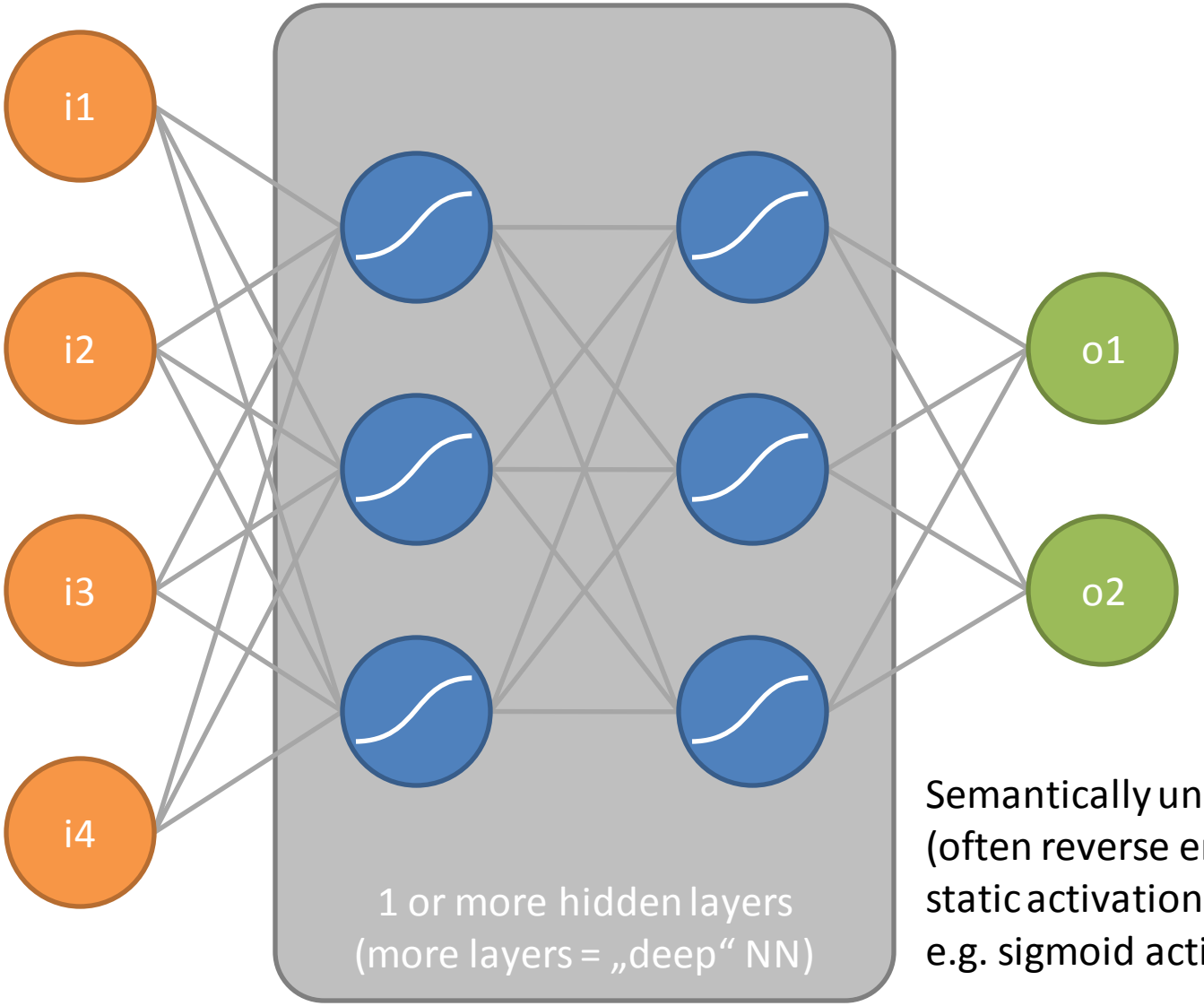
o2

i4

1 or more hidden layers
(more layers = „deep“ NN)

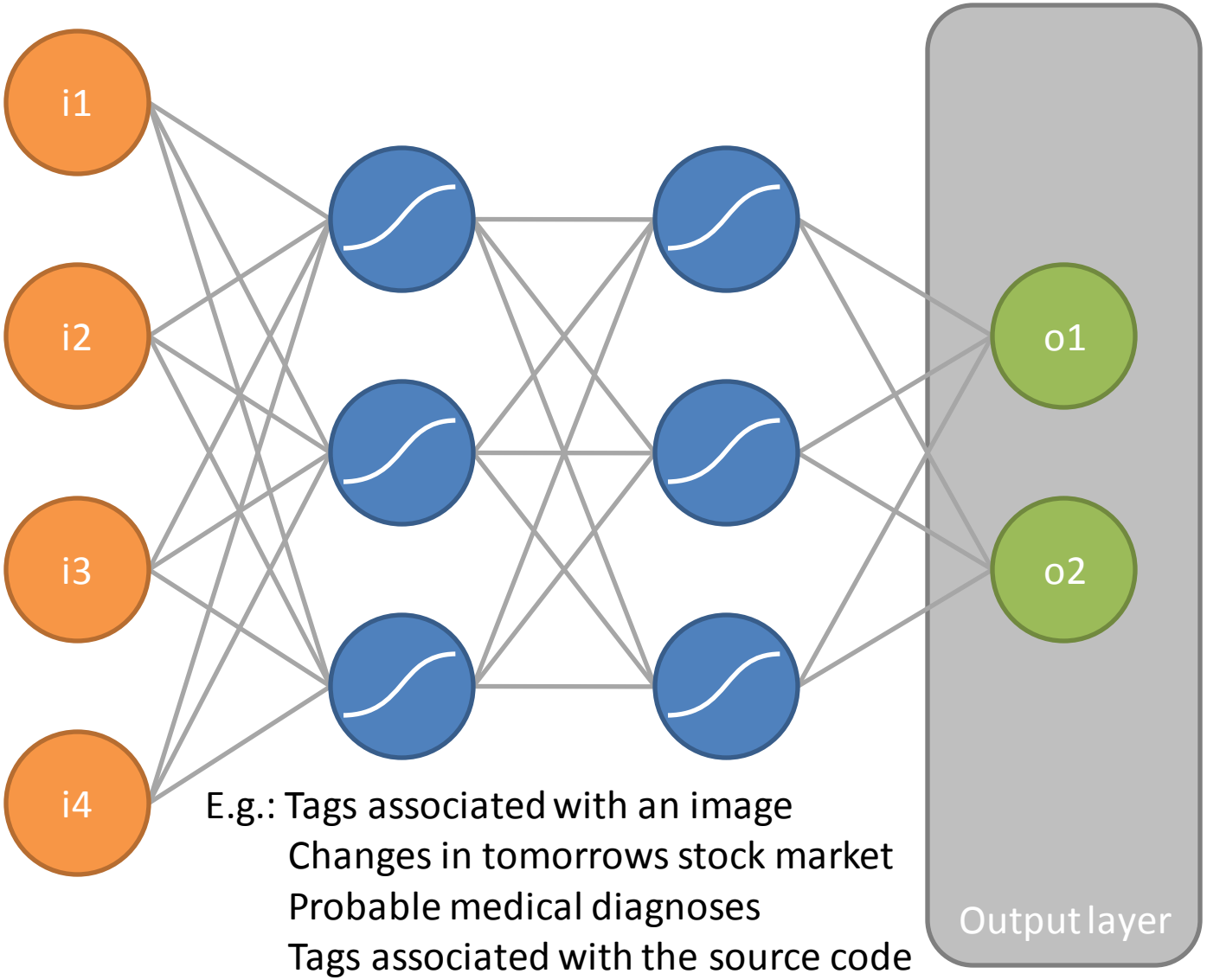
Semantically undetermined (often reverse engineered) **static activation function**, **e.g. sigmoid activation**

# Neural Networks - Hidden Layers



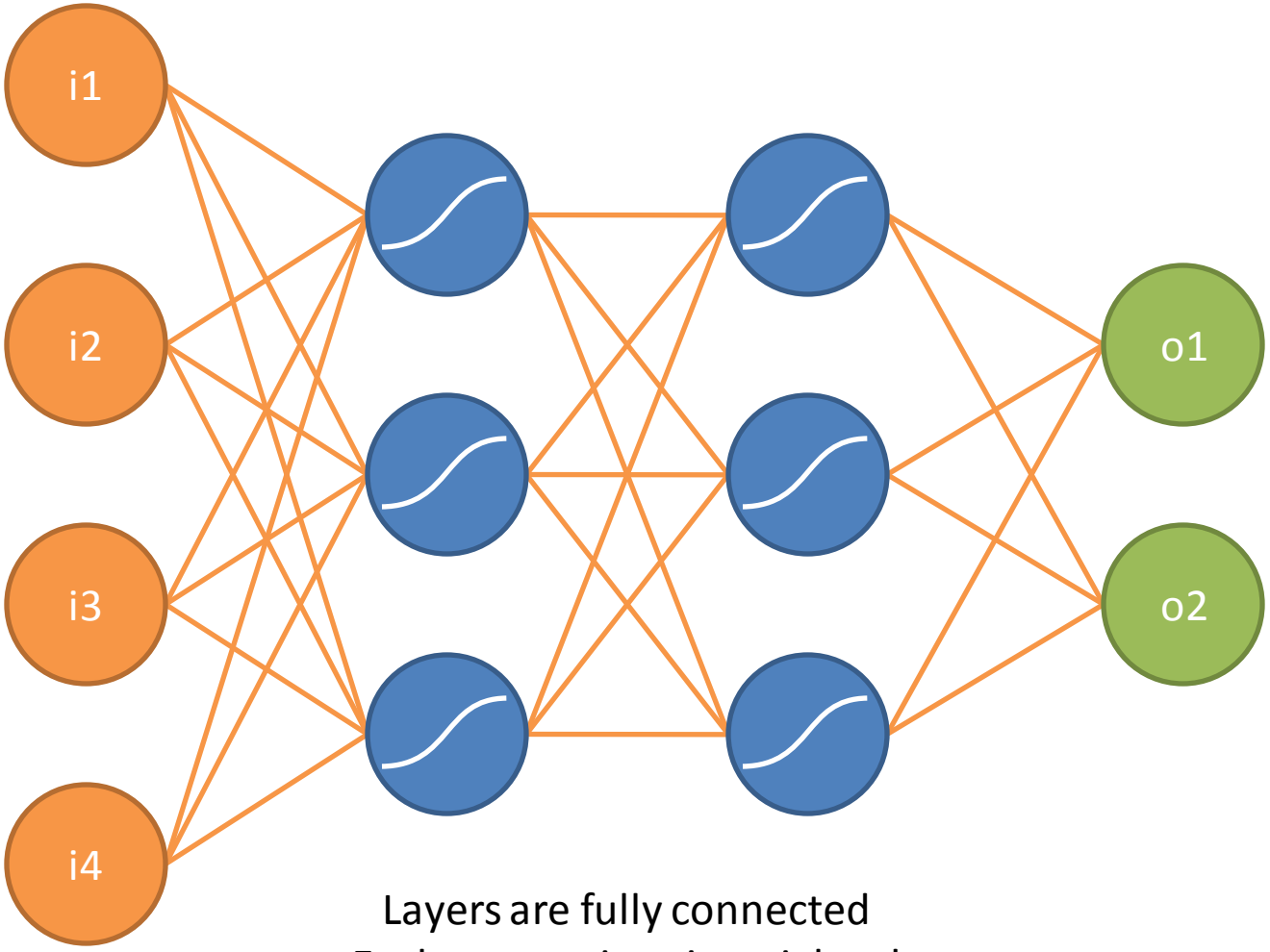

Semantically undetermined (often reverse engineered) static activation function, e.g. sigmoid activation

# Neural Networks – Output Layer



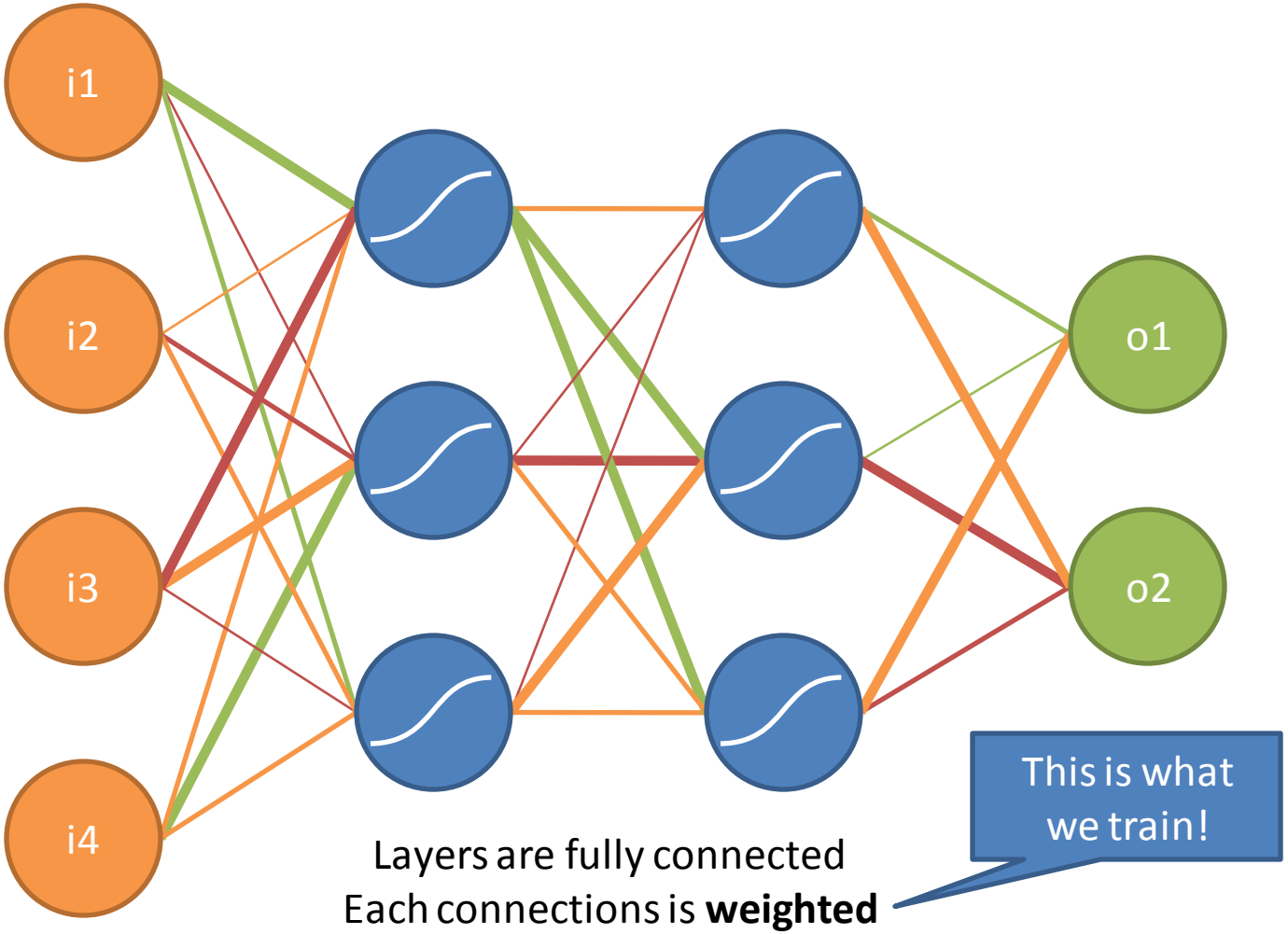

E.g.: Tags associated with an image
Changes in tomorrows stock market
Probable medical diagnoses
Tags associated with the source code

# Neural Networks



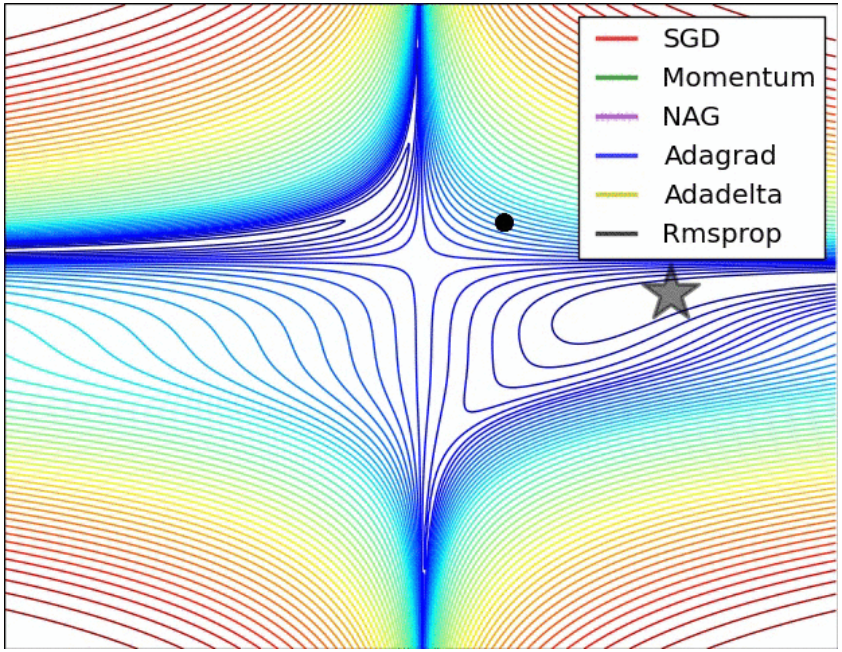

Layers are fully connected
Each connections is weighted

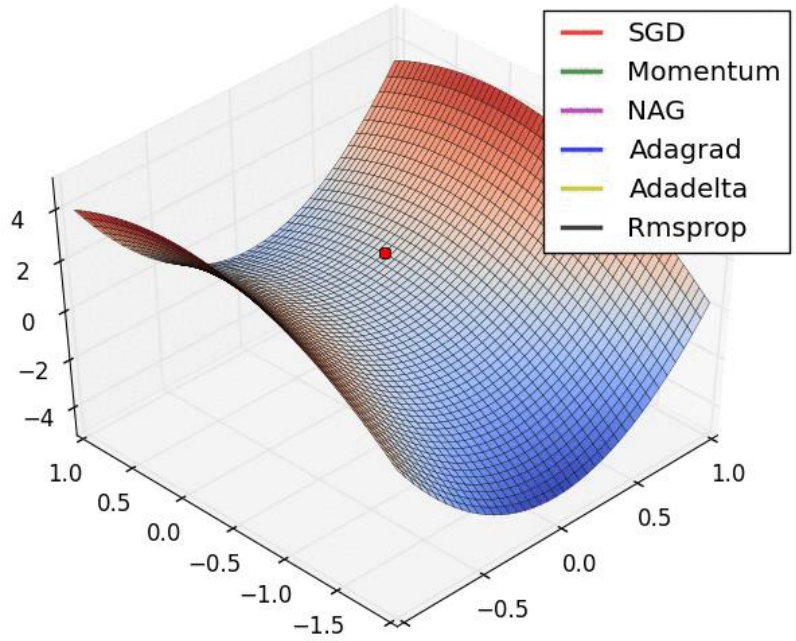
# Neural Networks




Layers are fully connected
Each connections is **weighted**
(weights can be negative)

# Neural Networks



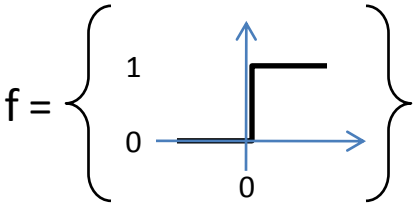

i4

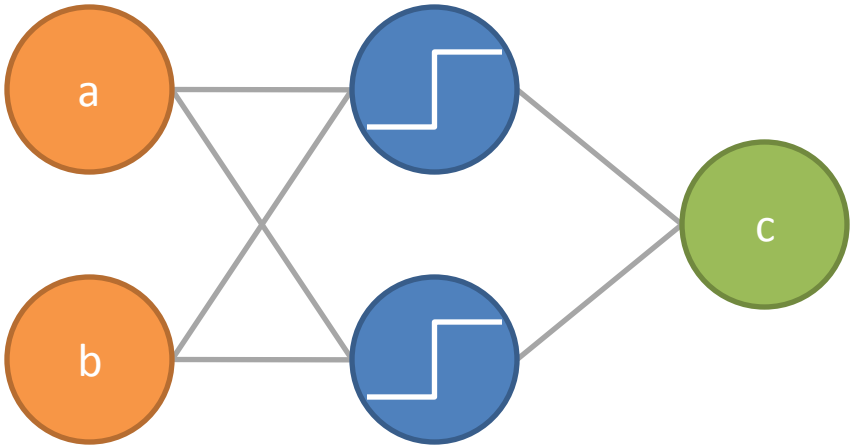
Layers are fully connected
Each connections is **weighted**
(weights can be negative)

This is what we train!

# Concrete Example: XOR

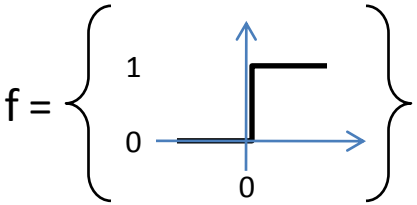
| a | b | c |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 1 | 0 |

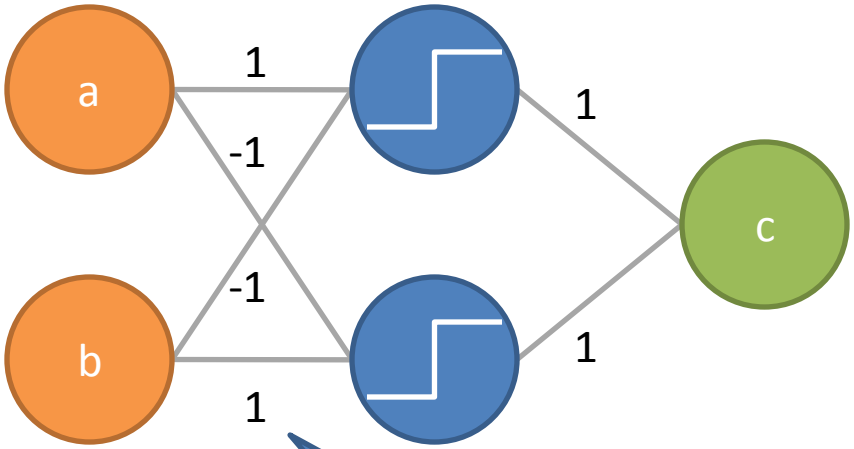


# Concrete Example: XOR

| a | b | c |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 1 | 0 |

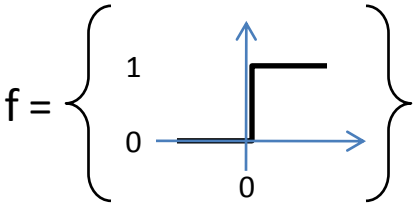



An ANN would likely come up with different weights

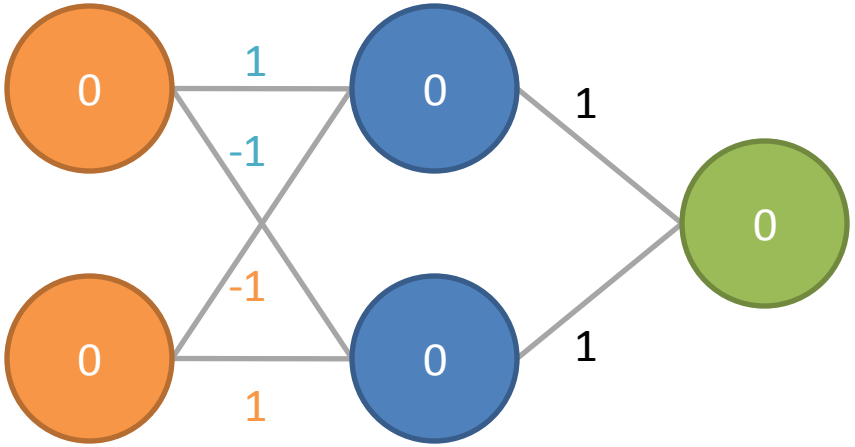
# Concrete Example: XOR input [0,0]

| a | b | c |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 1 | 0 |




simplified

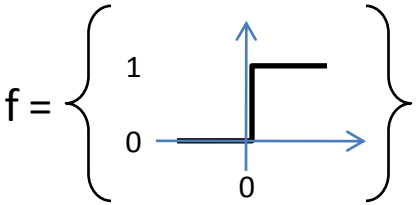
f(0*1+0*-1) = f(0) = 0

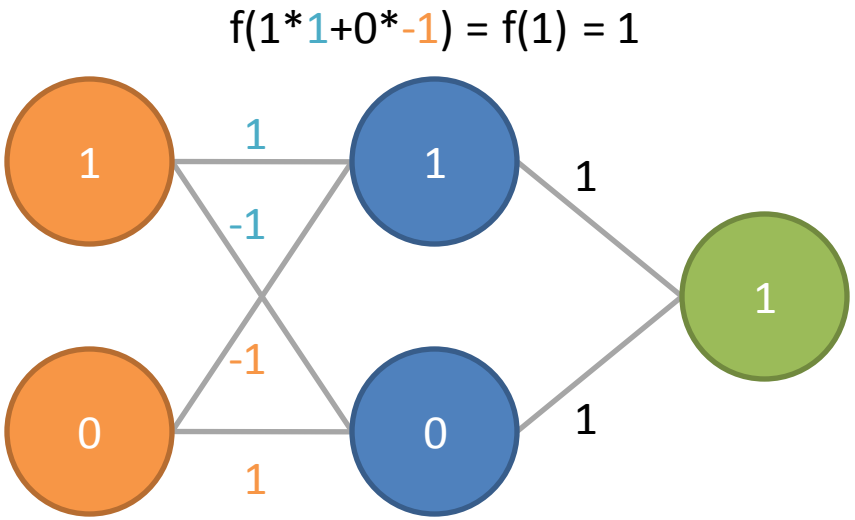



f(0*-1+0*1) = f(0) = 0

# Concrete Example: XOR input [1,0]

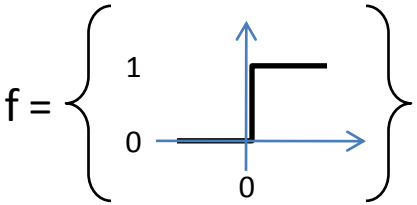
| a | b | c |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 1 | 0 |

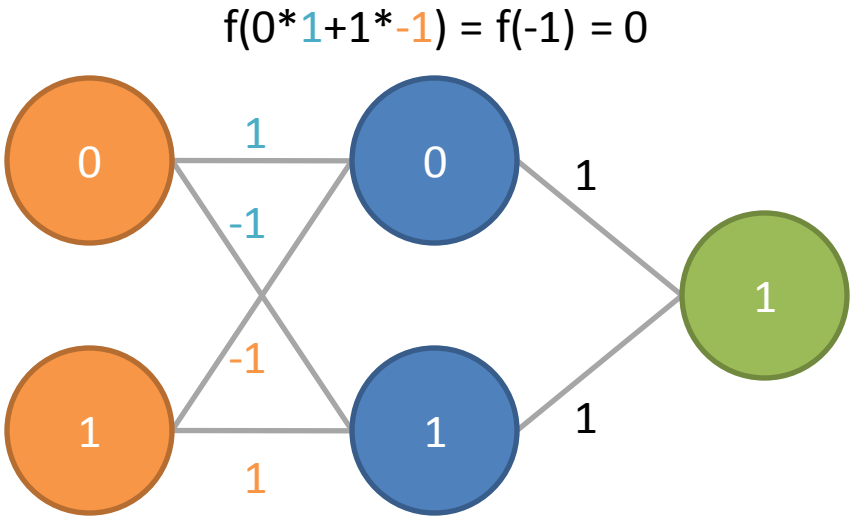


# Concrete Example: XOR input [0,1]

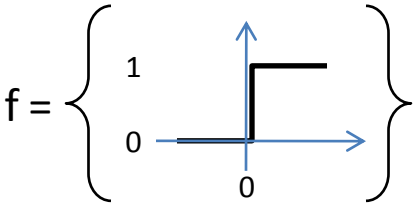
| a | b | c |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 1 | 0 |

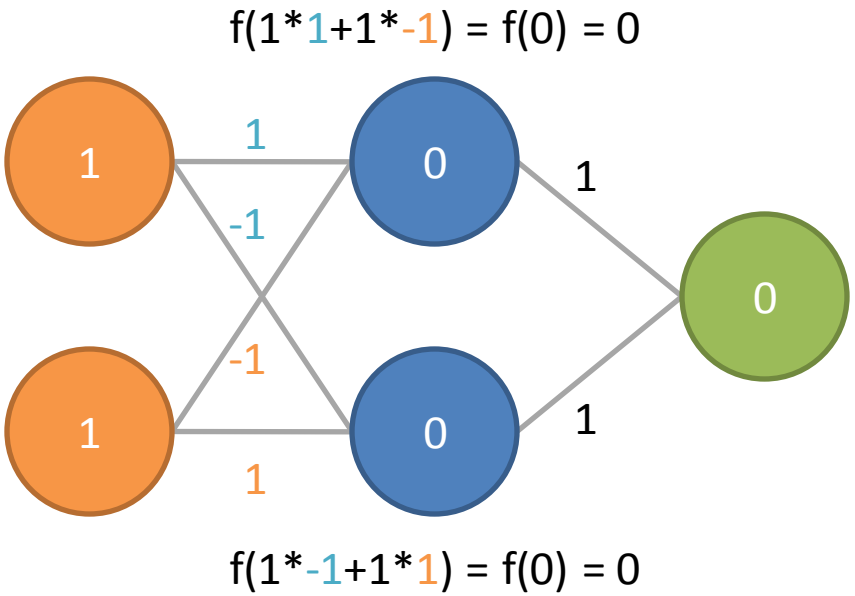


# Concrete Example: XOR input [1,1]

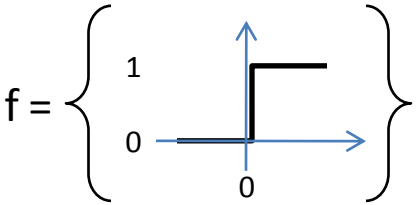
| a | b | c |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 1 | 0 |

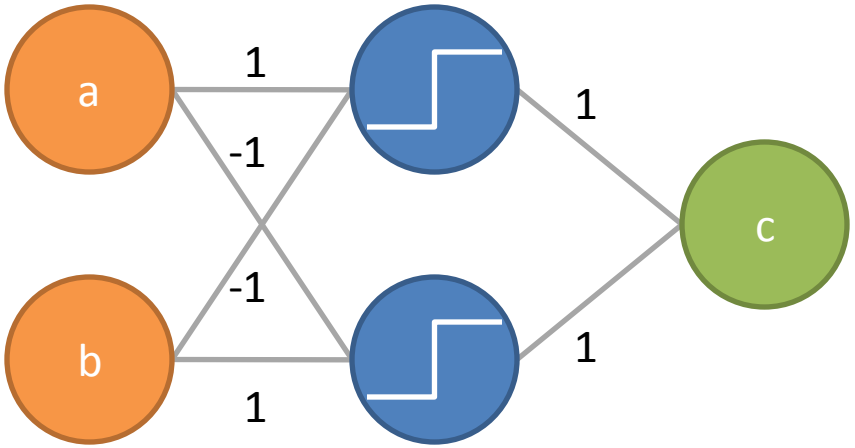


# Concrete Example: XOR

| a | b | c |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 1 | 0 |



# Basic Neural Networks - Summary

- Any number of inputs and outputs
- Any number of hidden layers (even just 1)
- Activation function (often sigmoid + bias)
- Training happens on the weights of the links
- There is no real „signaling“, the entire network can be represented as a single math function
- Can be used for both regression and classification problems

# Part 2

# Related Work applying ANN to SE

# Software Reliability Lab (ETHZ)

SLANG (PLDI 14):

Code Completion (gap filling) using n-grams and ANN with regard to API usage

```
void exampleMediaRecorder() throws IOException {
 Camera camera = Camera.open();
 camera.setDisplayOrientation(90);
   ?  // (H1)
 SurfaceHolder holder = getHolder();
 holder.addCallback(this);
 holder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
 MediaRecorder rec = new MediaRecorder();
   ?  // (H2)
 rec.setAudioSource(MediaRecorder.AudioSource.MIC);
 rec.setVideoSource(MediaRecorder.VideoSource.DEFAULT);
 rec.setOutputFormat(MediaRecorder.OutputFormat.MPEG_4);
   ? {rec}  // (H3)
 rec.setOutputFile("file.mp4");
 rec.setPreviewDisplay(holder.getSurface());
 rec.setOrientationHint(90);
 rec.prepare();
   ? {rec}  // (H4)
}
```

```
void exampleMediaRecorder() throws IOException {
 Camera camera = Camera.open();
 camera.setDisplayOrientation(90);
 camera.unlock();
 SurfaceHolder holder = getHolder();
 holder.addCallback(this);
 holder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
 rec = new MediaRecorder();
 rec.setCamera(camera);
 rec.setAudioSource(MediaRecorder.AudioSource.MIC);
 rec.setVideoSource(MediaRecorder.VideoSource.DEFAULT);
 rec.setOutputFormat(MediaRecorder.OutputFormat.MPEG_4);
 rec.setAudioEncoder(1);
 rec.setVideoEncoder(3);
 rec.setOutputFile("file.mp4");
 rec.setPreviewDisplay(holder.getSurface());
 rec.setOrientationHint(90);
 rec.prepare();
 rec.start();
}
```

# Software Reliability Lab (ETHZ)

JSNice (POPL 15):

Predicting variable names and inferring types in obfuscated JavaScript code using Conditional Random Fields (CRF)

Special kind of classifier that takes neighboring output nodes into account when making output predictions

# Software Reliability Lab (ETHZ)

JSNice (POPL 15):

Predicting variable names and inferring types in obfuscated JavaScript code using Conditional Random Fields (CRF)

```
function chunkData(e, t) {
  var n = [];
  var r = e.length;
  var i = 0;
  for (; i < r; i += t) {
    if (i + t < r) {
      n.push(e.substring(i, i + t));
    } else {
      n.push(e.substring(i, r));
    }
  }
  return n;
}
```

```
/* str: string, step: number, return: Array */
function chunkData(str, step) {
  var colNames = []; /* colNames: Array */
  var len = str.length;
  var i = 0; /* i: number */
  for (; i < len; i += step) {
    if (i + step < len) {
      colNames.push(str.substring(i, i + step));
    } else {
      colNames.push(str.substring(i, len));
    }
  }
  return colNames;
}
```

# Other Related Work

Arar et al.: “Software defect prediction using cost-sensitive neural network” (ASC 2015)

ANN + Artifical Beehive Colony algorithm, Similar performance to existing bug prediction approaches

Corley et al.: “Exploring the Use of Deep Learning for Feature Location” (ICSME15)

Preliminary Study using Latent Dirichlet Allocation (LDA) vs. Document Vectors (DV). DV are very fast - could be used for IDE-based search functionality

White et al.: „Toward Deep Learning Software Repositories“ (MSR15)

ANN better than n-gram model at predicting the next token (code suggestion)

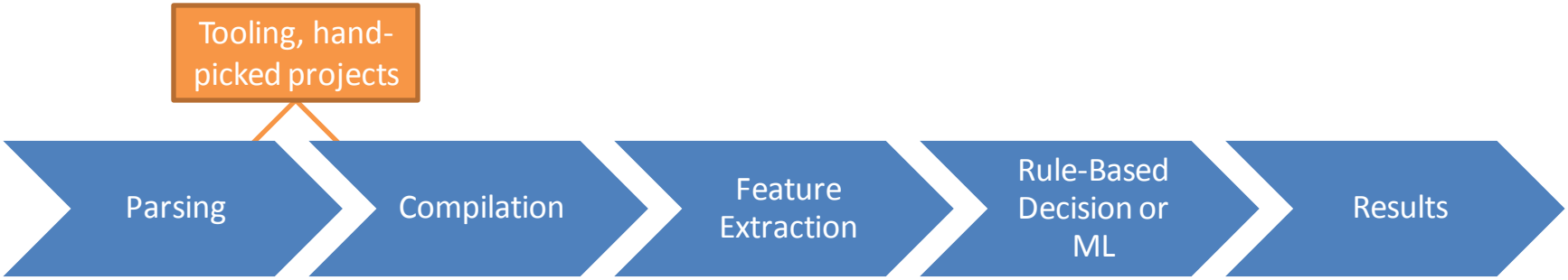
# Part 3

# Current Work & Avenues for further research

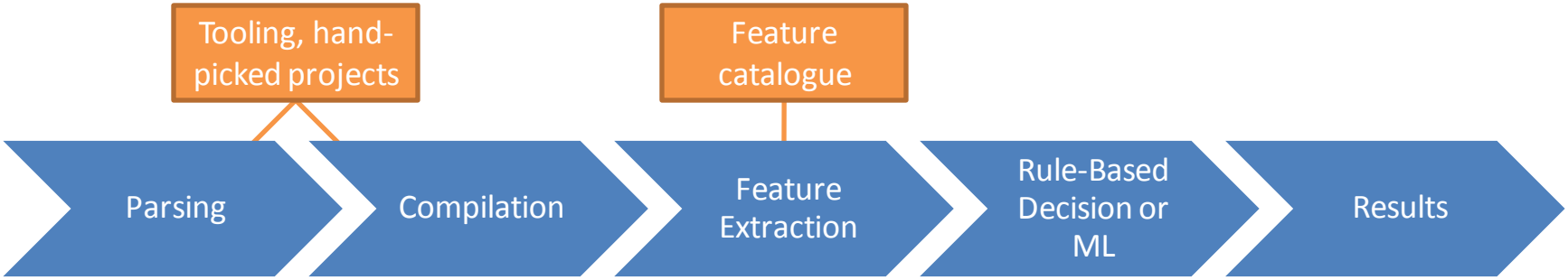
# Models vs. ANN

# Models vs. ANN

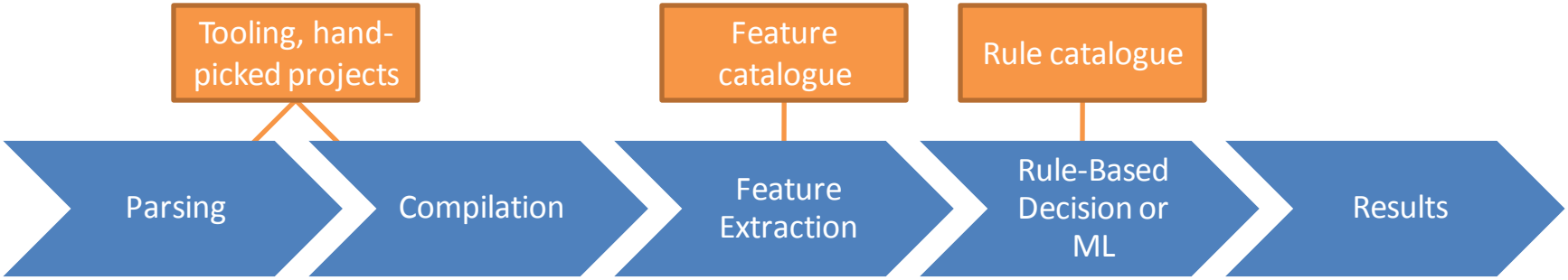
# Models vs. ANN



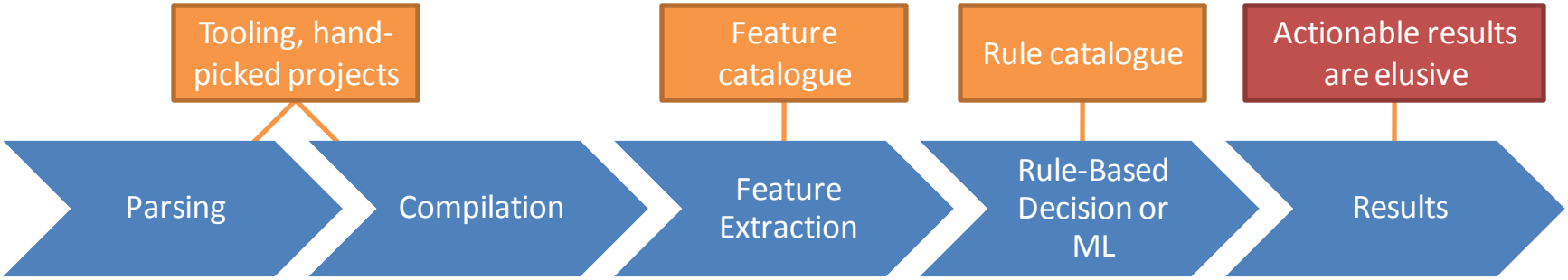
# Models vs. ANN



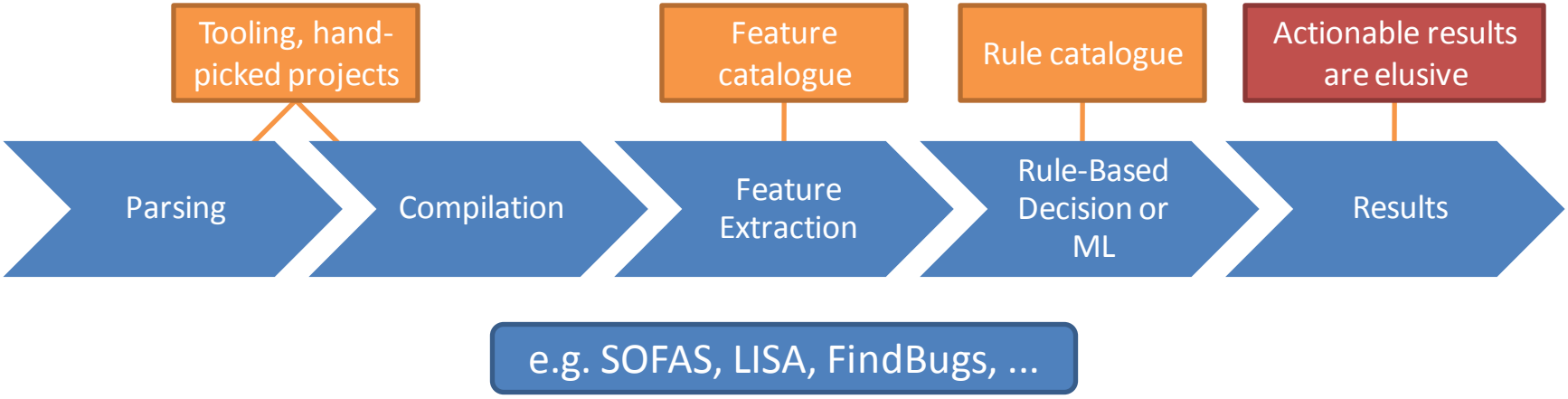
# Models vs. ANN



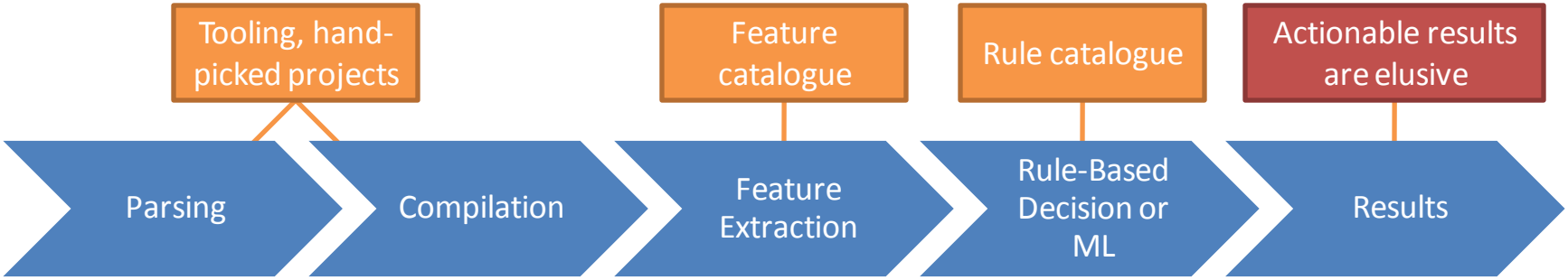
# Models vs. ANN



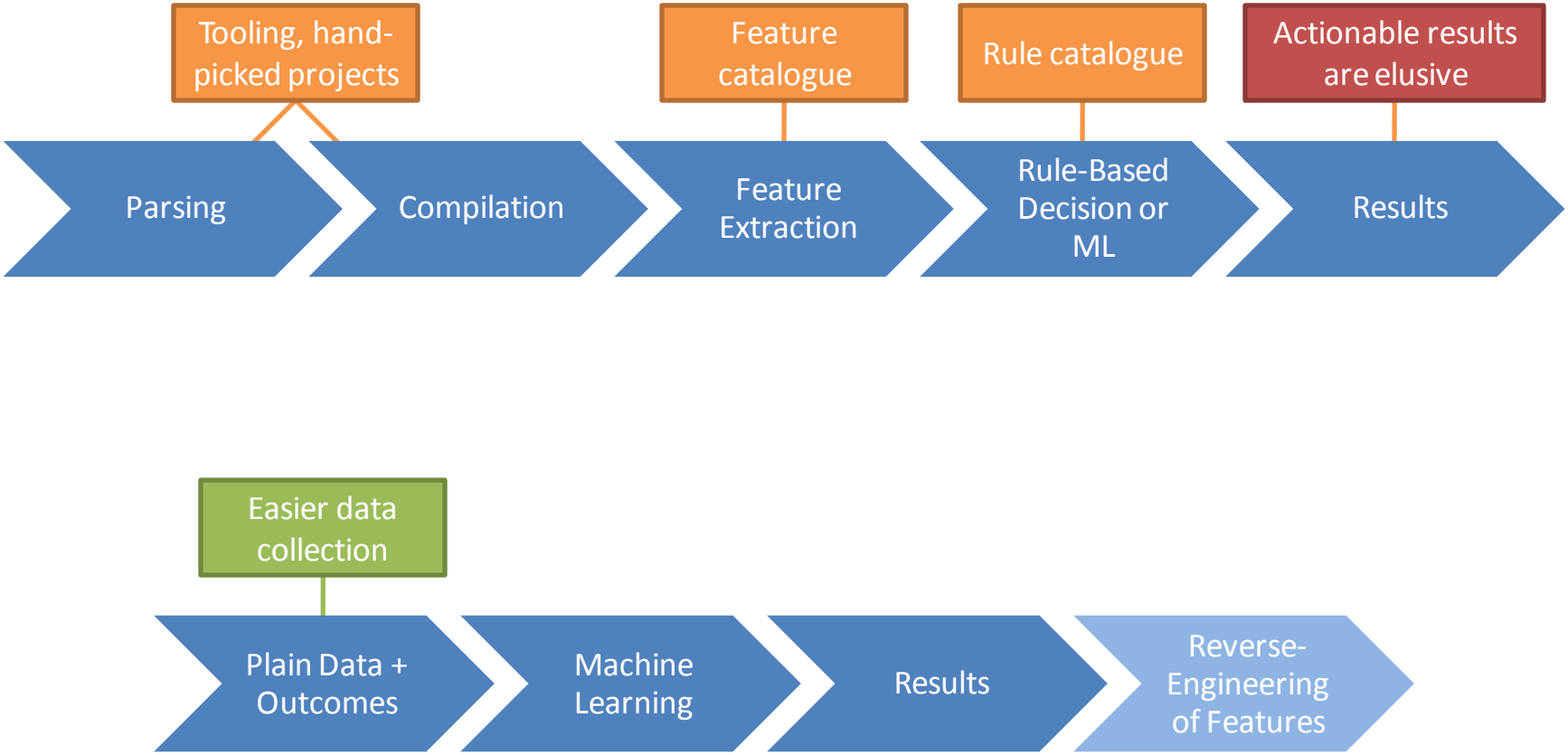
# Models vs. ANN



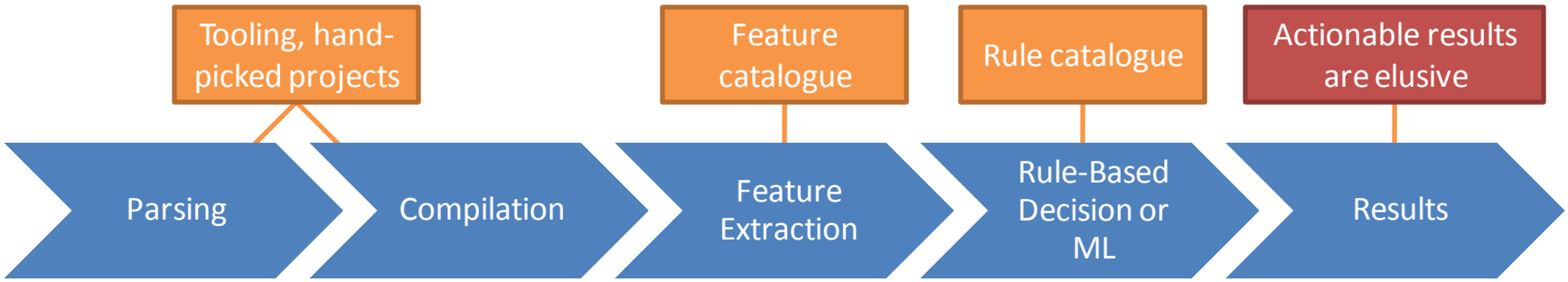
# Models vs. ANN

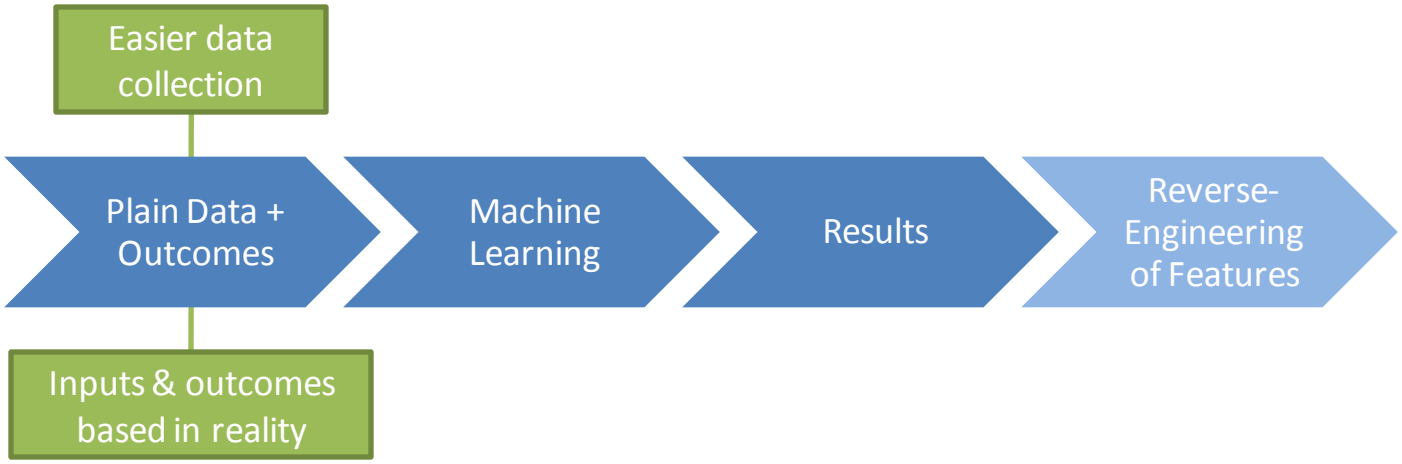



# Models vs. ANN

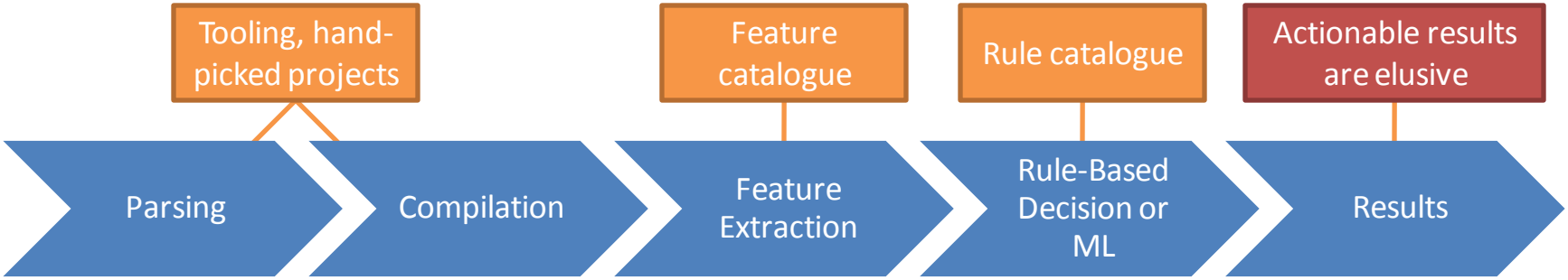


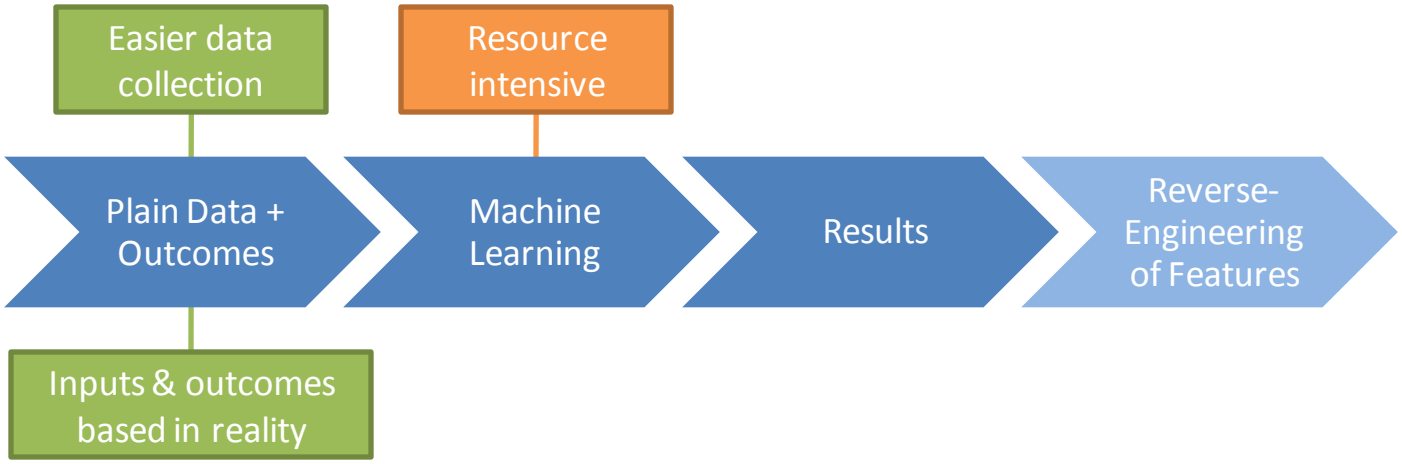
# Models vs. ANN



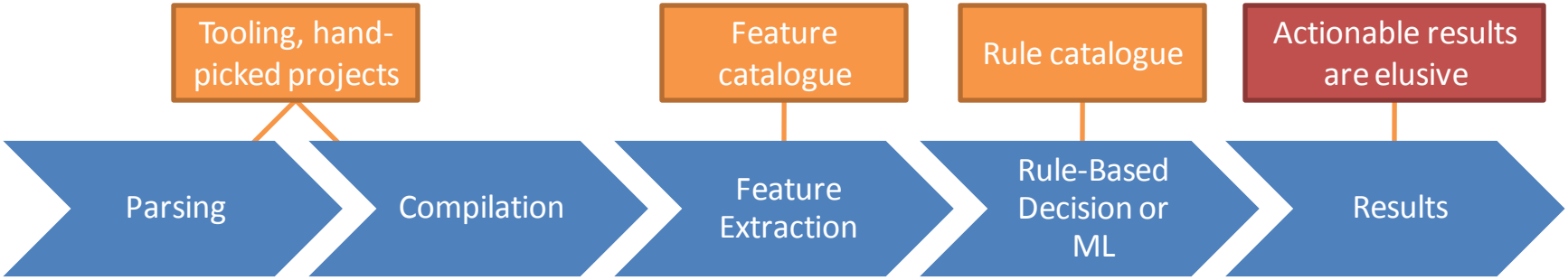



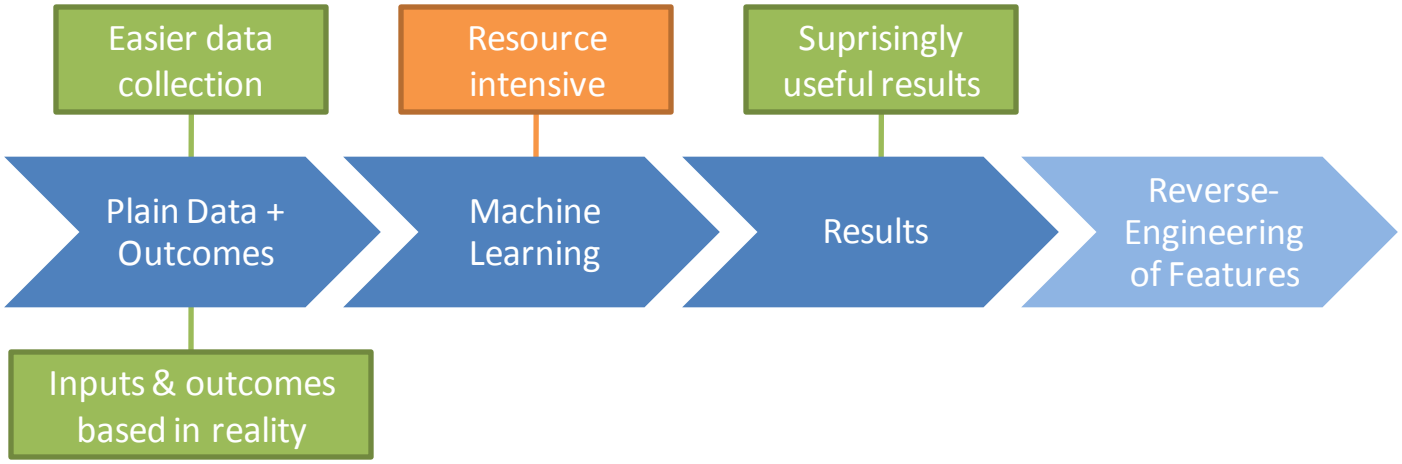
# Models vs. ANN



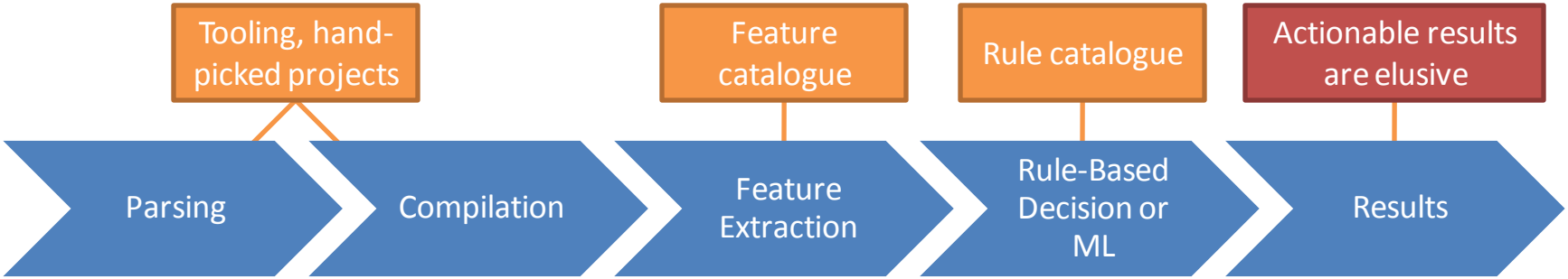



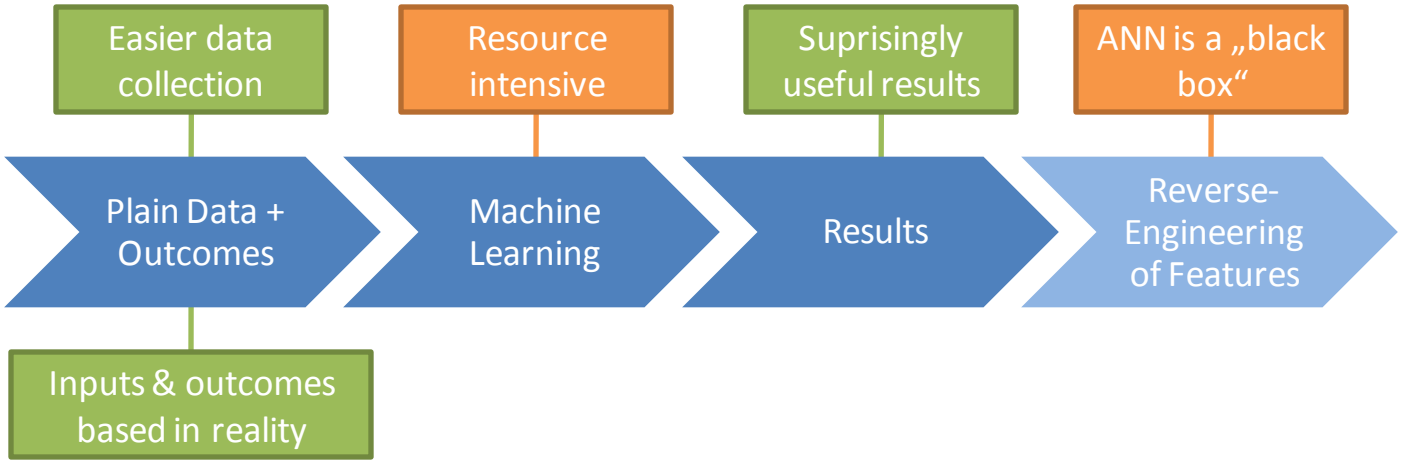
# Models vs. ANN



# Models vs. ANN



# Models vs. ANN

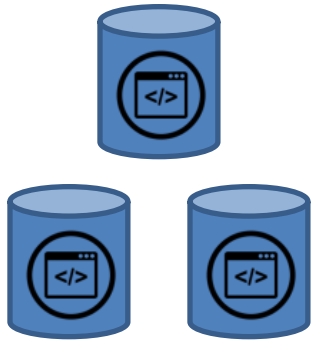
Instead of trying to model the complexity of source code, let the machine figure out what matters to make useful predictions

# First Steps...

Based on work by Andrej Karpathy (http://karpathy.github.io):
train an RNN to predict the next character
in a sequence of Java code

# First Steps...

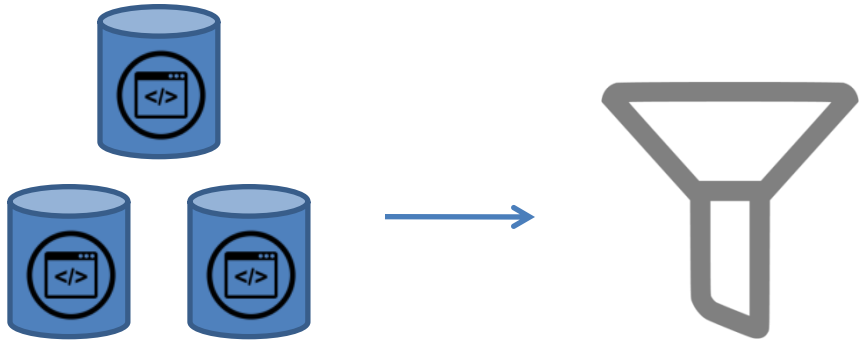
Based on work by Andrej Karpathy (http://karpathy.github.io):
train an RNN to predict the next character
in a sequence of Java code



Clone
693 Apache Projects

# First Steps...

Based on work by Andrej Karpathy (http://karpathy.github.io):
train an RNN to predict the next character
in a sequence of Java code

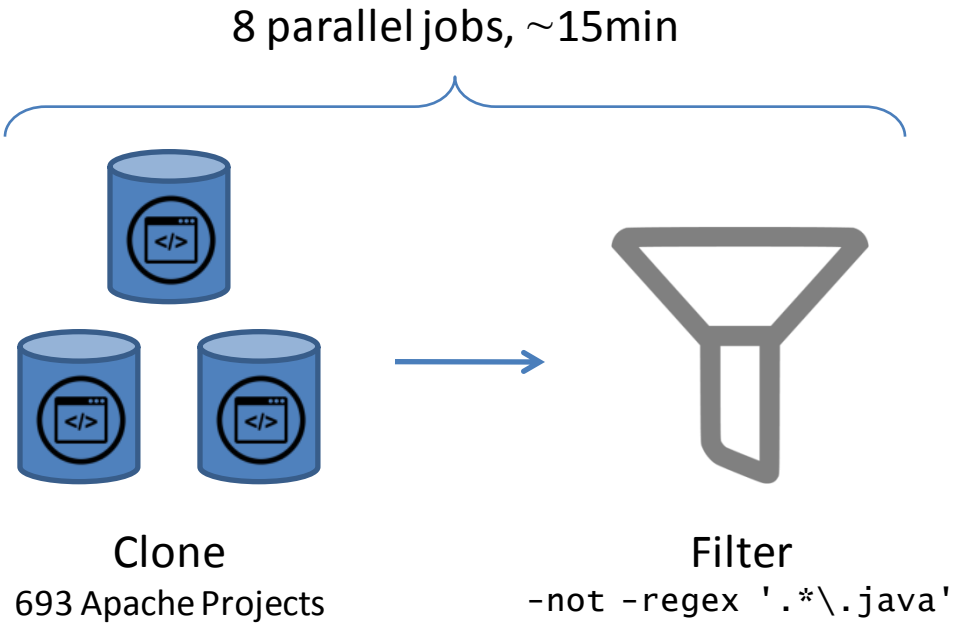


Clone
693 Apache Projects

Filter
`-not -regex '.*\.java'`

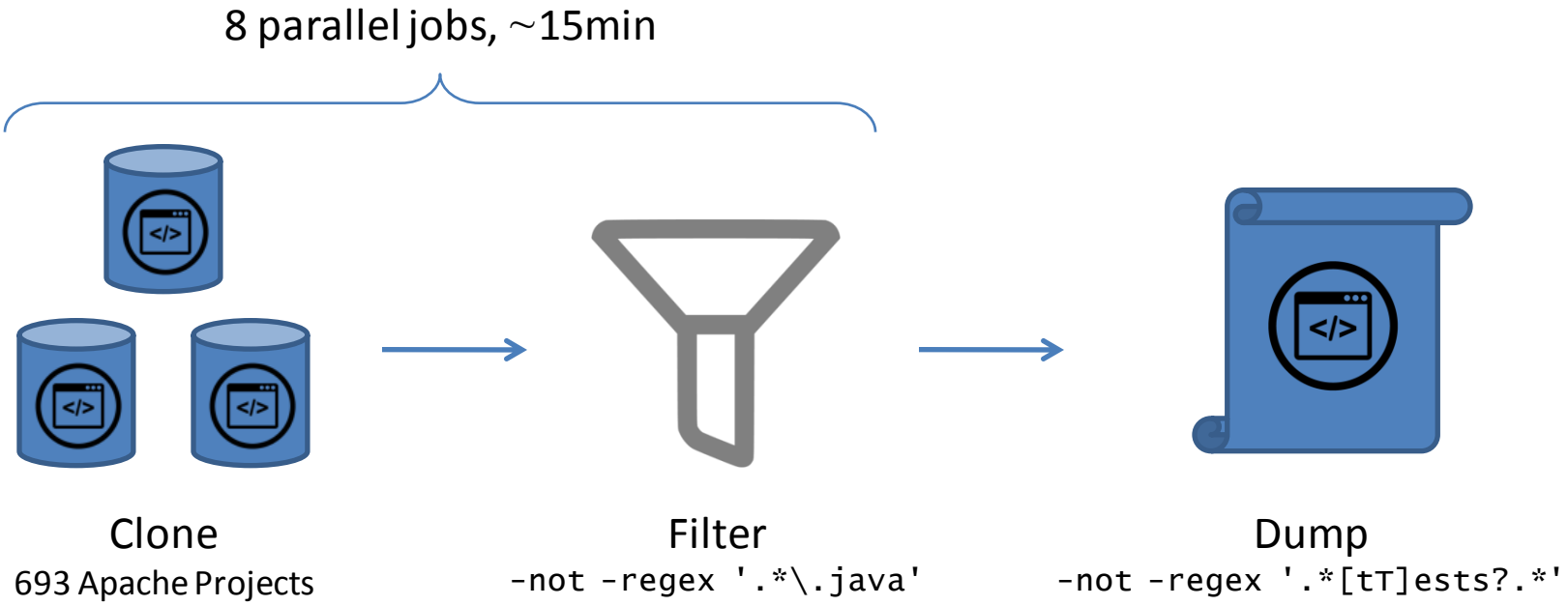
# First Steps...

Based on work by Andrej Karpathy (http://karpathy.github.io):
train an RNN to predict the next character
in a sequence of Java code



# First Steps...

Based on work by Andrej Karpathy (http://karpathy.github.io):
train an RNN to predict the next character
in a sequence of Java code



# First Steps...

Based on work by Andrej Karpathy (http://karpathy.github.io):
train an RNN to predict the next character
in a sequence of Java code

8 parallel jobs, ~15min

Clone
693 Apache Projects

Filter
`-not -regex '.*\.java'`

Dump
`-not -regex '.*[tT]ests?.*'`

= 1.8GB Java Code

# First Steps...

| | |
|---|---|
| Input file size | 1.8 GB |
| Characters | 1 860 428 381 |
| rnn_size | 1500 |
| # Parameters | 46 776 232 |
| Compute time | |

# First Steps...

| | |
|---|---|
| Input file size | 1.8 GB |
| Characters | 1 860 428 381 |
| rnn_size | 1500 |
| # Parameters | 46 776 232 |
| Compute time | **315 days** |

# First Steps...

| | |
|---|---|
| Input file size | 1.8 GB |
| Characters | 1 860 428 381 |
| rnn_size | 1500 |
| # Parameters | 46 776 232 |
| Compute time | **315 days** |

# First Steps...

| Input file size | 1.8 GB |
|---|---|
| Characters | 1 860 428 381 |
| rnn_size | 1500 |
| # Parameters | 46 776 232 |
| Compute time | **315 days** |

| Input file size | 64 MB |
|---|---|
| Characters | 64 042 220 |
| rnn_size | 1500 |
| # Parameters | 46 776 232 |
| Compute time | **60 days** |

| Input file size | 24 MB |
|---|---|
| Characters | 24 014 894 |
| rnn_size | 1024 |
| # Parameters | 21 503 075 |
| Compute time | **1.6 days** |

| Input file size | 6.8 MB |
|---|---|
| Characters | 7 034 943 |
| rnn_size | 580 |
| # Parameters | 7 026 218 |
| Compute time | **6.8 hours** |

# First Steps...

- RNN Training performed using
  - Torch (Lua Scientific Computing Framework)
  - Nvidia GeForce GTX 970 GPU
    - Torch supports CUDA
    - 15x faster than using CPU (i7-3770)!

| | |
|---|---|
| Input file size | 6.8 MB |
| Characters | 7 034 943 |
| rnn_size | 580 |
| # Parameters | 7 026 218 |
| Compute time | **6.8 hours** |

demo

# Next Steps

- RNN:
  - Larger/Longer training (2 weeks)
  - Adapt Character-based RNN for AST tokens
  - Predict next n tokens instead of just one
  - Allow „backtracking“ and „cycling“ in the predictions (using different
- Code completion using RNN has been evaluated but not demonstrated (I think...)
- This is simply a feasible starting point

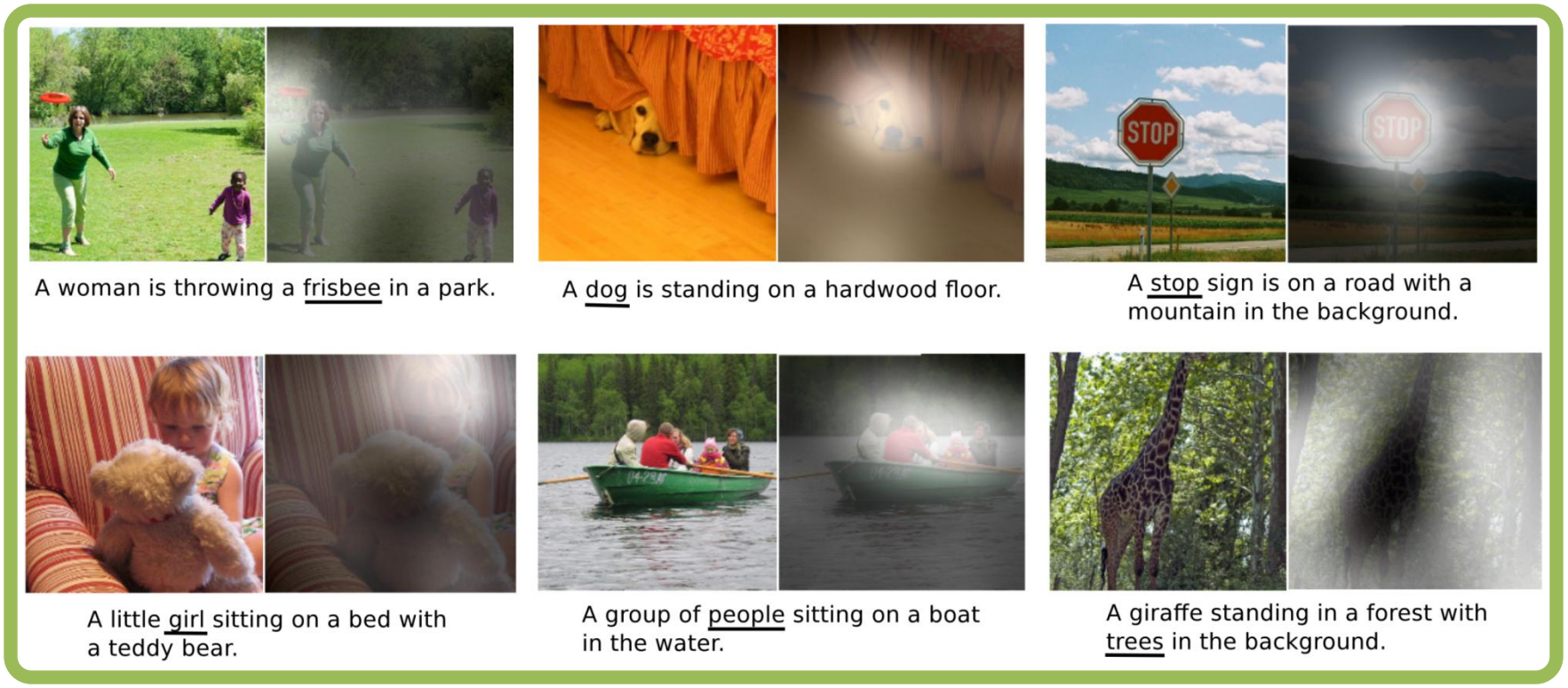
# ANN solutions for SE problems

- Code completion, deobfuscation, synthesis
- Translation
- Classification (concept/feature location)
- Prediction (bugs, changes, effort)
- Detection (memory leaks, antipatterns)

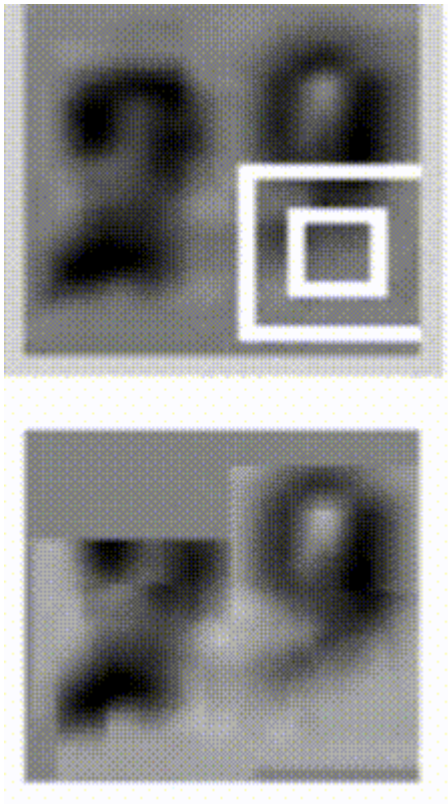
# Concept/Feature location

| | |
|---|---|
| Problem | Feature/Concept location is a hard problem in SE and encompasses many issues, e.g. finding relevant search results during code search, giving useful code suggestions, linking code to bugs/reviews and many more |
| Goal | Enrich and tag code snippets with relevant information. |
| Approach | Use *convolutional* recurrent networks and use *attention steering* to tag varying-size snippets in existing code. |

# Directing „Attention“

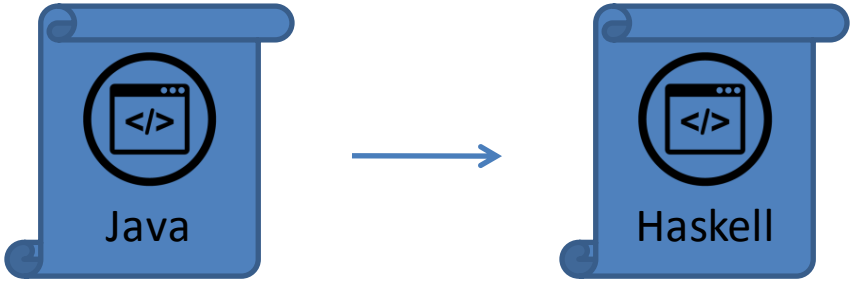


# Directing „Attention“

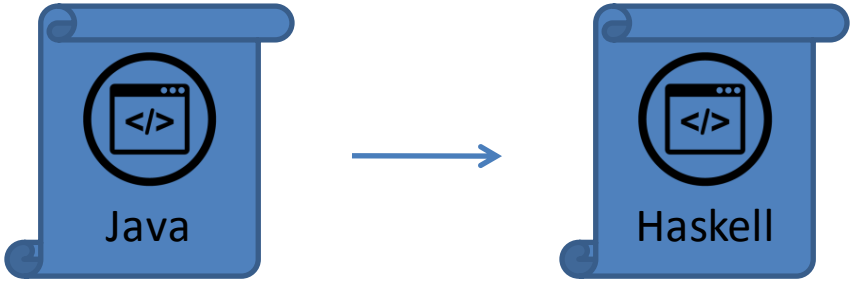


- Teaching a NN to sequentially direct attention
- Example: Reading house numbers left to right
- For code: read code in order of execution/control flow?

# Programming Language Translation

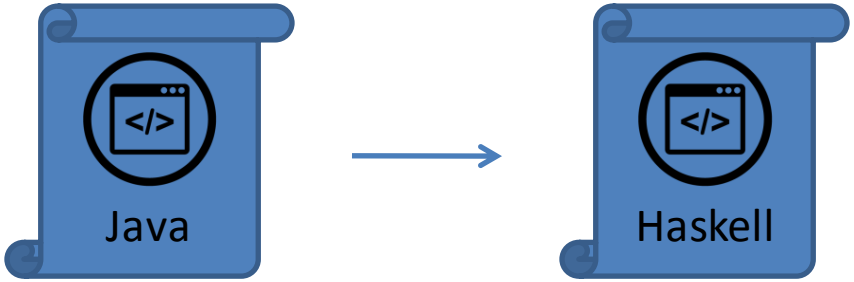


# Programming Language Translation




Train on:
Rosetta Code
Project Euler Solutions
Multi-Platform Projects

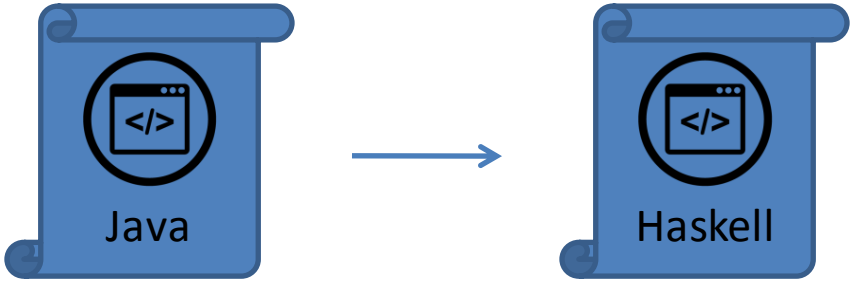
# Programming Language Translation




Train on:
Rosetta Code
Project Euler Solutions
Multi-Platform Projects

Use for:
Low-level Migration & Porting
Learning & Education
„Adaptive“ Rosetta Code

# Programming Language Translation



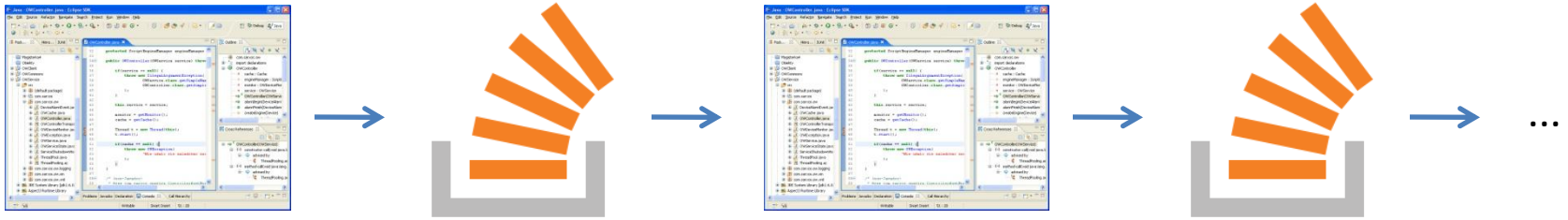

Train on:
Rosetta Code
Project Euler Solutions
Multi-Platform Projects

Use for:
Low-level Migration & Porting
Learning & Education
„Adaptive“ Rosetta Code

Also consider: Pseudo-Code ↔ Source Code
Natural Language ↔ Source Code

# Guided Code Synthesis

| | |
|---|---|
| Problem | When writing Code, developers frequently access SO, Google or other source code within the project, causing a large number of context switches |



# Guided Code Synthesis

| | |
|---|---|
| Problem | When writing Code, developers frequently access SO, Google or other source code within the project, causing a large number of context switches |
| Goal | Keep the developer in the IDE.<br>Use *intent* to steer the provided suggestions. |

# Guided Code Synthesis

```
import java.util.ArrayList;

public class Test {
  main(

}
```

# Guided Code Synthesis

```
import java.util.ArrayList;

public class Test {
  main(public static void main(String[] args) {

}
```

# Guided Code Synthesis

```
import java.util.ArrayList;

public class Test {
  public static void main(String[] args) {
    new list
  }
}
```

# Guided Code Synthesis

```
import java.util.ArrayList;

public class Test {
  public static void main(String[] args) {
    new listArrayList list = new ArrayList<>();
  }
}
```

# Guided Code Synthesis

```
import java.util.ArrayList;

public class Test {
  public static void main(String[] args) {
    new list stringArrayList<String> list = new ArrayList<>();
  }
}
```

# Guided Code Synthesis

```
import java.util.ArrayList;

public class Test {
  public static void main(String[] args) {
    ArrayList<String> list = new ArrayList<>();
  }
}
```

# Guided Code Synthesis

```
import java.util.ArrayList;

public class Test {
  public static void main(String[] args) {
    ArrayList<String> list = new ArrayList<>();
    write CSV file
  }
}
```

# Guided Code Synthesis

```
import java.util.ArrayList;

public class Test {
  public static void main(String[] args) {
    ArrayList<String> list = new ArrayList<>();
    write CSV fileCSVWriter writer = new CSVWriter(new
FileWriter("file.csv"), '\t');
    // feed in your array (or convert your data to an array)
    String[] entries = "first#second#third".split("#");
    writer.writeNext(entries);
    writer.close();
  }
}
```

# Guided Code Synthesis

```
import java.util.ArrayList;

public class Test {
  public static void main(String[] args) {
    ArrayList<String> list = new ArrayList<>();
    CSVWriter writer = new CSVWriter(new FileWriter("file.csv"), '\t');
    // feed in your array (or convert your data to an array)
    String[] entries = "first#second#third".split("#");
    writer.writeNext(entries);
    writer.close();
  }
}
```

# Guided Code Synthesis

```
import java.util.ArrayList;

public class Test {
  public static void main(String[] args) {
    ArrayList<String> list = new ArrayList<>();
    CSVWriter writer = new CSVWriter(new FileWriter("file.csv"), '\t');
    loop over argsfor (String s : args) {

    }
    writer.writeNext(entries);
    writer.close();
  }
}
```

# Guided Code Synthesis

```
import java.util.ArrayList;

public class Test {
  public static void main(String[] args) {
    ArrayList<String> list = new ArrayList<>();
    CSVWriter writer = new CSVWriter(new FileWriter("file.csv"), '\t');
    loop over argsfor (String s : args) {
      writer.writeNext(s);

    }
    writer.close();
  }
}
```

# Guided Code Synthesis

```
import java.util.ArrayList;

public class Test {
  public static void main(String[] args) {
    ArrayList<String> list = new ArrayList<>();
    CSVWriter writer = new CSVWriter(new FileWriter("file.csv"), '\t');
    for (String s : args) {
      writer.writeNext(s);
    }
    writer.close();
  }
}
```

# Guided Code Synthesis

```
import java.util.ArrayList;

public class Test {
  public static void main(String[] args) {
    ArrayList<String> list = new ArrayList<>();
    CSVWriter writer = new CSVWriter(new FileWriter("file.csv"), '\t');
    for (String s : args) {
      writer.writeNext(s);
    }
    writer.close();
  }
}
```

# Specifics of SE-Problems

# Specifics of SE-Problems

- Code is not just data (like an image) but a data *transformer*

# Specifics of SE-Problems

- Code is not just data (like an image) but a data *transformer*
- We don‘t want to generate superfluous code

# Specifics of SE-Problems

- Code is not just data (like an image) but a data *transformer*
- We don‘t want to generate superfluous code
- Source code does not necessarily have a fixed „ordering“ in the data

# Specifics of SE-Problems

- Code is not just data (like an image) but a data *transformer*
- We don‘t want to generate superfluous code
- Source code does not necessarily have a fixed „ordering“ in the data
- Source code is almost „noise free“ and highly structured (more than natural language)

# Specifics of SE-Problems

- Code is not just data (like an image) but a data *transformer*
- We don‘t want to generate superfluous code
- Source code does not necessarily have a fixed „ordering“ in the data
- Source code is almost „noise free“ and highly structured (more than natural language)
- Source code is richer (we can use parsers & compilers to enrich input data)

# Specifics of SE-Problems

- Code is not just data (like an image) but a data *transformer*
- We don‘t want to generate superfluous code
- Source code does not necessarily have a fixed „ordering“ in the data
- Source code is almost „noise free“ and highly structured (more than natural language)
- Source code is richer (we can use parsers & compilers to enrich input data)
- We can automatically rate generated output

# Summary

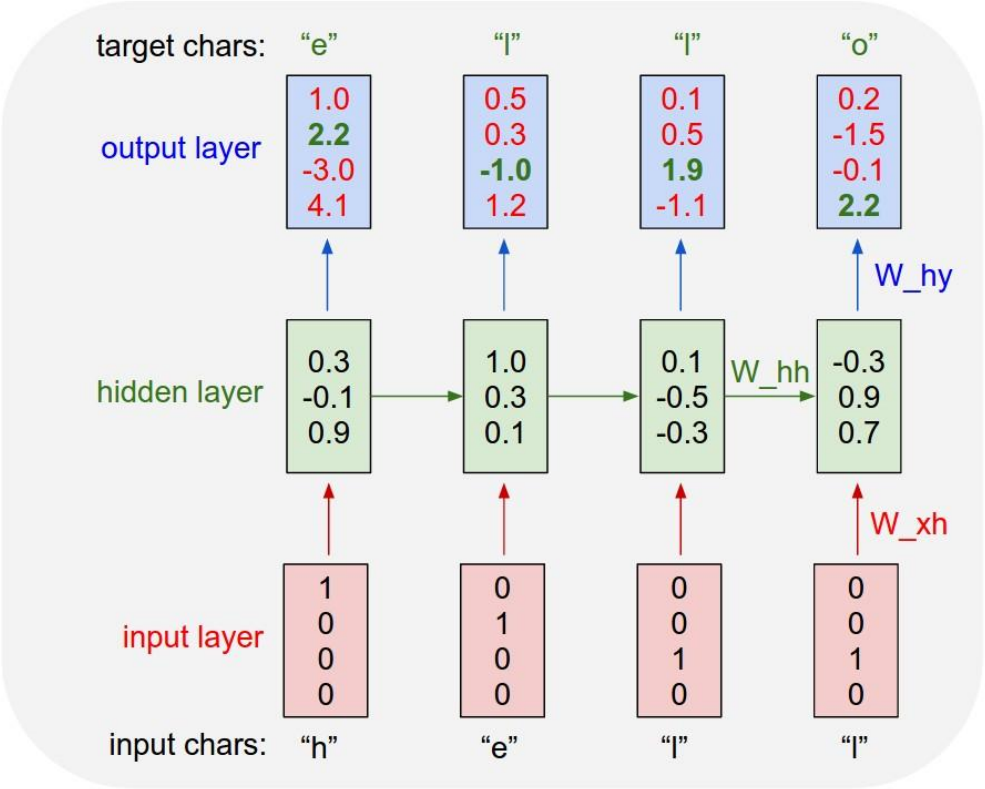
Thank you

# Recurrent Neural Networks

- Output does not only depend on given Input, but also on the **entire history** of inputs
- E.g. Long Short-Term Memory (LSTM) Networks

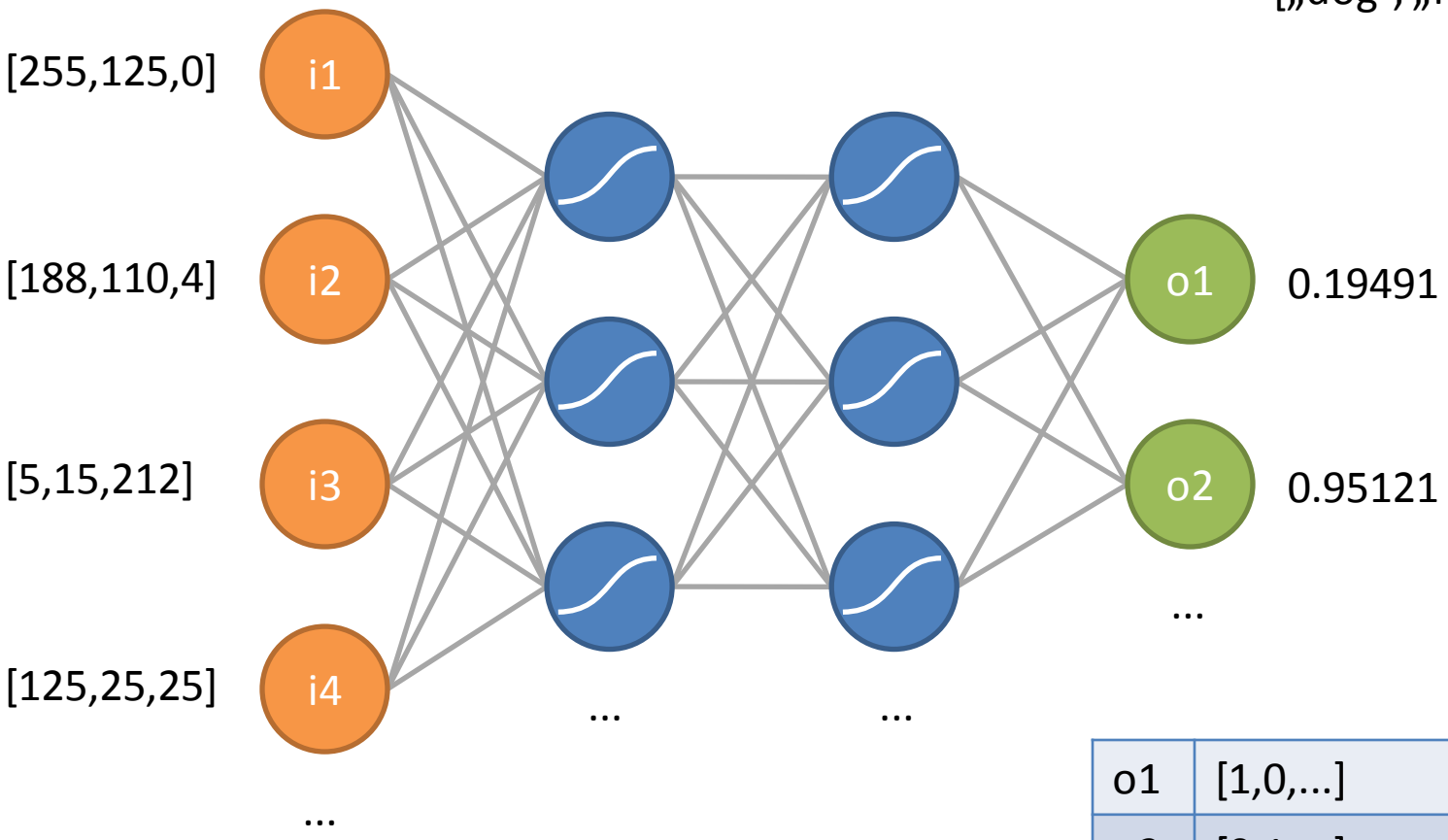
# Recurrent Neural Networks

- Output does not only depend on given Input, but also on the **entire history** of inputs
- E.g. Long Short-Term Memory (LSTM) Networks



# ANN work on numerical values

Classes:
[„dog“, „frog“,...]




| | |
|---|---|
| o1 | [1,0,...] |
| o2 | [0,1,...] |
| ... | ... |

# Guided Code Synthesis

| | |
|---|---|
| Problem | When writing Code, developers frequently access SO, Google or other source code within the project, causing a large number of context switches |
| Goal | Keep the developer in the IDE by continously providing appropriate feedback and code suggestions. The developer should be able to input *intent* to steer the provided suggestions. |
| Approach | Use multi-layer recurrent networks that draw on multiple sources, i.e. user-provided code tokens, tags and task context. Use existing tool (Adapt) for translating user input to *intent*. |

# Concept tagging

| | |
|---|---|
| Problem | Feature/Concept location is a hard problem in SE and encompasses many issues, e.g. finding relevant search results during code search, giving useful code suggestions, linking code to bugs/reviews and many more |
| Goal | Enrich and tag code snippets with relevant information. |
| Approach | Use *convolutional* recurrent networks and use *attention steering* to tag varying-size snippets in existing code. |

# Co-Change suggestions

| | |
|---|---|
| Problem | Developers often need to change related code that is not connected explicitely (especially true in weak typed languages) |
| Goal | When the developer changes some code, suggest other locations and maybe even the required code |
| Approach | Pair-wise training on patches: Given one patch as input, expect another patch as output – either coarse (file and location within file) or fine (file, location and code suggestion). Take temporal distance into account (same commit == very close). Maybe include commit message in output. |

# „Neural Linter“

| | |
|---|---|
| Problem | Linters & tools like findbugs are useful but often give too many results that are ignored. Also, different development teams have different priorities and coding styles. |
| Goal | Give only actionable, relevant and tailored information to developers |
| Approach | Train an ANN on existing data regarding proposed fixes and performed fixes to be used as a filter over new predictions. The system could learn over time for a single user as well. |

# Programming language translation

| | |
|---|---|
| Problem | Upgrading legacy systems and writing the same code for different platforms can be difficult |
| Goal | Create template source code to serve as a starting point for manual translation |
| Approach | Use rosetta code problem solutions and multi-platform projects to train one model per language pair. |
| Existing? | Mostly rule based, i.e. compilers |

# Source Code from Pseudocode

| | |
|---|---|
| Problem | Writing code is time consuming and difficult |
| Goal | Create valid source code from pseudo code definitions |
| Approach | Create a corpus of pseudo code to source code translations and train an ANN in order to create new translations |
| Existing? | Again, mostly rule-based with many restriction on the input syntax |

# Source Code from natural language

| | |
|---|---|
| Problem | Writing code is time consuming and difficult |
| Goal | Create valid source code from human natural language. |
| Approach | Add a speech recognition tool to guided code synthesis. |

# Guided Code Synthesis

```
import java.util.ArrayList;

public class Test {
  main(

}
```

User inputs:
main(

# Guided Code Synthesis

```
import java.util.ArrayList;

public class Test {
  main(public static void main(String[] args) {

}
```

User inputs:
main(

# Guided Code Synthesis

```
import java.util.ArrayList;

public class Test {
  public static void main(String[] args) {
    new list
  }
}
```

User inputs:
main(<TAB>
new list

# Guided Code Synthesis

```
import java.util.ArrayList;

public class Test {
  public static void main(String[] args) {
    new listArrayList list = new ArrayList<>();
  }
}
```

User inputs:
main(<TAB>
new list string

# Guided Code Synthesis

```
import java.util.ArrayList;

public class Test {
  public static void main(String[] args) {
    new list stringArrayList<String> list = new ArrayList<>();
  }
}
```

User inputs:
main(<TAB>
new list string

# Guided Code Synthesis

```
import java.util.ArrayList;

public class Test {
  public static void main(String[] args) {
    ArrayList<String> list = new ArrayList<>();
  }
}
```

User inputs:
main(<TAB>
new list string<TAB>

# Guided Code Synthesis

```
import java.util.ArrayList;

public class Test {
  public static void main(String[] args) {
    ArrayList<String> list = new ArrayList<>();
    write CSV file
  }
}
```

User inputs:

```
main(<TAB>
new list string<TAB>
write CSV file
```

# Guided Code Synthesis

```
import java.util.ArrayList;

public class Test {
  public static void main(String[] args) {
    ArrayList<String> list = new ArrayList<>();
    write CSV fileCSVWriter writer = new CSVWriter(new
FileWriter("file.csv"), '\t');
    // feed in your array (or convert your data to an array)
    String[] entries = "first#second#third".split("#");
    writer.writeNext(entries);
    writer.close();
  }
}
```

User inputs:
main(<TAB>
new list string<TAB>
write CSV file

# Guided Code Synthesis

```
import java.util.ArrayList;

public class Test {
  public static void main(String[] args) {
    ArrayList<String> list = new ArrayList<>();
    CSVWriter writer = new CSVWriter(new FileWriter("file.csv"), '\t');
    // feed in your array (or convert your data to an array)
    String[] entries = "first#second#third".split("#");
    writer.writeNext(entries);
    writer.close();
  }
}
```

User inputs:
main(<TAB>
new list string<TAB>
write CSV file<TAB>

# Guided Code Synthesis

```
import java.util.ArrayList;

public class Test {
  public static void main(String[] args) {
    ArrayList<String> list = new ArrayList<>();
    CSVWriter writer = new CSVWriter(new FileWriter("file.csv"), '\t');


    writer.writeNext(entries);
    writer.close();
  }
}
```

User inputs:
main(<TAB>
new list string<TAB>
write CSV file<TAB>
SHIFT-ALT-D-J-2

# Guided Code Synthesis

```
import java.util.ArrayList;

public class Test {
  public static void main(String[] args) {
    ArrayList<String> list = new ArrayList<>();
    CSVWriter writer = new CSVWriter(new FileWriter("file.csv"), '\t');
    loop over argsfor (String s : args) {

    }
    writer.writeNext(entries);
    writer.close();
  }
}
```

User inputs:
main(<TAB>
new list string<TAB>
write CSV file<TAB>
SHIFT-ALT-D-J-2
loop over args

# Guided Code Synthesis

```
import java.util.ArrayList;

public class Test {
  public static void main(String[] args) {
    ArrayList<String> list = new ArrayList<>();
    CSVWriter writer = new CSVWriter(new FileWriter("file.csv"), '\t');
    loop over argsfor (String s : args) {
      writer.writeNext(s);
      writer.close();
    }
  }
}
```

User inputs:
main(<TAB>
new list string<TAB>
write CSV file<TAB>
SHIFT-ALT-D-J-2
loop over args<SHIFT-M-I-2>

# Guided Code Synthesis

```
import java.util.ArrayList;

public class Test {
  public static void main(String[] args) {
    ArrayList<String> list = new ArrayList<>();
    CSVWriter writer = new CSVWriter(new FileWriter("file.csv"), '\t');
    for (String s : args) {
      writer.writeNext(s);
      writer.close();
    }
  }
}
```

User inputs:
main(<TAB>
new list string<TAB>
write CSV file<TAB>
SHIFT-ALT-D-J-2
loop over args<SHIFT-M-I-2><TAB>