ORACLE®

Biscotti and Cannoli

An Initial Exploration into Machine Learning for the Purposes of Finding Bugs in Source Code

Tim Chappell*, Cristina Cifuentes, Paddy Krishnan
Queensland University of Technology*, Oracle Labs
November 15, 2016

Oracle Labs

ORACLE

Project Overview

- Imagine if machine learning could detect bugs for us in software
  - With good precision
  - With good recall
  - With good performance
  - And beat Parfait and other static code analysis tools at finding bugs in software

- This Friday Project is an investigation into what is feasible in this space
  - Project started in February 2016

ORACLE

Machine Learning is the subfield of computer science that “gives computers the ability to learn without being explicitly programmed” (Arthur Samuel, 1959)

– Wikipedia

Machine Learning Approaches

Supervised Learning

- The learning algorithm is given example inputs and their desired outputs, with the goal to learn a general rule that maps inputs to outputs

Unsupervised Learning

- The learning algorithm infers structure in its inputs to produce the outputs of interest

ORACLE

Machine Learning Approaches

Supervised Learning

- The learning algorithm is given example inputs and their desired outputs, with the goal to learn a general rule that maps inputs to outputs
- Two tools
  - Biscotti
  - Cannoli

Unsupervised Learning

- The learning algorithm infers structure in its inputs to produce the outputs of interest

ORACLE

Supervised Learning – Classifiers and Decision Trees

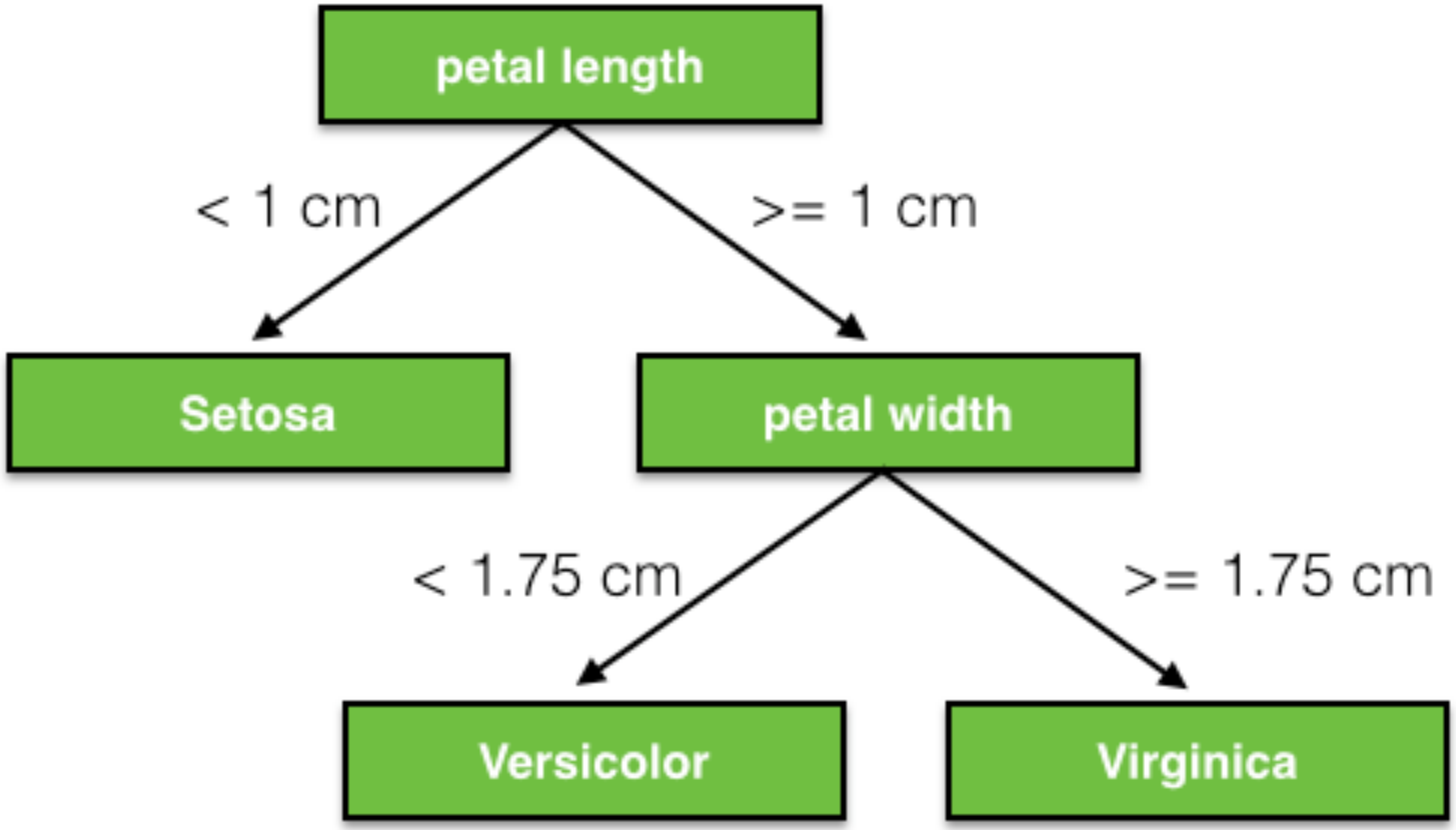
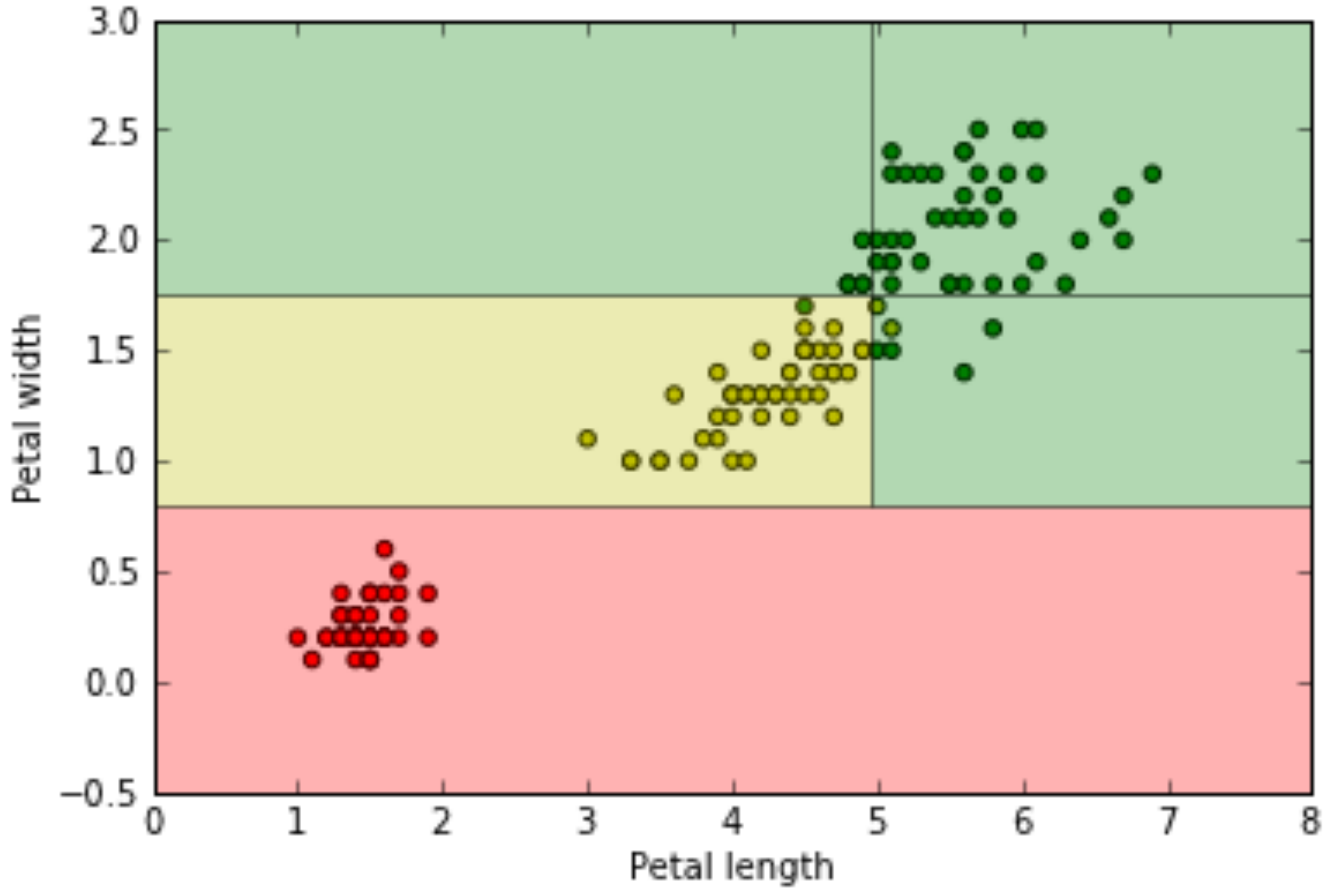


Diagram from: http://sebastianraschka.com/images/blog/2014/intro_supervised_learning/decision_tree_1.png

2D Decision Boundary



http://statweb.stanford.edu/~jtaylo/courses/stats202/_images/trees_fig_03.png

ORACLE®

Iris Dataset Example

- Made use of two petal **features** (length and width)
- Classified into three **classes** of Irises (setosa, versicolor, virginica)

ORACLE

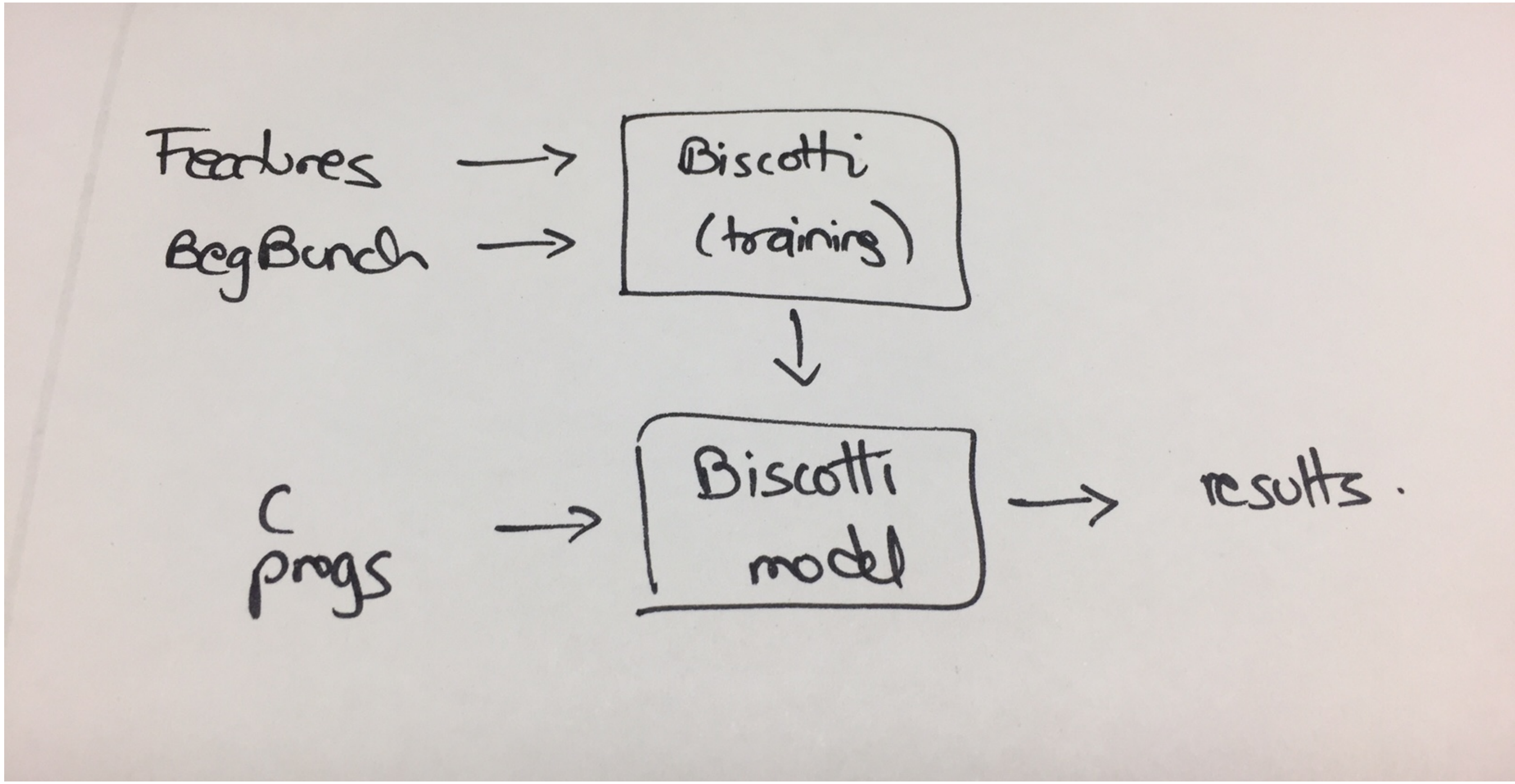
Abstracting The Iris Dataset Example

- **Features** are inputs
- **Classes** are outputs
- Dataset needs to contain features and classes

Abstracting The Iris Dataset Example

- **Features** are inputs
- **Classes** are outputs
- Dataset needs to contain features and classes
- For bugs in source code
  - Features == ?
  - Classes == bug type

Biscotti


Features
BegBunch
Biscotti
(training)
Biscotti
model
C
progs
results.

Biscotti's Feature Selection

- Complexity of the code
  - Cyclomatic complexity
  - Def-use chains
  - # edges
  - # knots
  - Length of code
  - Line count
  - Nesting level
  - Vocabulary
  - Function start line
  - Function end line
  - …

- Text features
  - !
  - (
  -)
  - ,
  - 00
  - 1
  - …
  - FILE
  - …
  - Input
  - Logged
  - …

- Intermediate Code instruction frequency
  - add
  - alloca
  - and
  - ashr
  - bitcast
  - br
  - call
  - extractvalue
  - fadd
  - …

ORACLE

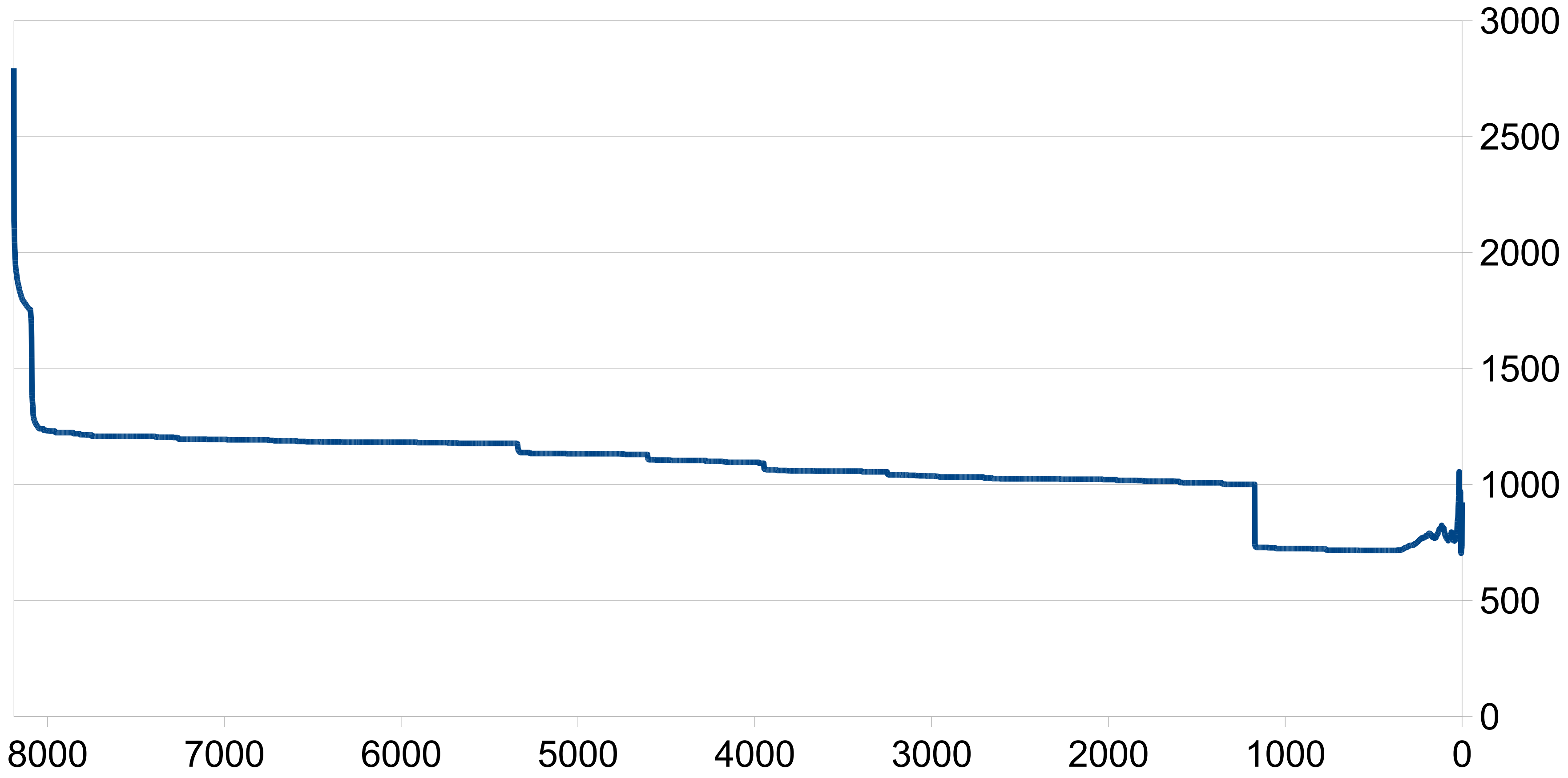
Biscotti's Feature Selection

- Intermediate Code 2-grams
  - alloca-alloca
  - store-store
  - store-br
  - br-load
  - load-icmp
  - icomp-br
  - br-br
  - …

- Clang –analyze output
  - Array-subscript-is-undefined
  - Bad-free
  - Dead-assignment
  - Dead-increment
  - Dereference-of-null-pointer
  - Double-free
  - Function-call-argument-is-an-uninitialized-value
  - Memory-leak
  - Out-of-bound-array-access
  - …

- Output from other Static Code Analysis tools
  - Parfait
  - Splint
  - UNO

Feature Selection – Dimensionality Reduction



8,190 features reduced to 500

Feature Selection – Dimensionality Reduction

- LOONNE: leave one out nearest neighbour error
  - Removes the least distinguishing feature at each step by minimising the global error

Given a feature set FS,

GlobalError(FS) = Sum of all misclassifications for FS

LOONNE removes feature f if

for all other features f’, GlobalError(FS-{f}) > GlobalError(FS-{f’})

Biscotti's Classification Algorithm

- Random Forests
  - Forest of 100 randomly-seeded decision trees using random subsets of the feature set
  - The outcomes of the decision trees are combined to produce a single outcome for each result
  - Useful when no natural probabilistic distribution amongst features
- Granularity of analysis: function level
  - Line number level too fine for initial experimentation

ORACLE

Training and Test Datasets: BegBunch's Accuracy Suites

Bugs are marked up in the suites

BegBunch Suite	Type of Benchmark	Average Non-Commented Lines of Code	# Functions	# and Types of Bugs
Cigital	Synthetic	15	50	Buffer overruns: 1709 Memory leaks: 196 Uninitialised vars: 131
Samate	Synthetic	20	2,366	
Iowa	Synthetic	31	1,686	
OracleLabs*	Real	917	547	

Trained with 4-fold cross-validation over test datasets

* These bug kernels were extracted from open source code, including relevant flow of control.

ORACLE

Results ML (Biscotti) vs Static Code Analysis Tools

Type of Bug	Splint		Parfait		Biscotti 500 features	
Buffer overrun	581/999 TP (58%)	343 FP	885/999 (89%)	14 FP	910/999 (91%)	262 FP
Memory leak	-		9/42 (21%)	10 FP	17/42 (40%)	3 FP
Uninitialised variable	12/15 TP (80%)	54 FP	13/15 (87%)	11 FP	8/15 (53%)	0 FP

Evaluated using 4-fold cross-validation over BegBunch dataset

ORACLE

What Did Biscotti Learn?

- Top 10 features
  - [Parfait] buffer overflow
  - [Parfait] read outside array bounds
  - [Splint] fresh storage not released before return
  - [Text] ,
  - [Complexity] function end line
  - [Parfait] uninitialised variable
  - [Splint] function exported but not used outside
  - [Splint] for body not block
  - [Text] contents
- Training datasets have high number of synthetic benchmarks
  - Biscotti learnt to rely on features that don't make sense (e.g., end of line)
- None of the features are representative of a bug

Results ML (Biscotti) vs Static Code Analysis Tools

Type of Bug	Splint		Parfait		Biscotti			
					500 features		1-&2-grams + complexity features (553 features)	
Buffer overrun	581/999 TP (58%)	343 FP	885/999 (89%)	14 FP	910/999 (91%)	262 FP	23/999 (2%)	5 FP
Memory leak	-		9/42 (21%)	10 FP	17/42 (40%)	3 FP	5/42 (12%)	0 FP
Uninitialised variable	12/15 TP (80%)	54 FP	13/15 (87%)	11 FP	8/15 (53%)	0 FP	0/15 (0%)	0 FP

Evaluated using 4-fold cross-validation over BegBunch dataset

ORACLE

Biscotti Conclusions

- Need more datasets of representative bugs; marked up
  - I.e., not synthetic benchmarks

- The crux of supervised learning is determining the **right set of features**
  - What features make a bug a bug?

“Deep Learning succeeds when it’s difficult to figure out what features you want to use in your classifier”

Machine Learning Approaches

Supervised Learning

- The learning algorithm is given example inputs and their desired outputs, with the goal to learn a general rule that maps inputs to outputs

- Two tools
  - Biscotti
  - Cannoli

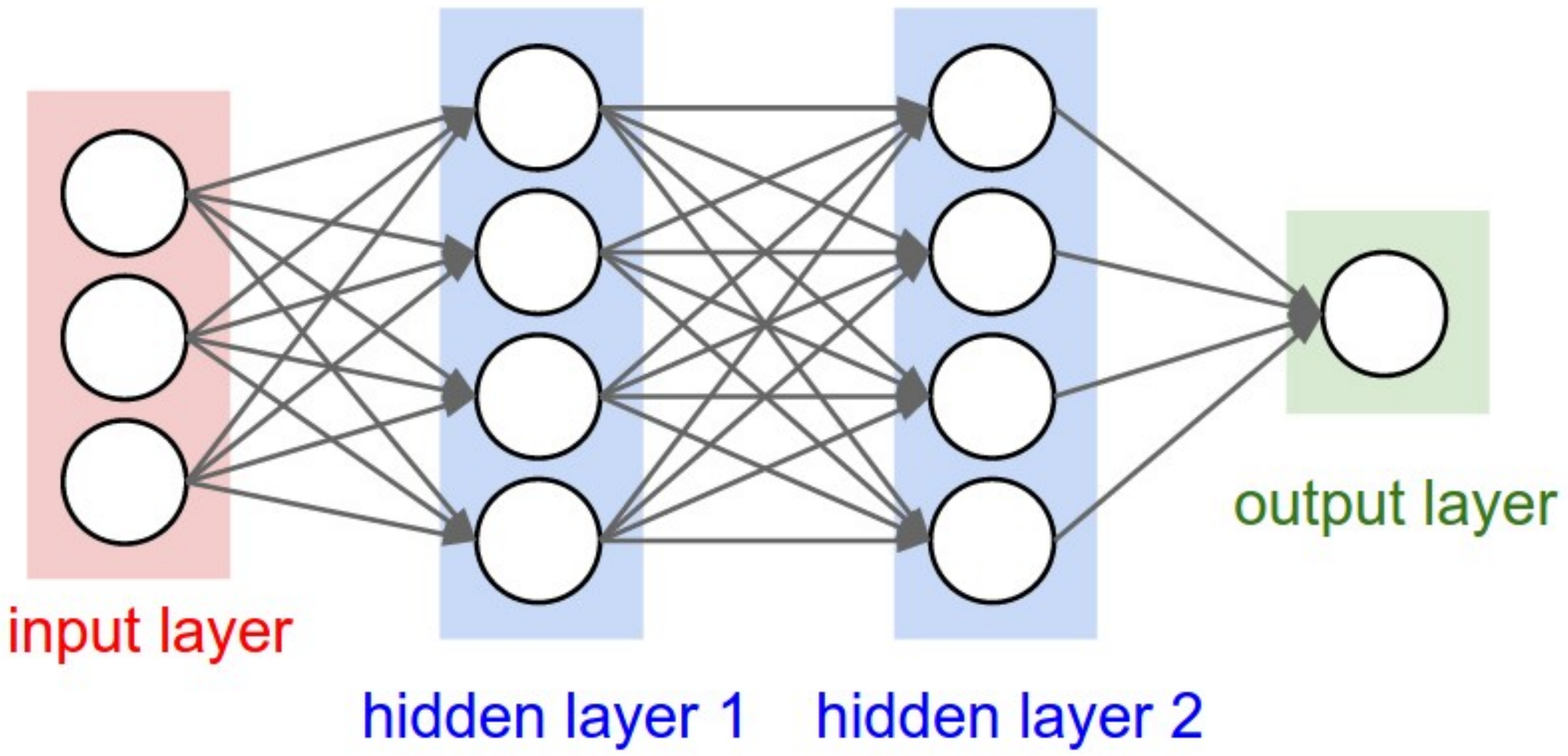
Unsupervised Learning

- The learning algorithm infers structure in its inputs to produce the outputs of interest

ORACLE

Supervised Learning – Convolutional Neural Networks

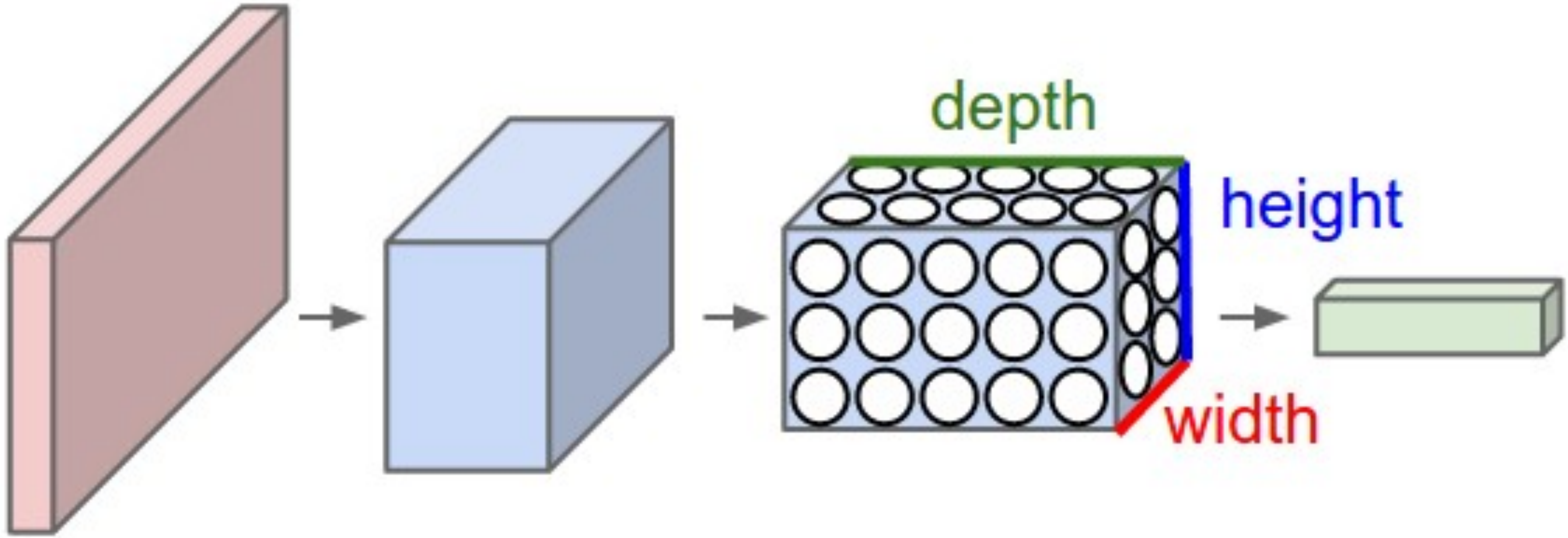
3-layer neural network



http://cs231n.github.io/assets/nn1/neural_net2.jpeg

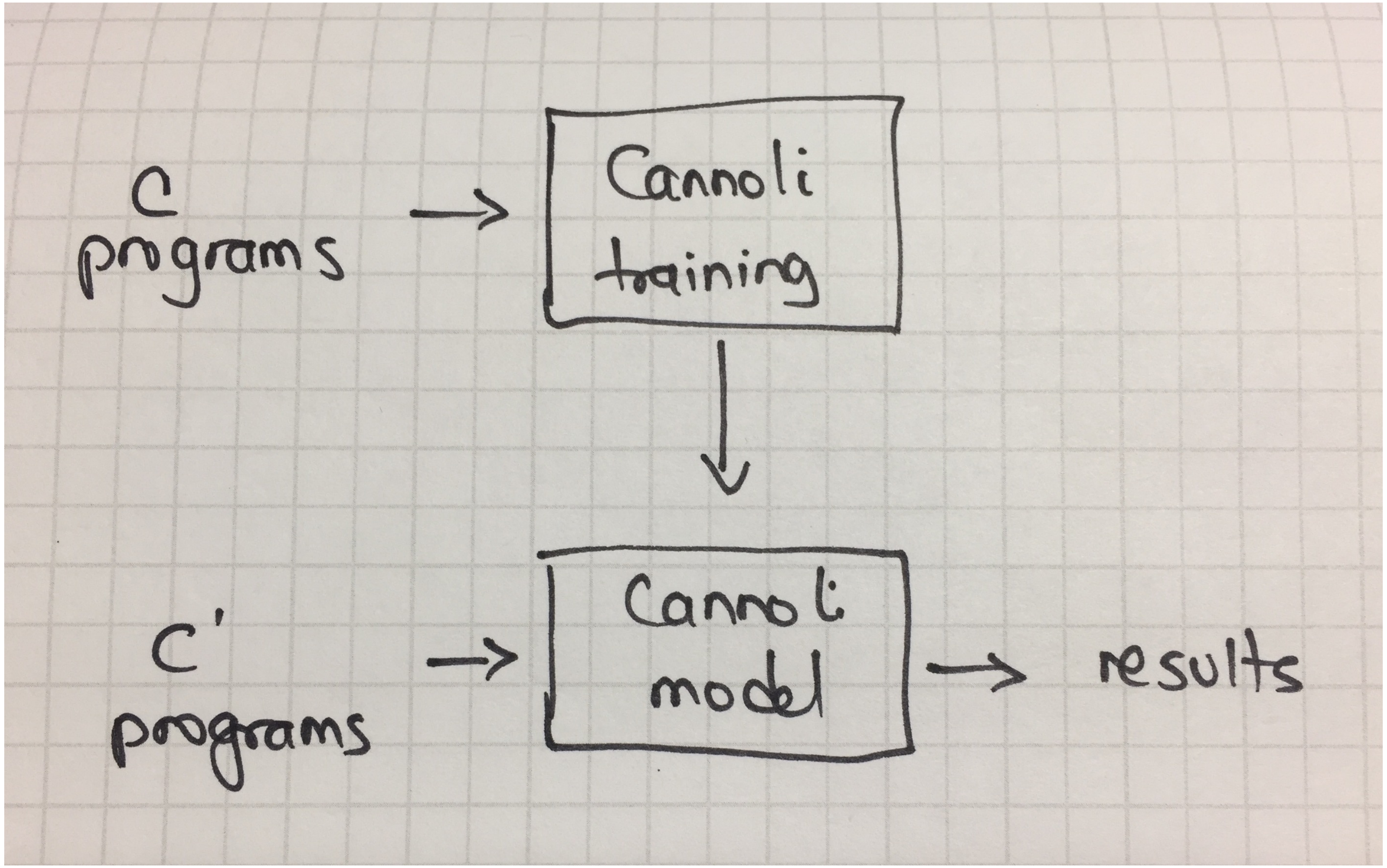
Supervised Learning – Convolutional Neural Networks

Convolutional neural network

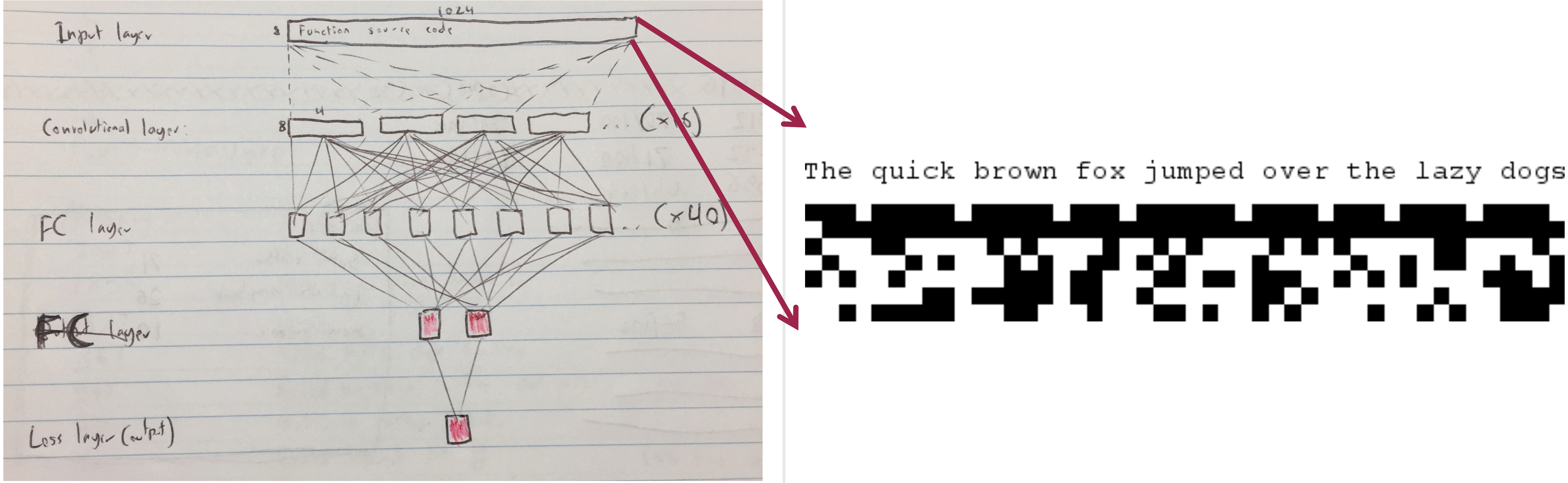


http://cs231n.github.io/assets/cnn/cnn.jpeg

Cannoli


C
programs
Cannoli
training
C'
programs
Cannoli
model
results

Cannoli's Architecture



Training Dataset: BegBunch’s Scalability Suites

Bugs are not marked up in these suites

BegBunch Suite	Average Non-Commented Lines of Code	# Functions
Calysto	87,636	11,214
OracleLabs	394,739	53,448

ORACLE

Results ML (Cannoli) vs Static Code Analysis Tools

Training on Scalability Suite (50/50 split), testing on OpenSolaris ONNV b93* (no split)

Type of Bug	Parfait v0.4.1	Cannoli
Buffer overrun	221 TP, 81 FP	213/221 TP, 56095 FP
Memory leak	506 TP, 94 FP	497/506 TP, 47414 FP

Training on Scalability Suites using Parfait v1.7.1.3 results as ground truth

* 168,666 functions

ORACLE

Results ML (Cannoli) vs Static Code Analysis Tools

Training on BegBunch's Accuracy Suites (no split), testing on OpenSolaris ONNV b93*

Type of Bug	Parfait v0.4.1	Cannoli
Buffer overrun	221 TP, 81 FP	23/221 TP, 9146 FP
Memory leak	506 TP, 94 FP	0/506 TP, 174 FP
Uninitialised variable	30 TP, 16 FP	0/30 TP, 153 FP

Training on Scalability Suites using Parfait v1.7.1.3 results as ground truth

* 168,666 functions

ORACLE

What Did Cannoli Learn?

?

Cannoli Conclusions

- Image recognition techniques not ideal for source code analysis

- Results from black-box techniques are not very useful for bug detection
  - No bug traces can be derived for developers to understand the results of the tool

Summary Of The State Of The Art

Paper	Venue-Year	Summary
Brun, Ernst	ICSE-04	Properties inferred using both buggy and fixed code
Yamaguchi et al.	ACSAC-12	Extrapolate vulnerabilities from known vulnerabilities using AST representations
ALETHEIA	CCS-14	Statistical analyses to predict "rare" vulnerabilities; tunable to focus on FP elimination/TP detection. Basic features (per Biscotti)
JSNice	POPL-15	Use program dependence graphs and statistical prediction to deobfuscate JavaScript code
Mou et al.	AAAI-16	Convolutional Neural Networks using AST representation to identify code similarities
Wang et al.	ICSE-16	Use Deep Belief Networks and AST representation to detect within project and cross project defects
Greico et al.	CODASPY-16	Use static and dynamic features (state of memory) to detect vulnerabilities

ORACLE

Summary

- Two ML approaches were implemented to find bugs in C code
  - Biscotti: supervised learning using a random forest of decision trees and LOONNE
  - Cannoli: supervised learning using a convolutional neural network
- Both learned "something"
  - But results are tied to the datasets used; i.e., doesn't learn to find bugs in unseen code
- Biscotti captures syntactic features of the program
  - ***Need to capture semantic features***
- **Need a lot more representative data**

Future Plans

1. Create enough data for datasets
   – Representative proportion of buggy vs non-buggy code
   – Representative number of bugs for each bug type of interest
   – Fixed version of each buggy example
2. Explore different approaches to encode semantics
   – Use of buggy vs fixed code to determine features of interest [Ernst'04]
   – Use of recurrent neural network with long short-term memory (LSTM) [Tristan'16]

Q&A

Integrated Cloud

Applications & Platform Services

ORACLE

ORACLE®