SWE 795:

Intersections of Deep Learning & Software Engineering

Spring 2022

George Mason University

Dr. Kevin Moran

Week 1: Course Overview & Intro to Research Area

Welcome to SWE 795!

- **<u>Initial Logistics:</u>**
  - Welcome to the Lecture!
  - This Lecture is being recorded
  - ***<u>Masks are required</u>*** during class time
  - For the safety of everyone, there is ***<u>no eating or drinking</u>*** during lectures/in-class activities
    - However, there will be a 10 minute break in the middle of class.

Introductions

Instructor: Kevin Moran

Education: Ph.D. from William & Mary - 2018

Research Interests: Software Engineering , UI Analysis, Machine Learning

Office Hours: *Monday & Wednesday 1:00pm-2:00pm*

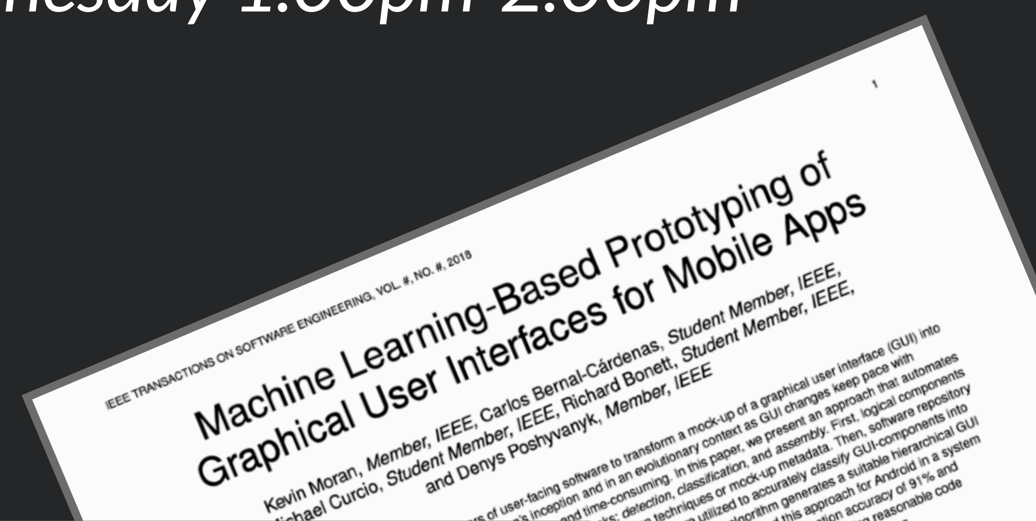
Introductions

Instructor: Kevin Moran

Education: Ph.D. from William & Mary - 2018

Research Interests: Software Engineering , UI Analysis, Machine Learning

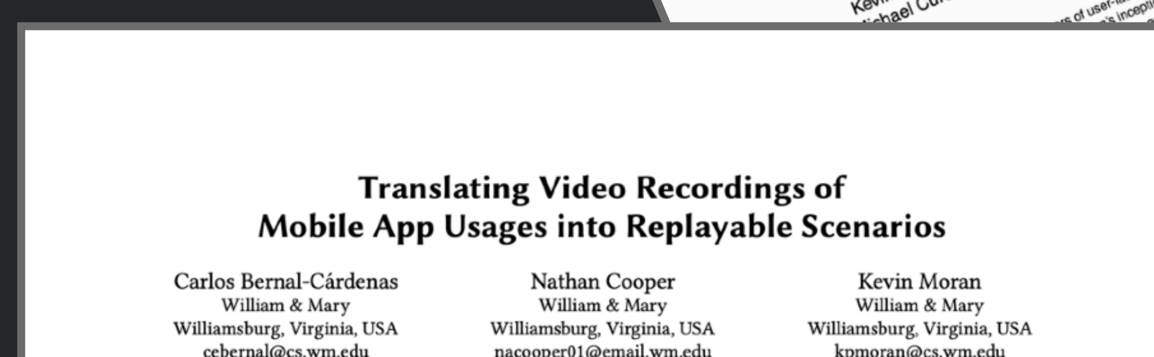
Office Hours: *Monday & Wednesday 1:00pm-2:00pm*



IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. #, NO. #, 2018

Machine Learning-Based Prototyping of Graphical User Interfaces for Mobile Apps

Kevin Moran, *Member, IEEE*, Carlos Bernal-Cárdenas, *Student Member, IEEE*, Michael Curcio, *Student Member, IEEE*, Richard Bonett, *Student Member, IEEE*, and Denys Poshyvanyk, *Member, IEEE*



Translating Video Recordings of Mobile App Usages into Replayable Scenarios

Carlos Bernal-Cárdenas
William & Mary
Williamsburg, Virginia, USA
cebernal@cs.wm.edu

Nathan Cooper
William & Mary
Williamsburg, Virginia, USA
nacooper01@email.wm.edu

Kevin Moran
William & Mary
Williamsburg, Virginia, USA
kpmoran@cs.wm.edu

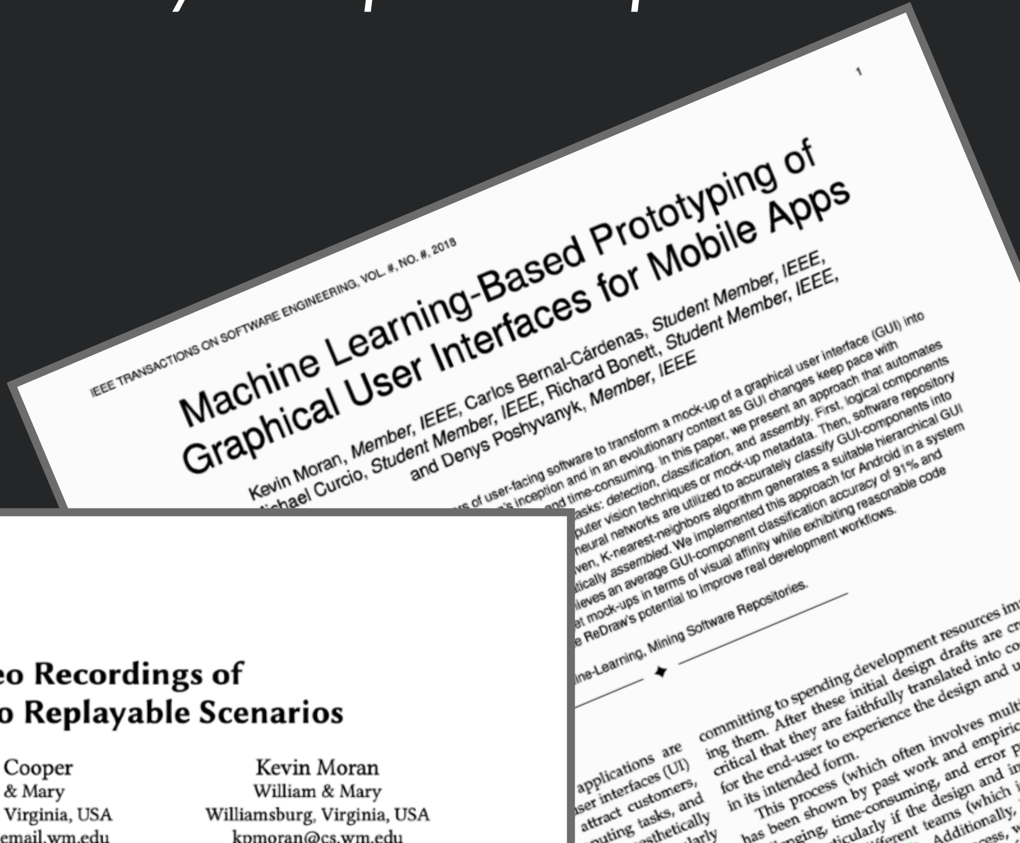
Introductions

Instructor: Kevin Moran

Education: Ph.D. from William & Mary - 2018

Research Interests: Software Engineering , UI Analysis, Machine Learning

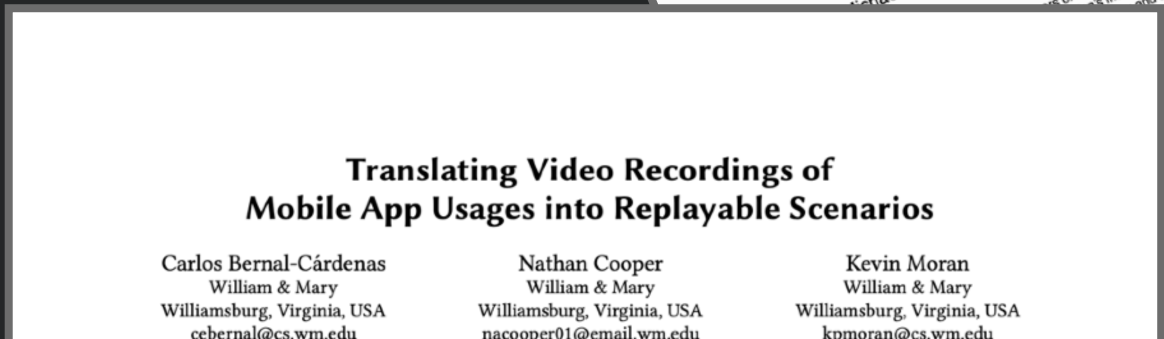
Office Hours: *Monday & Wednesday 1:00pm-2:00pm*



IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. #, NO. #, 2018

Machine Learning-Based Prototyping of Graphical User Interfaces for Mobile Apps

Kevin Moran, *Member, IEEE,* Carlos Bernal-Cárdenas, *Student Member, IEEE,* Michael Curcio, *Student Member, IEEE,* Richard Bonett, *Student Member, IEEE,* and Denys Poshyvanyk, *Member, IEEE*



Translating Video Recordings of Mobile App Usages into Replayable Scenarios

Carlos Bernal-Cárdenas William & Mary Williamsburg, Virginia, USA cebernal@cs.wm.edu	Nathan Cooper William & Mary Williamsburg, Virginia, USA nacooper01@email.wm.edu	Kevin Moran William & Mary Williamsburg, Virginia, USA kpmoran@cs.wm.edu

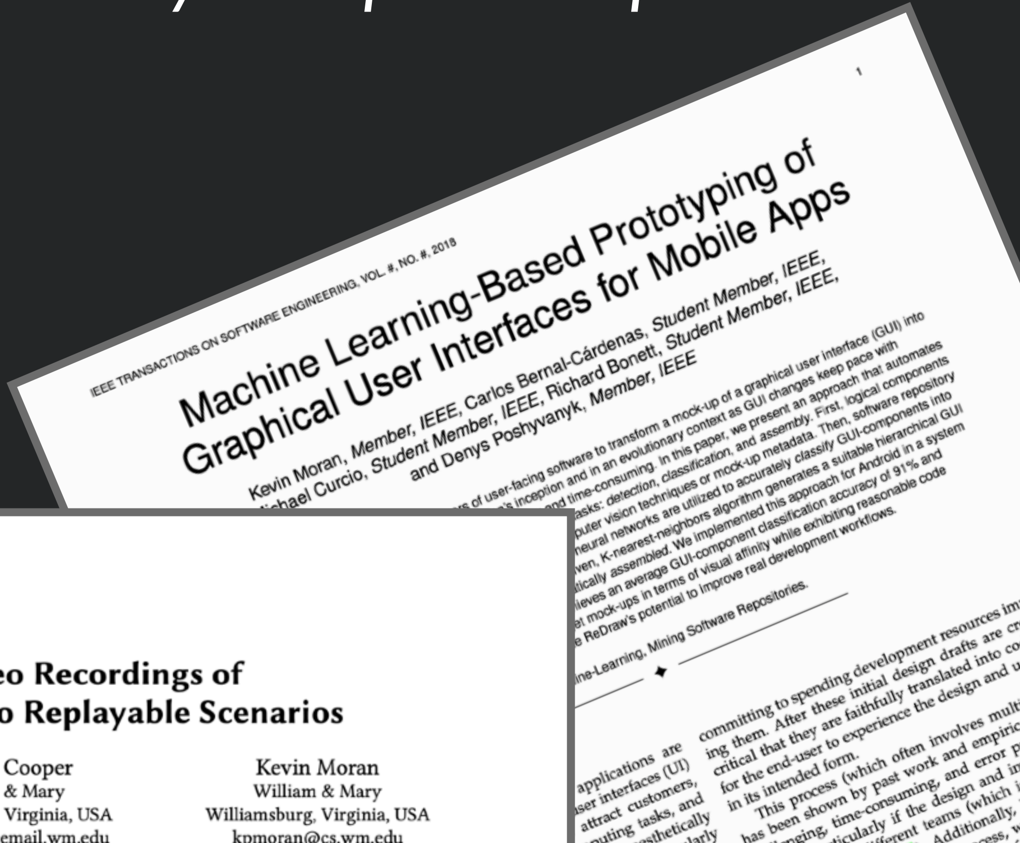
Introductions

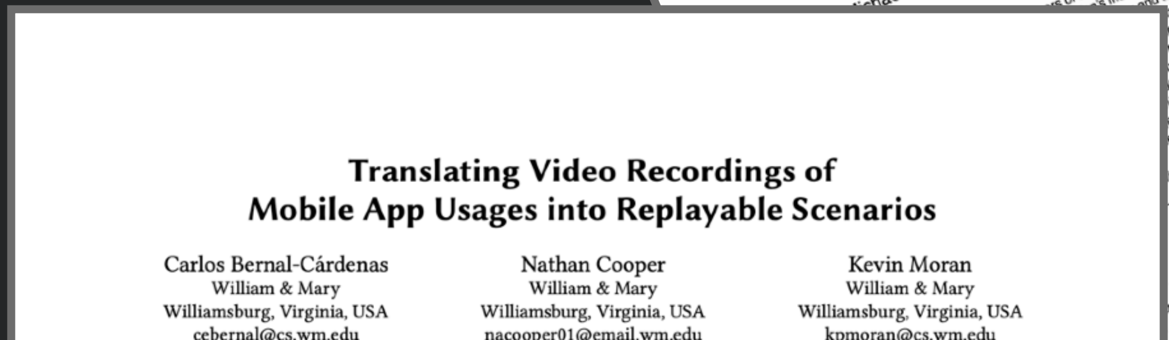
Instructor: Kevin Moran

Education: Ph.D. from William & Mary - 2018

Research Interests: Software Engineering , UI Analysis, Machine Learning

Office Hours: *Monday & Wednesday 1:00pm-2:00pm*





Student Introductions

Today's Agenda

1. Provide an overview of the ***Course Logistics*** - (~20 mins)

2. Discuss how to ***Read a Research Paper*** - (~20 mins)

3. ***10 Minute Break***

4. ***Research Talk -*** Deep Learning & Software Engineering: A Retrospective & Paths Forward - (~40 mins mins)

5. ***In-class Questions & Discussion*** - DL & SE - (~15-20 mins)

Course Logistics

Course Philosophy

- This is a research intensive class!
- Primarily designed for Ph.D. students
- We will mostly be reading, presenting, and discussing cutting edge research papers.
- Students will also be expected to carry out a ***significant*** semester-long research paper with a formal write up.

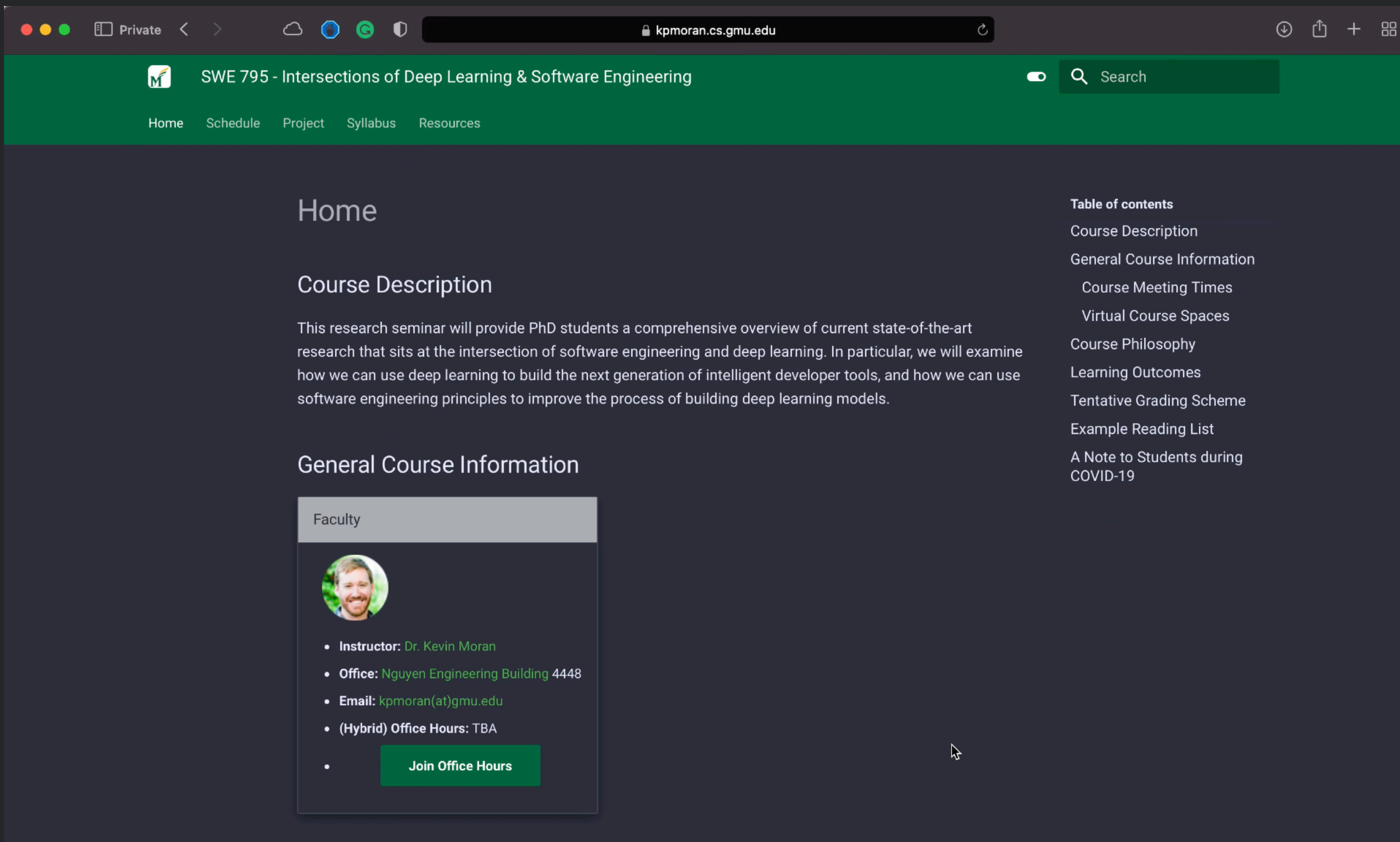
Course Meetings

- Given that this is a research-focused class we will primarily be presenting and discussing research papers.
- Each class we will present/discuss 2-3 research papers.
- You are expected to have read the research papers ***before class*** and contribute to discussion.
- ***However, if you are feeling ill, please do not come to class. I will try to make arrangements for you attend virtually.***

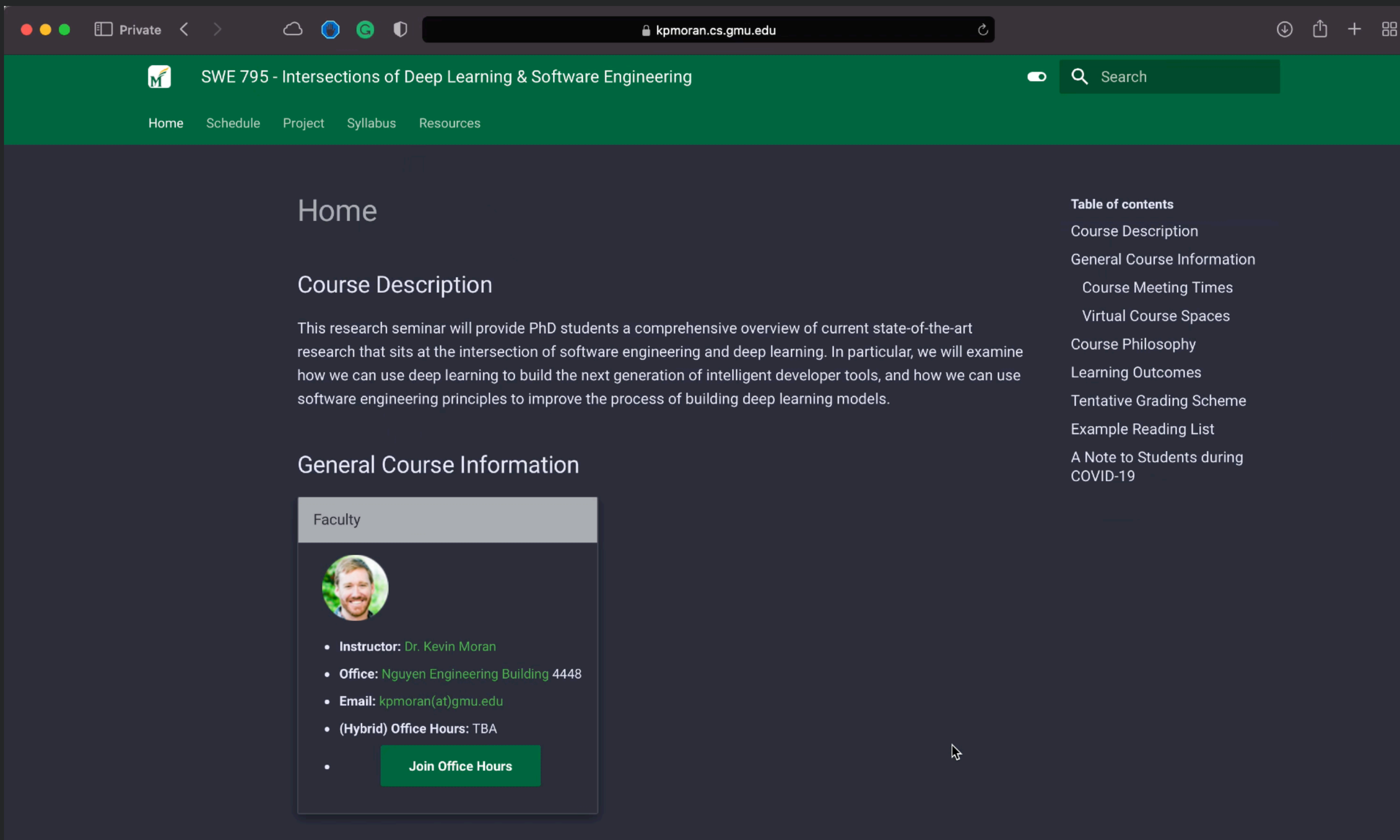
Course Resources

- Course Website: Syllabus, Schedule, Assignments, Lecture slides/recordings
- Ed Discussions: Announcements, Discussions
- Blackboard (MyMason): Grades
- Zoom: Hybrid/Virtual Office Hours

Course Website



Course Website



Course Materials

- There is no course textbook, however readings will be posted to the course website or Ed Discussions.
- Lecture Slides and Videos will be made available on the Course Webpage.

Grading Breakdown

- Research Project- (50%)
- Critical Paper Reviews - (20%)
- Research Paper Presentations - (20%)
- In-Class Discussion - (10%)

Course Project

- Semester-long research project.
- I will provide a ***list of potential projects*** to choose from next week.
- Two types of projects: ***original research*** or ***replication study***.
- You can also propose your ***own project*** and have it ***approved by the instructor***.
- ***Up to two students*** can work on a single project. However, only one grade will be assigned per project.
- We will have a sign up sheet for projects that will be first-come first-served.

Late Policy - Project Checkpoints

- You will have ~2-4 weeks to complete each Project Checkpoint
- Can submit up to:
  - 24 hours late, lose 10%
  - 48 hours late, lose 20%
- Submissions more than 48 hrs late will receive a 0
- ***These are still uncertain times, if you have unforeseen problems, please contact me <u>before</u> the deadline!***

Critical Paper Reviews

- Critiquing research papers is an imperative skill for PhD students.
- It will help you to understand the good, the bad, the exciting, and the ugly of a paper.
- No research paper is perfect!
- Conducting these reviews will help you to learn how to find and extract the most important information from a research paper.
- More on this in the next class!

Research Paper Presentations

- Good research is only useful to society when it is broadly disseminated to a wide audience.
- As such, presenting research through oral presentations is a necessary skill for computer scientists.
- You will be asked to present ***1-2 research papers*** over the course of the semester.
- We will have a sign up sheet for papers that will be ***first-come first-served***.
- More on this in the next class

Honor Code

- Refresh yourself of the department honor code
- This should go without saying in a Ph.D. level course, but all of your work should be your own, and should be original.

Policies

- ***<u>My promises to you:</u>***
  - I will provide detailed instructions and rubrics for paper presentations, critiques, and project checkpoints
  - Project Checkpoints will be graded within 1 week of submission
  - I will make myself available to discuss course projects as much as possible

Important Initial Dates/Deadlines

- The course reading list will be posted by *Monday, January 31st.*
- Paper selection (for in-class presentations) will be due by *Thurs, February 4th*.
- Project Topics will be presented in the next class.
- Project selection is due by *Monday, February 7th*.

SWE 795:

Intersections of Deep Learning & Software Engineering

Spring 2022

George Mason University

Dr. Kevin Moran

Week 1:

How to Read a Research Paper

Attribution

- Adapted from William G. Griswold's advice on "How to Read an Engineering Research Paper"
- http://www-cse.ucsd.edu/~wgg/CSE210/howtoread.html

Before Reading a Research Paper

- Reading research papers ***effectively*** is challenging
  - Why?
  - Condensed style, focused audience, paper organization
- To effectively read papers you should know:
  - What you should get out of the paper?
  - Where that information is located?

How a Research Paper is Organized

- Technical papers are repetitive in nature!
  - Introduction = motivation + solution outline
  - Related Work
  - Body of the Paper
    - Details on the solution
    - Detailed evaluation
  - Discussion of the results
  - Conclusions (recap of contributions and results)
  - Because of these repetitions, you can read the paper ***'out of order'***

Questions You Want to Answer - I

- *What are the motivations for this work?*
  - A published paper solves the problem and no one else has published in the literature
  - Why there is no trivial solution to this problem?
  - What are the previous solutions and why are they inadequate?
- *Specific research questions?*
  - Motivation and statement should lead to this
  - This does not always happen – your job is a bit more difficult in that case

Questions You Want to Answer - 2

- *What is the proposed solution?*
    - Hypothesis (until it has been evaluated) or idea
    - Why is this solution better than previous solutions?
    - How the solution is achieved (design, implementation)?
    - Is it achievable at all? To what extent?

Questions You Want to Answer - 3

- *<u>What is the work's evaluation of the proposed solution?</u>*
  - Just having an idea is not sufficient anymore (although it used to be many years ago ...)
  - This is a concrete engagement of the research question (e.g., numbers)
  - Under which circumstances does it work (e.g., numbers) ?
  - What benefits and problems are identified?

Questions You Want to Answer - 4

- ***What is your analysis of the identified problem, idea and evaluation?*** (remember paper reports and subjective evaluation ...)
  - Is this a good idea?
  - What flaws do you perceive in this work?
  - What are the most interesting points?
  - What are the most controversial ideas or points?
  - Is it really going to work?
  - When might it become a reality?

Questions You Want to Answer - 5

- *What are the contributions:*
  - A new understanding of a research problem?
  - A new methodology for solving a problem?
  - A new algorithm?
  - A new breed of software tools or systems?
  - A new experimental method?
  - A new formalism or notation?
  - A new evidence to substantiate or disprove a previously published claim?
  - A new research area?

Questions You Want to Answer - 6

- *What are the future directions for this research*
  - What do authors identify as a future work?
  - What ideas did you come up with while reading the paper?
  - You may get answers to these questions from the analysis of shortcomings or other critiques in the current work

Questions You Want to Answer - 7

- *What is your take-away message from this paper?*
    - Sum up the main implication of the paper from your perspective (e.g., from your class project's perspective)!
    - This is also useful for quick review and writing your final project paper!
    - It also focuses you to identify the essence of the work

Mechanics

- As you read/skim the paper, actively attempt to answer questions 1-7
- Get motivation from the intro
- Intro & conclusion – the solution and evaluation at a high level
- Body of the paper – all the meat
- Pay attention to the context – other papers that are presented in the class WILL be relevant (past or future work for some papers ...)

Template for Answering Questions

- Use this template: http://www.cse.ucsd.edu/~wgg/CSE210/paperform.pdf

Taking Notes on the Paper

- *Take Notes on the Paper!*
- Highlight important comments.
- Mark paragraphs: motivation, problem, idea/solution, evaluation, contributions
- Front of the paper: take away message
- Front of the paper: your key questions!
- Other questions are on the margins.
- Try to answer questions yourself. Use Wikipedia and Google (carefully!)

Deep Learning & Software Engineering – A Retrospective and Paths Forward

Kevin Moran, Ph.D.

George Mason University

SWE 795 – Intersections of Deep Learning & Software Engineering

Thursday, January 27th, 2022

GitHub Copilot

Sign up >

Technical Preview

Your AI pair programmer

With GitHub Copilot, get suggestions for whole lines or entire functions right inside your editor.

Sign up >

sentiment.ts | write_sql.go | parse_expenses.py | addresses.rb

```
#!/usr/bin/env ts-node

import { fetch } from "fetch-h2";
```

Talk Outline

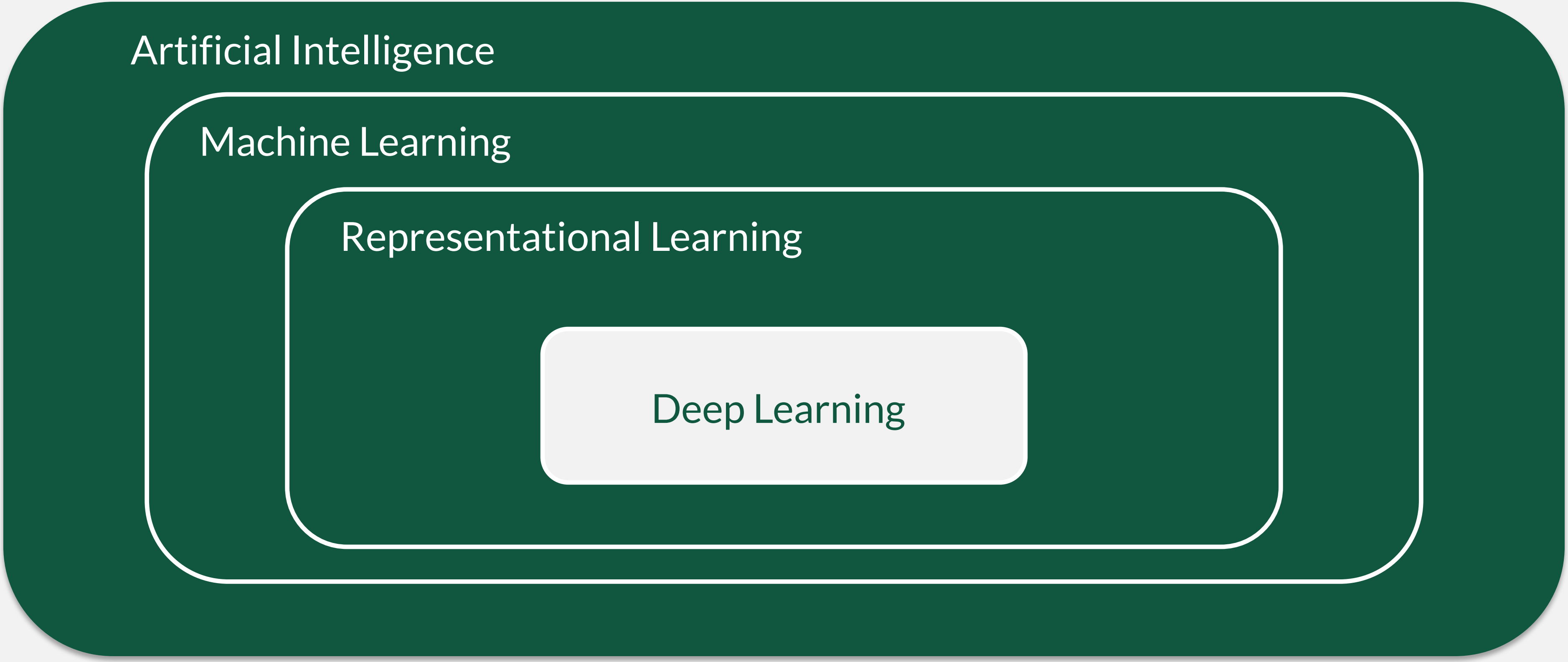
- **Topic 1 - Background:** The Evolution of Machine Learning (ML) to Deep Learning (DL)
- **Topic 2- DL4SE:** The Current State of Research
- **Topic 3 – Looking Forward:** Future Directions and Paths Forward

Topic 1 – Background: The Evolution of Machine Learning to Deep Learning

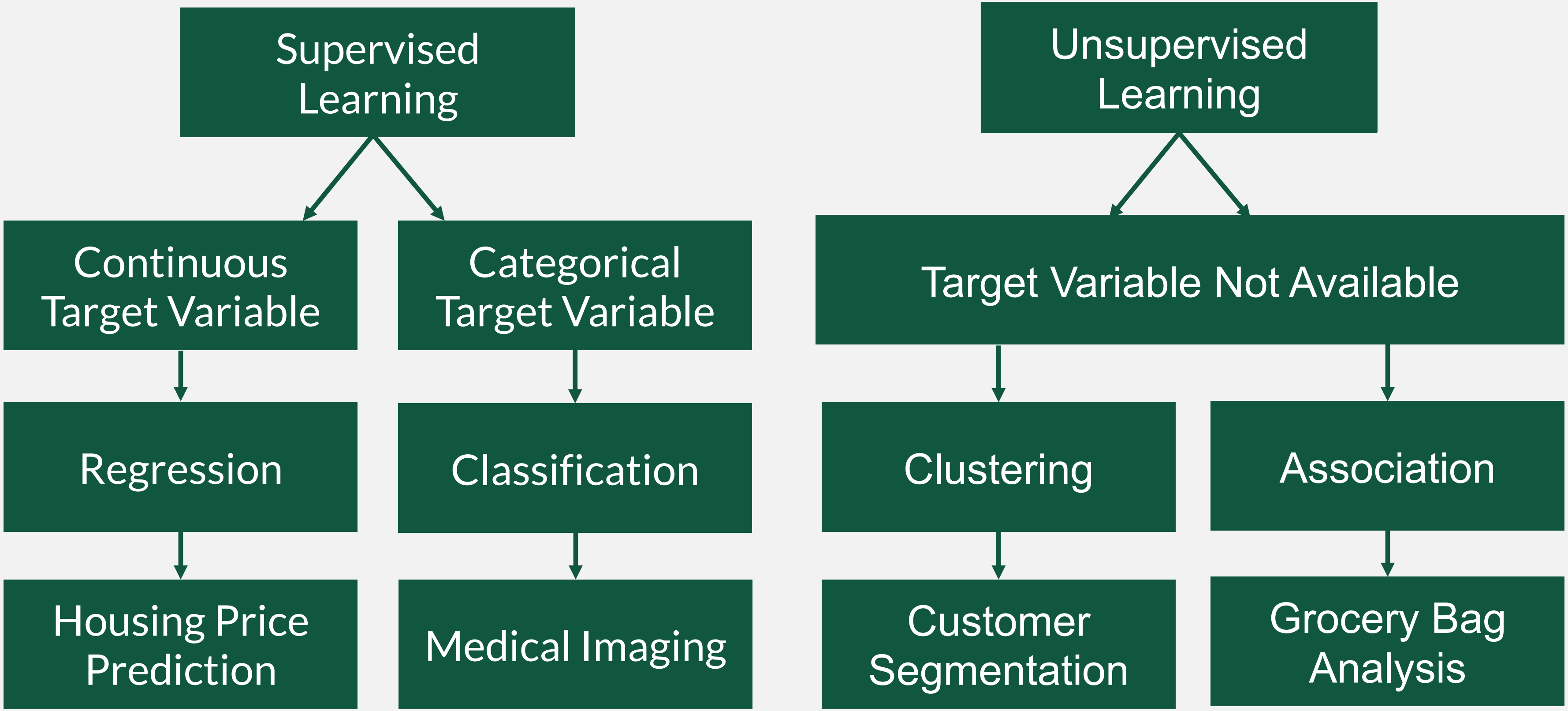
What is Machine Learning?

A branch of ***Artificial Intelligence*** that allows computers to ***infer patterns*** from data, which can be used for the ***prediction*** of new data points

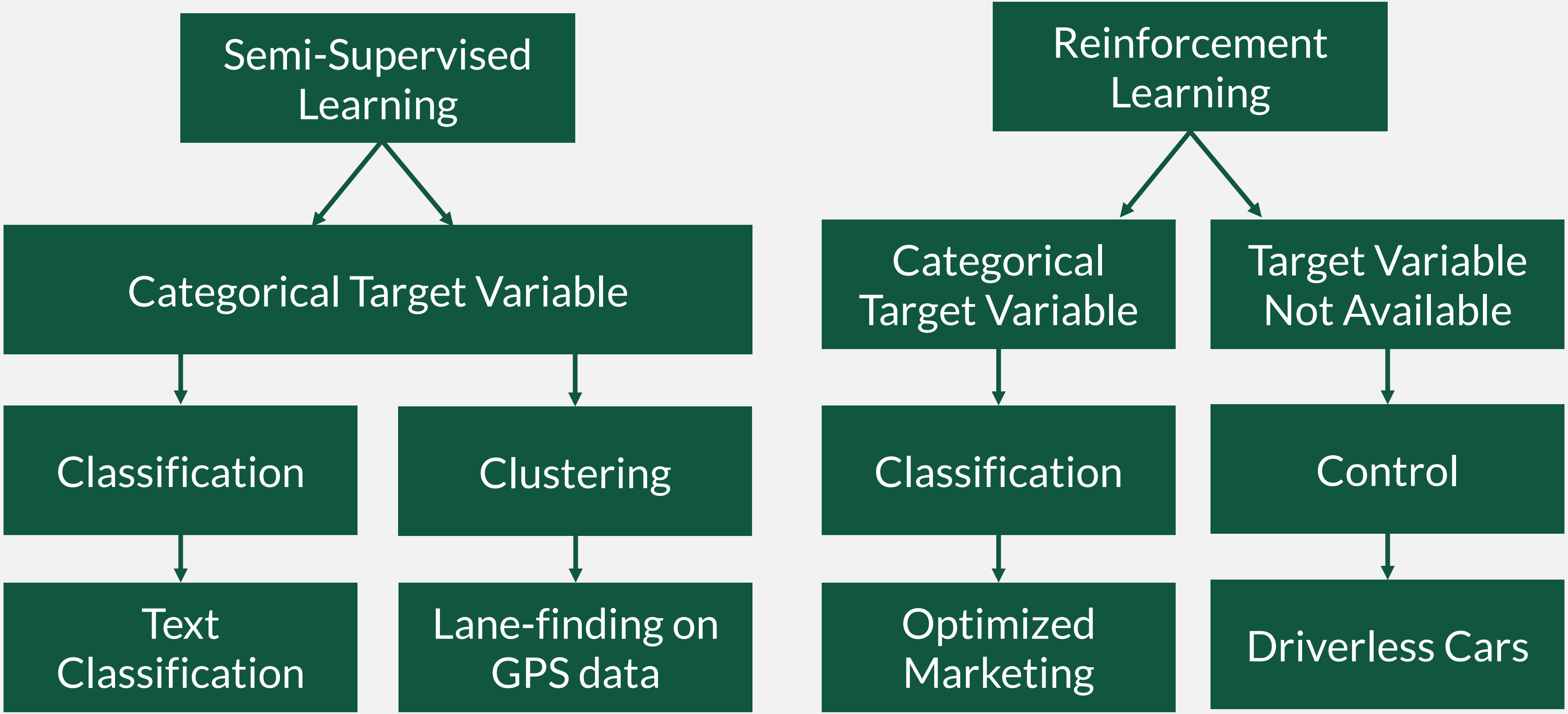
The Hierarchy of Artificial Intelligence



Machine Learning Taxonomy



Machine Learning Taxonomy



ML Representations

Supervised Learning	Unsupervised Learning	Semi-Supervised Learning	Reinforcement Learning
K Nearest Neighbor Naïve Bayes Decision Trees Linear Regression Support Vector Machine	K-means clustering Association rule learning	Self-Training of Existing Classifiers Hidden Markov Models Multiple Gaussian Distributions Semi-supervised support vector machines	Q-Learning Temporal Difference

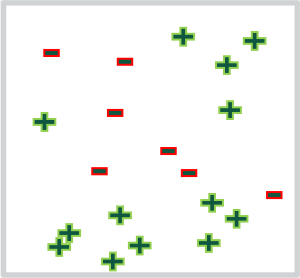
Canonical Representation

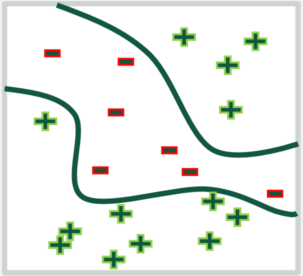
Machine Learning vs. Traditional Programming

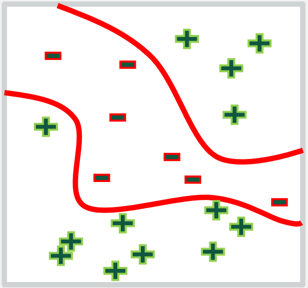
When do We Need Machine Learning?

Three Conditions:

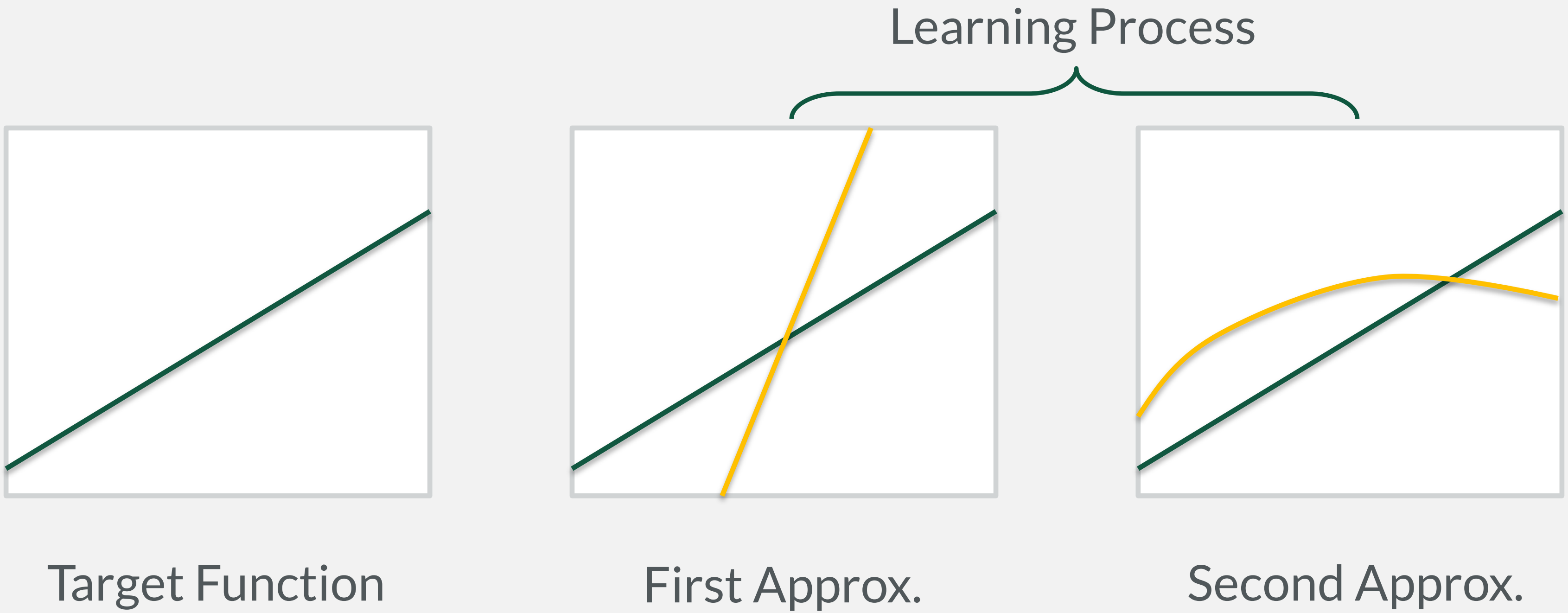
1. We have an Existing Dataset
2. A pattern exists in the data
3. The pattern is not <u>*easily*</u> defined by an equation


The Data


The Pattern


No Possible Equation

The Computational Learning Process



Supervised ML Applied to Image Classification

Important Note!

Our future examples focus on *Supervised Learning* for *Images*

However, the same principles apply to other types of data (*natural language* and *code)* and learning methods (*Unsupervised* and *Reinforcement*).

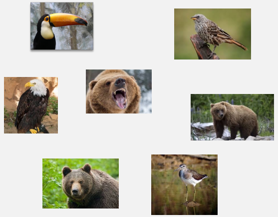
The Five Elements of the Learning Process

Input

Output

{Bird, Bear}

Data



Target Function

F(x)

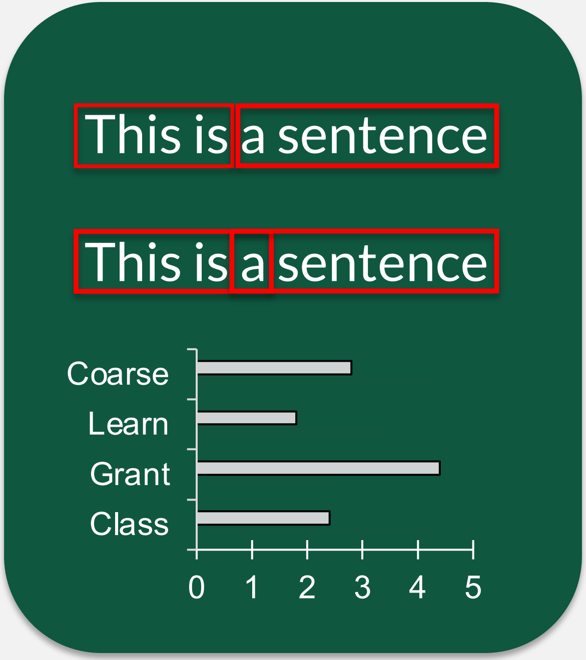
Hypothesis

G(x) ≈ F(x)

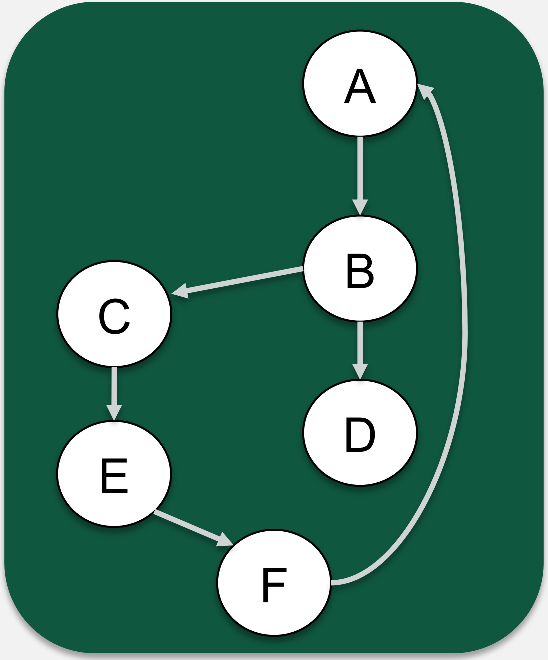
Feature Engineering for “Canonical” Machine Learning

Images

Text



Source Code



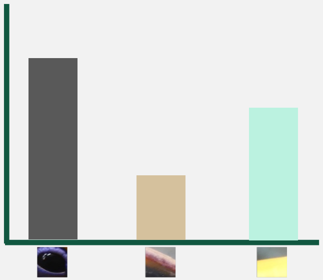
"Canonical" ML Image Classification

On the Large-Scale *ImageNet* Dataset,
which contains millions of images
from over 1000 categories

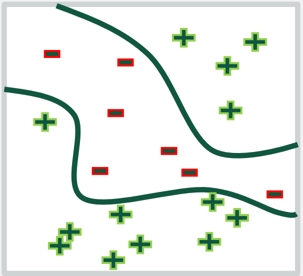
Canonical ML techniques have only been
able to achieve ***~ 60% accuracy***

Shortcomings of Traditional ML Techniques

Manually Derived Features



Complex Kernels



Shallow Representation

1	0	1	1	0	1	1	0	1

The Advent of Deep Learning

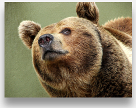
Input

1	0	1	1	0	1	1	0	1

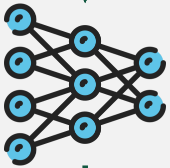
Feature Mapping

Output → Bird

"Canonical" Machine Learning



Input



→ Rich Hierarchical Feature "Embedding"

Feature Mapping

Output → Bear

Deep Learning

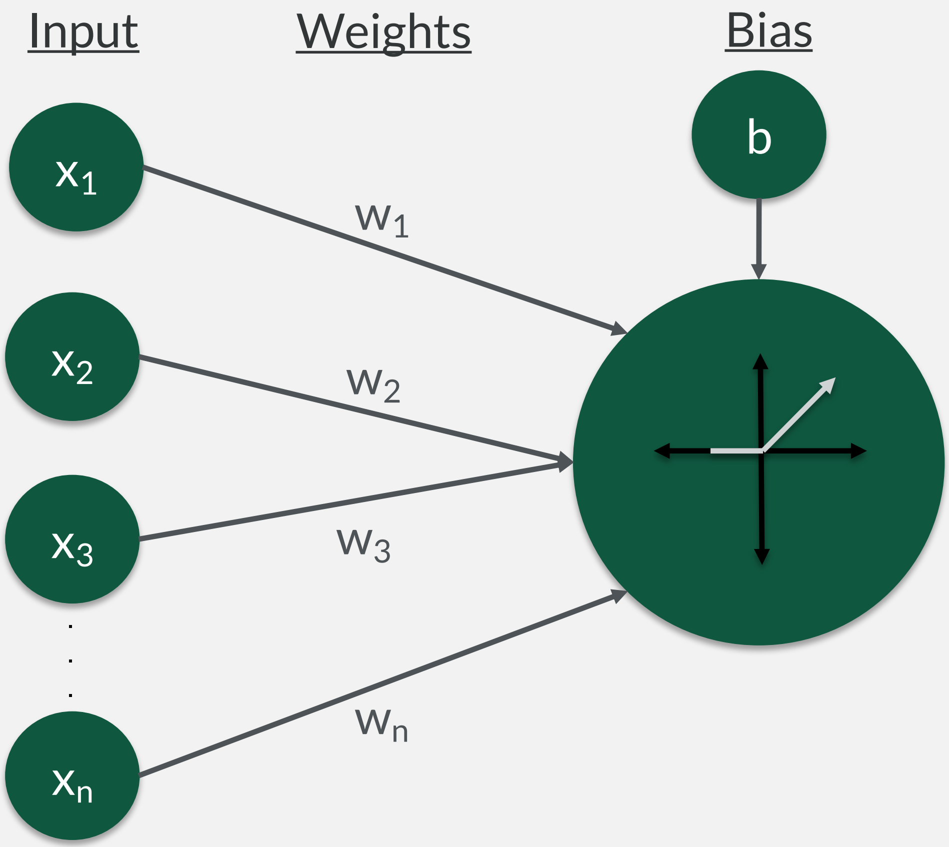
ML Representations

Supervised Learning	Unsupervised Learning	Semi-Supervised Learning	Reinforcement Learning
K Nearest Neighbor	K-means clustering	Self-Training of Existing Classifiers	Q-Learning
Naïve Bayes	Association rule learning	Hidden Markov Models	Temporal Difference
Decision Trees	Autoencoders	Multiple Gaussian Distributions	Deep Adversarial Networks
Linear Regression	Deep Belief Networks	Semi-supervised support vector machines	Deep Q-Learning
Support Vector Machine	Generative Adversarial Networks (GANs)	Neural networks	
Neural Networks (Convolutional, Recurrent, Feed-forward, etc.		Autoencoders	

Canonical Representation

Deep Representation

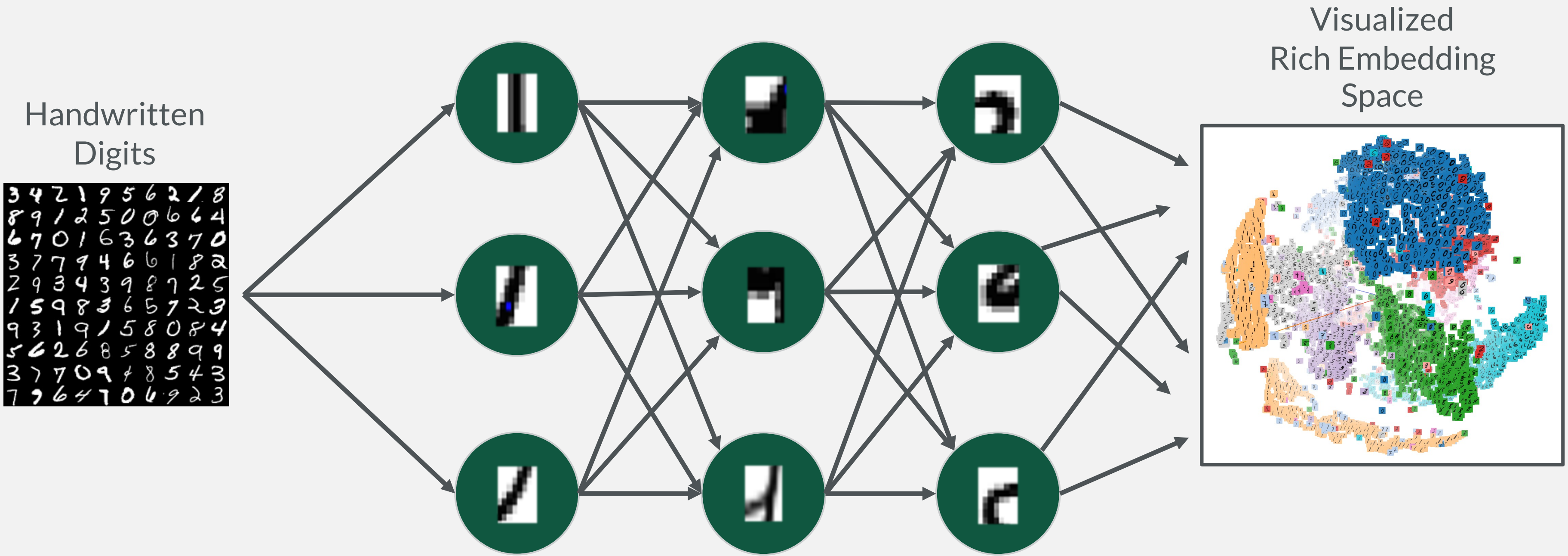
Neurons: The Building Blocks of Rich Features



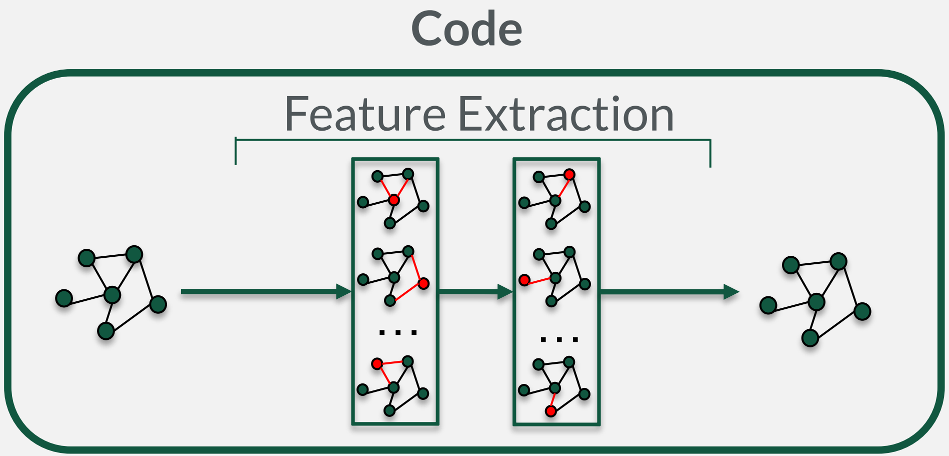
Additional Activation Functions

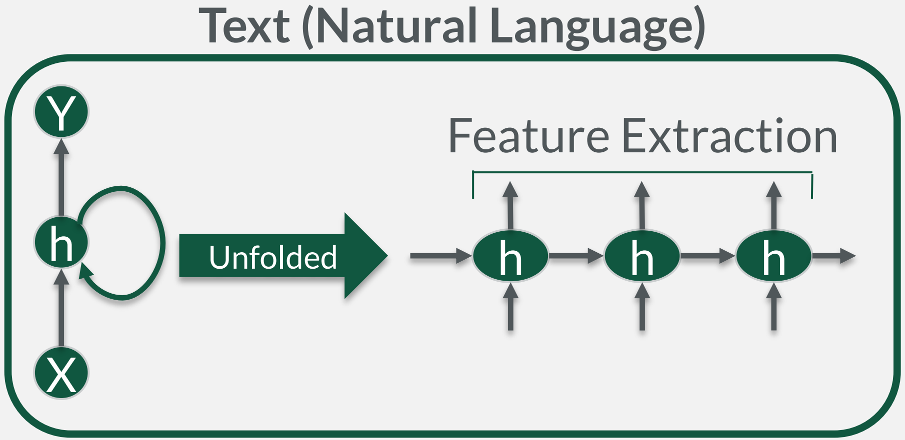
- Identity
- Binary Step
- Sigmoid
- Tanh
- Leaky ReLU
- Softmax

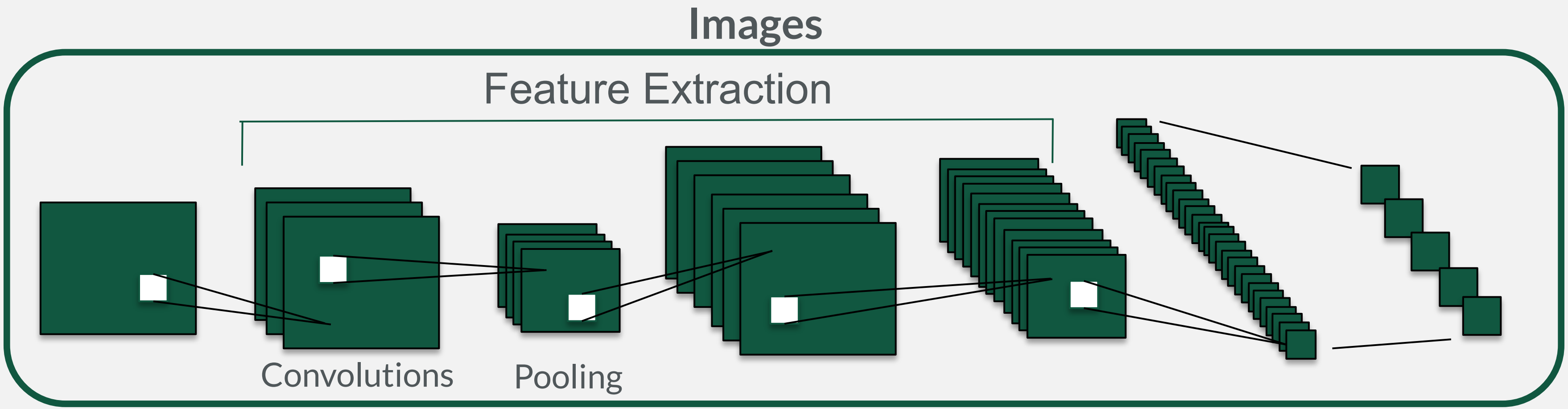
Neural “Networks” for Rich Embeddings



Automated Feature Discovery



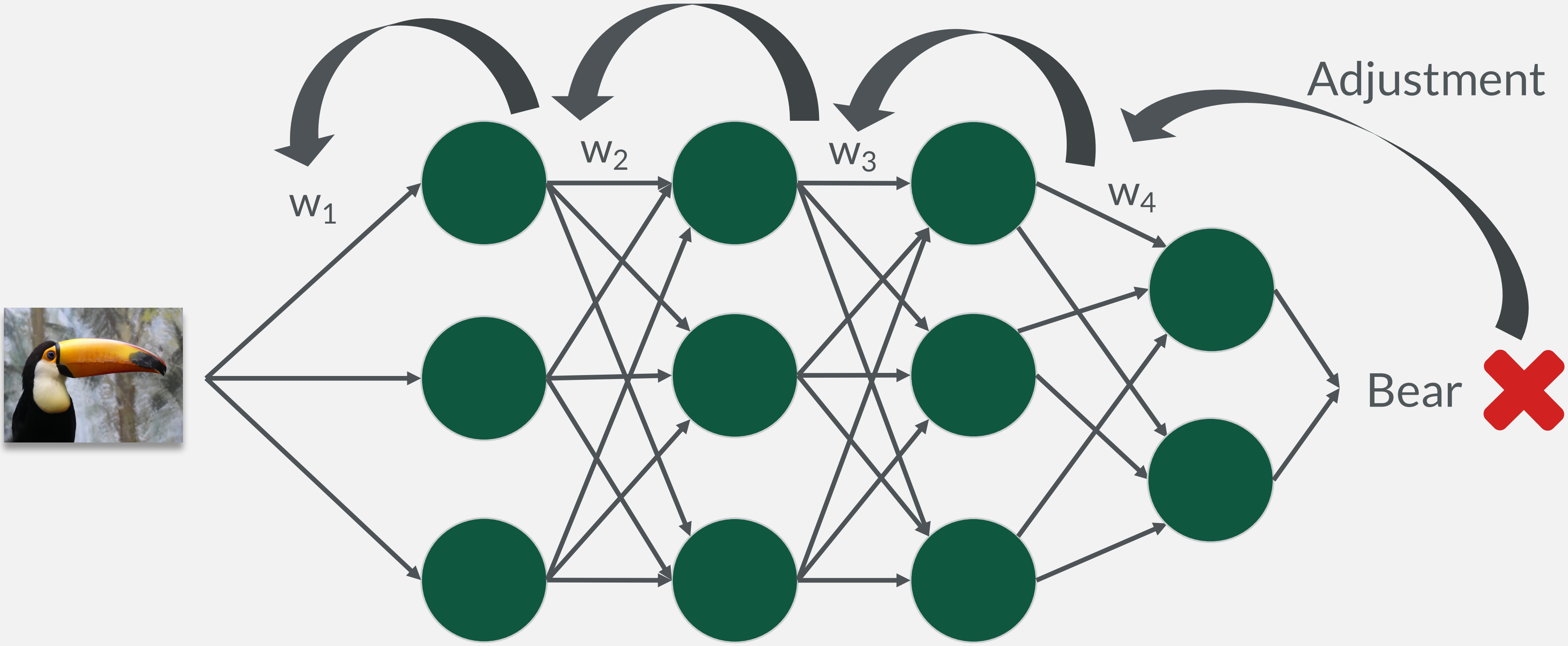




How Can a Model Learn from Deep Embeddings?

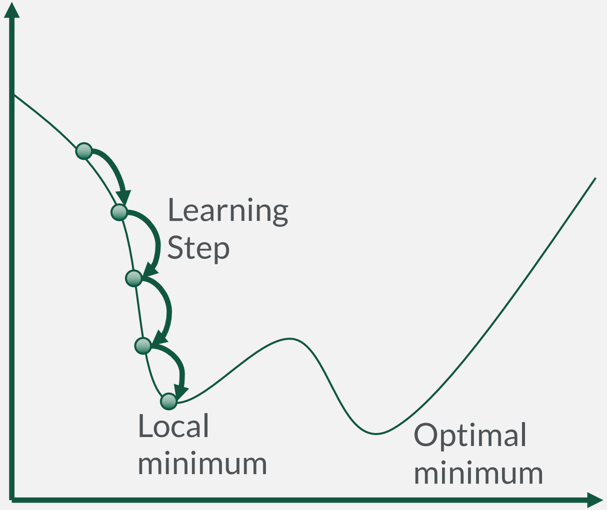
Adjust the Neuron "***weights***" according to ***errors*** made on a given task.

How Can a Model Learn from Deep Embeddings?

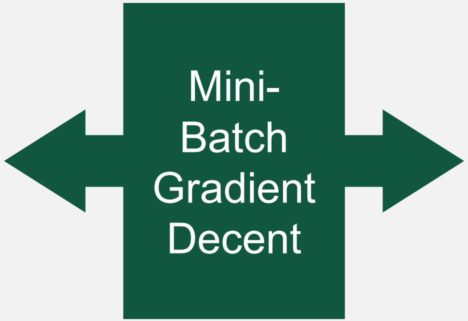


How Should the Weights be Updated?

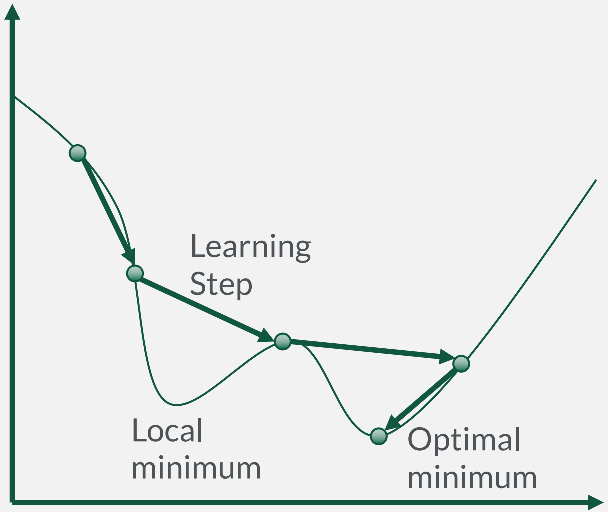
Batch Gradient Decent



Error Gradient



Stochastic Gradient Decent



Error Gradient

CNN-Accuracy

ConvNets have ***surpassed human levels of accuracy*** on the ImageNet classification dataset

Deep Learning Advantages and Drawbacks

Advantages

- Does not require manual feature engineering
- Capable of Learning Rich, Hierarchal Data Representations
- Can be trained for a given task end-to-end

Disadvantages

- Require massive datasets to function effectively
- Computationally expensive to train
- Models can difficult to interpret (Black Box)

Topic 2 – DL4SE: The Current State of Research

Mining Software Repositories

Bitbucket

GitHub

Google Play

SOURCEFORGE

GitLab

Automation in Software Engineering Research

Source Code Files

Software Documentation

Screenshots

Screen Recordings

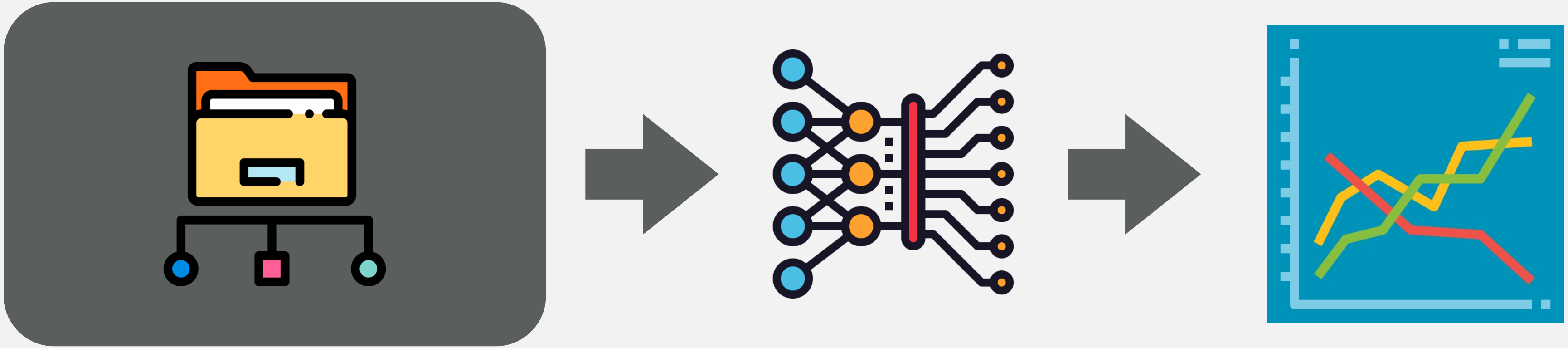
Bug Reports

Design Documents

Automation in Software Engineering Research

Software Repository Data

Salient Patterns



Deep Learning

What is the current state-of-the-art of DL4SE?

Systematic Literature Review

###

A Systematic Literature Review on the Use of Deep Learning in Software Engineering Research

CODY WATSON, Washington & Lee University
NATHAN COOPER, William & Mary
DAVID NADER PALACIO, William & Mary
KEVIN MORAN, George Mason University
DENYS POSHYVANYK, William & Mary

An increasingly popular set of techniques adopted by software engineering (SE) researchers to automate development tasks are those rooted in the concept of Deep Learning (DL). The popularity of such techniques largely stems from their automated feature engineering capabilities, which aid in modeling software artifacts. However, due to the rapid pace at which DL techniques have been adopted it is difficult to distill the current successes, failures, and opportunities of the current research landscape. In an effort to bring clarity to this cross-cutting area of work, from its modern inception to the present, this paper presents a systematic literature review of research at the intersection of SE & DL. The review canvases work appearing in the most prominent SE and DL conferences and journals and spans 84 papers across 22 unique SE tasks. We center our analysis around the *components of learning*, a set of principles that govern the application of machine learning techniques (ML) to a given problem domain, discussing several aspects of the surveyed work at a granular level. The end result of our analysis is a *research roadmap* that both delineates the foundations of DL techniques applied to SE research, and likely areas of fertile exploration for the future.

CCS Concepts: • **Software and its engineering** → *Software creation and management*; *Software development techniques*;

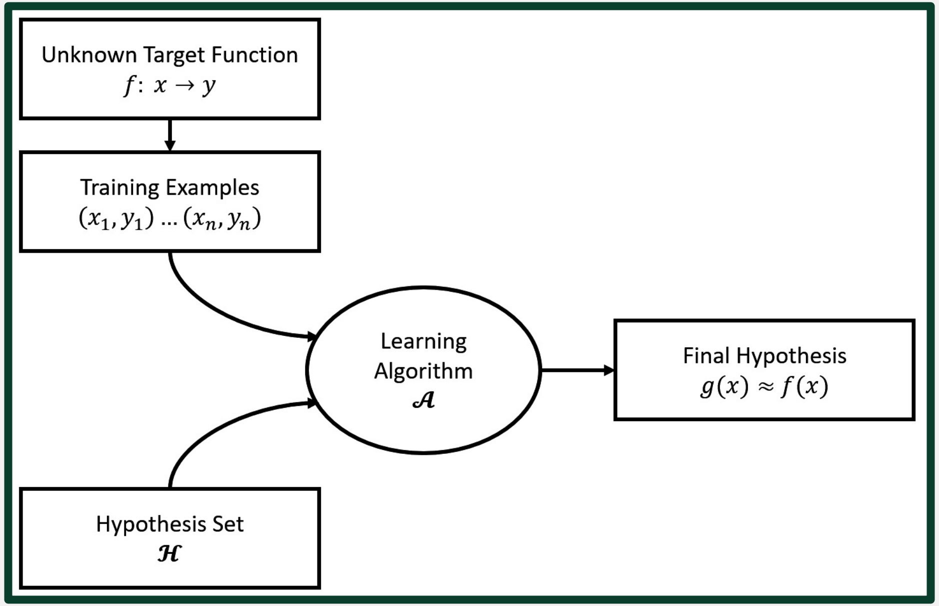
Additional Key Words and Phrases: deep learning, neural networks, literature review, software engineering, machine learning

Systematic Literature Review

Research Questions (RQs) centered upon the “components of learning”

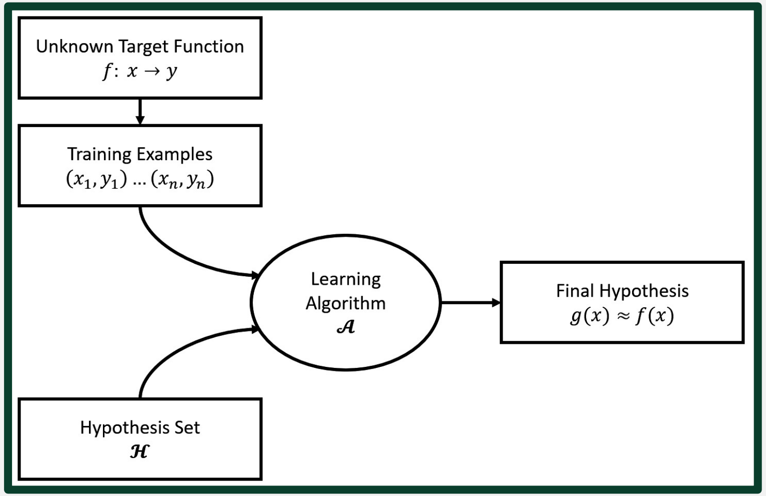
Systematic Literature Review

Research Questions (RQs) centered upon the “components of learning”



Systematic Literature Review

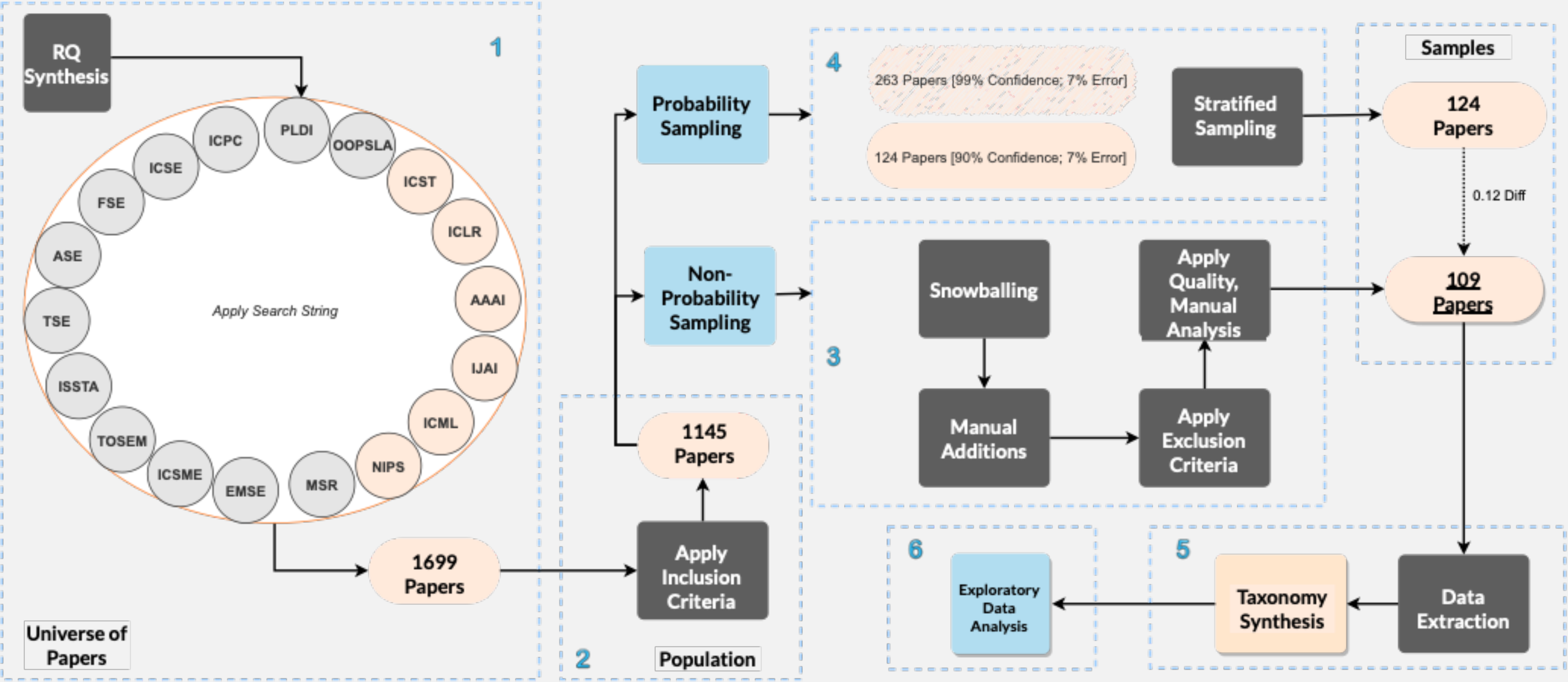
- **RQ_1**: Target Function (*SE Task*)
- **RQ_2**: Data (*Training/Testing Data*)
- **RQ_3**: Learning Model (*Algorithm + Hypothesis Set*)
- **RQ_4**: Final Hypothesis (*Results*)
- **RQ_5**: Reproducibility and Replicability



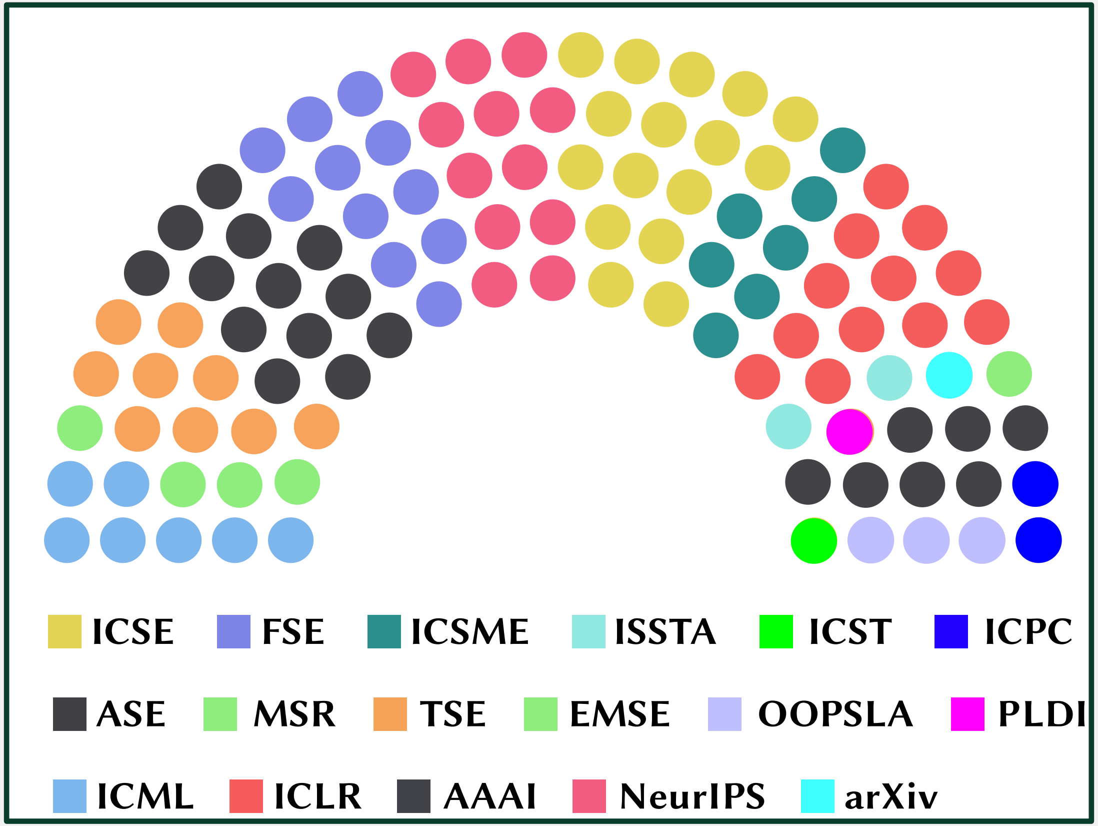
Systematic Literature Review

- **Time Period:** 2009-(mid)2019
- **Venues:** ICLR, NeurIPS, FSE, ICML, MSR, ISSTA, ICST, ICSE, ASE, ICSME, TSE, TOSEM, EMSE, OOPSLA, ICPC, PLDI, AAAI, IJCAI.
- **Methodology:** Following *Kitchenham*, *et.al.*

SLR Search Process

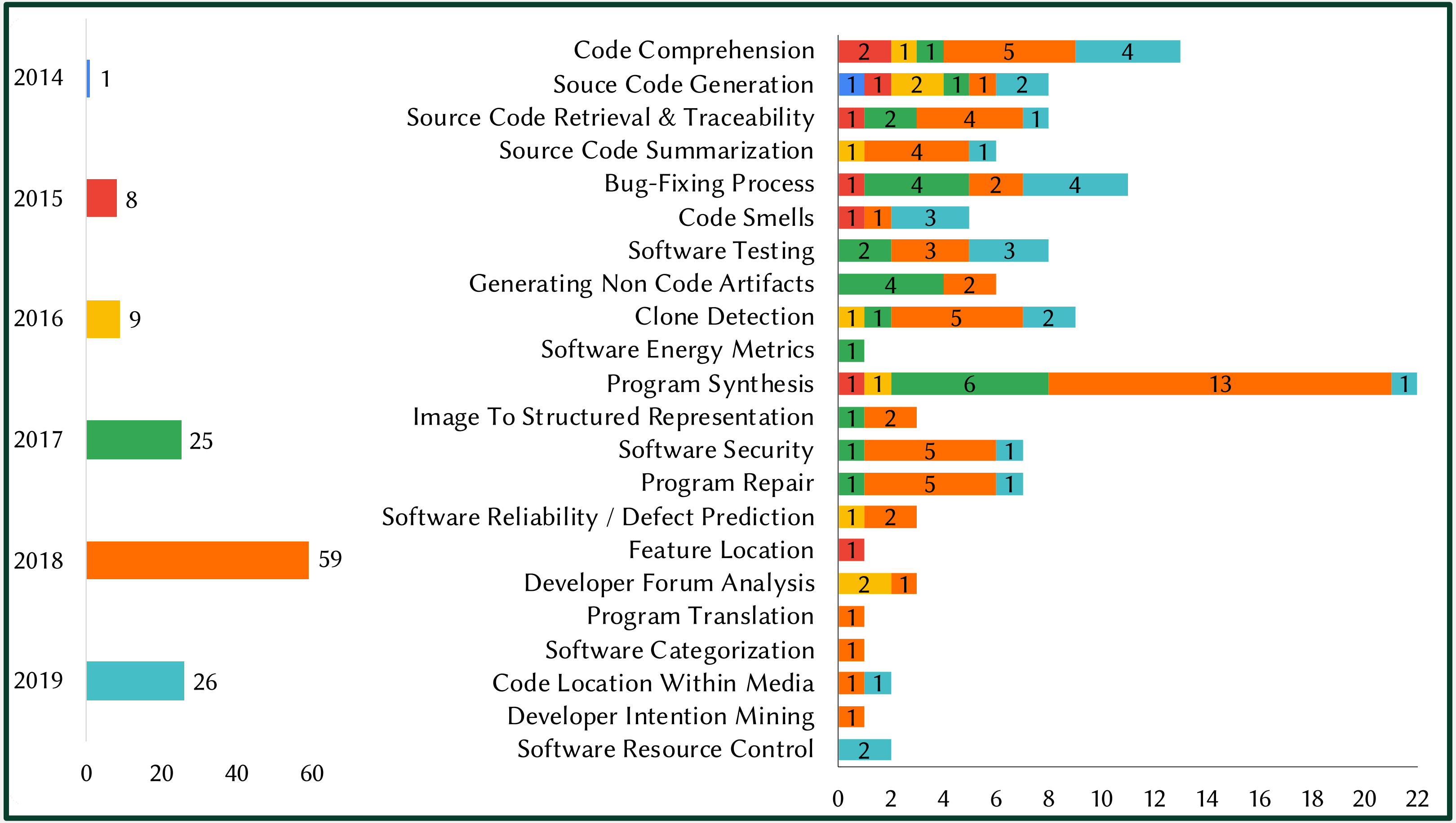


Publication Distribution By Venue



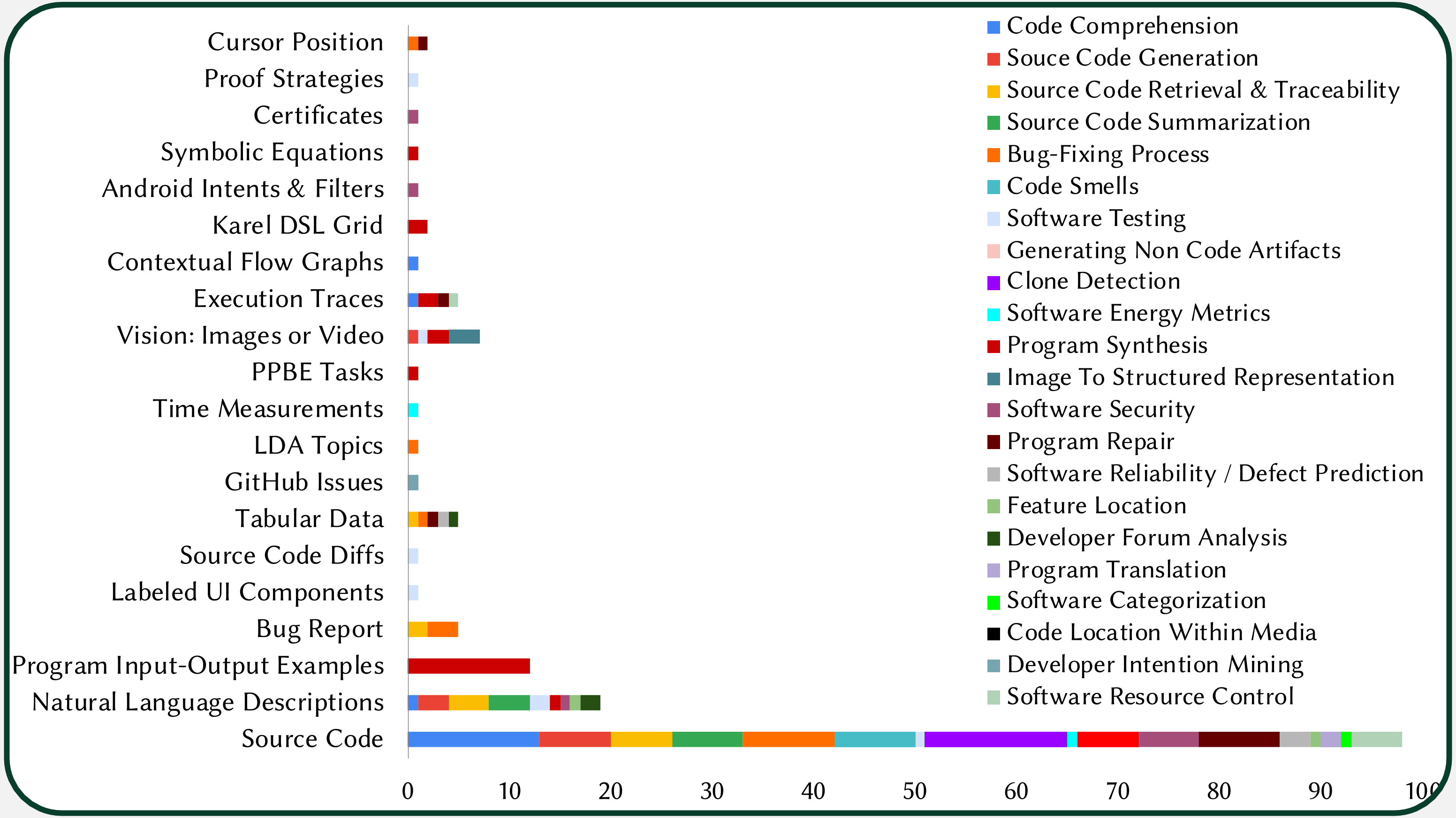
RQ_1: Target Function (SE Task)

DL4SE Publications Over Time and SE Tasks

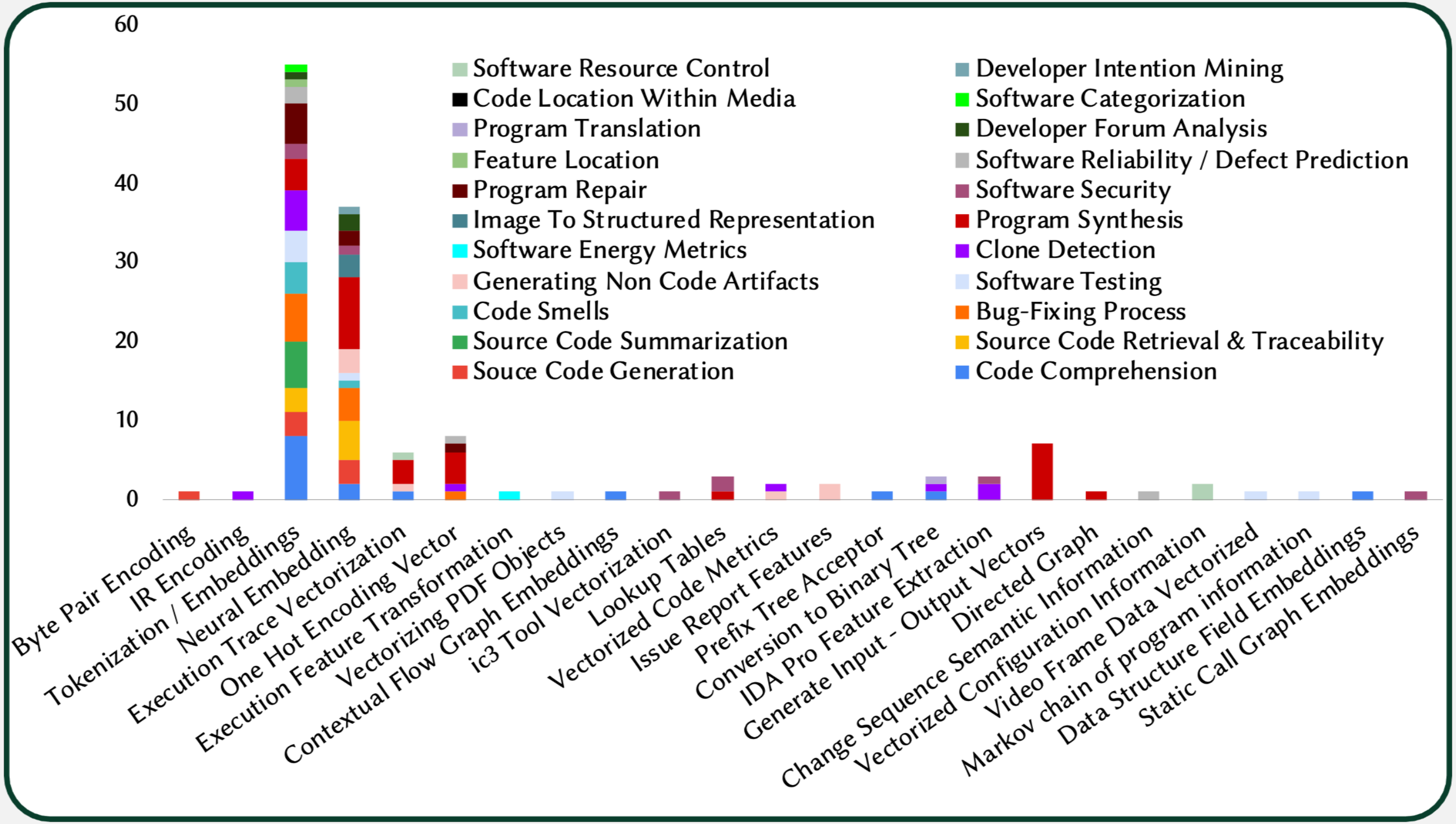


RQ_2: Data (Training/Testing Data)

Data Used in DL4SE Approaches by SE Task

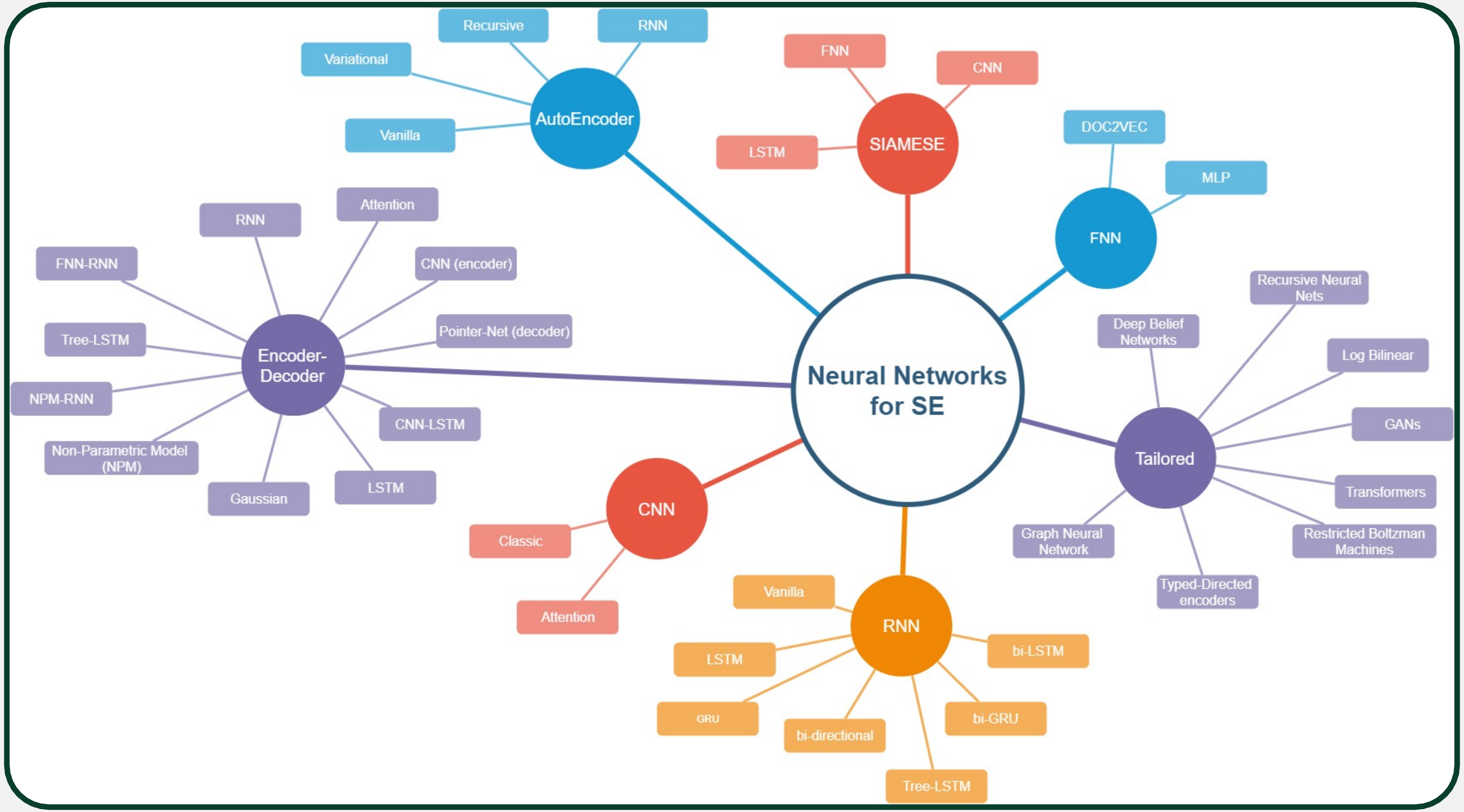


Data Processing Techniques by SE Task

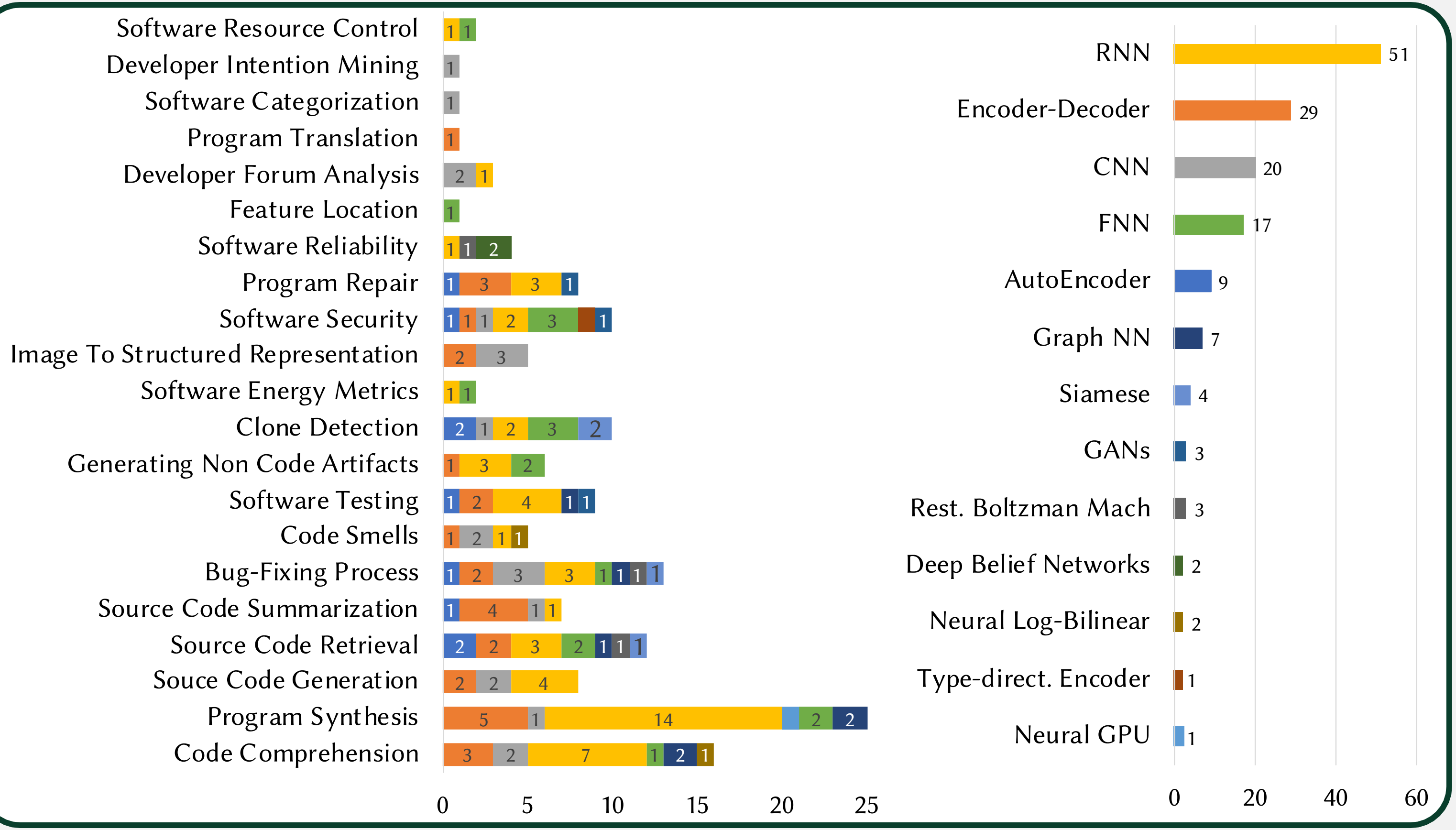


RQ_2: Learning Model (Algorithm + Hypothesis Set)

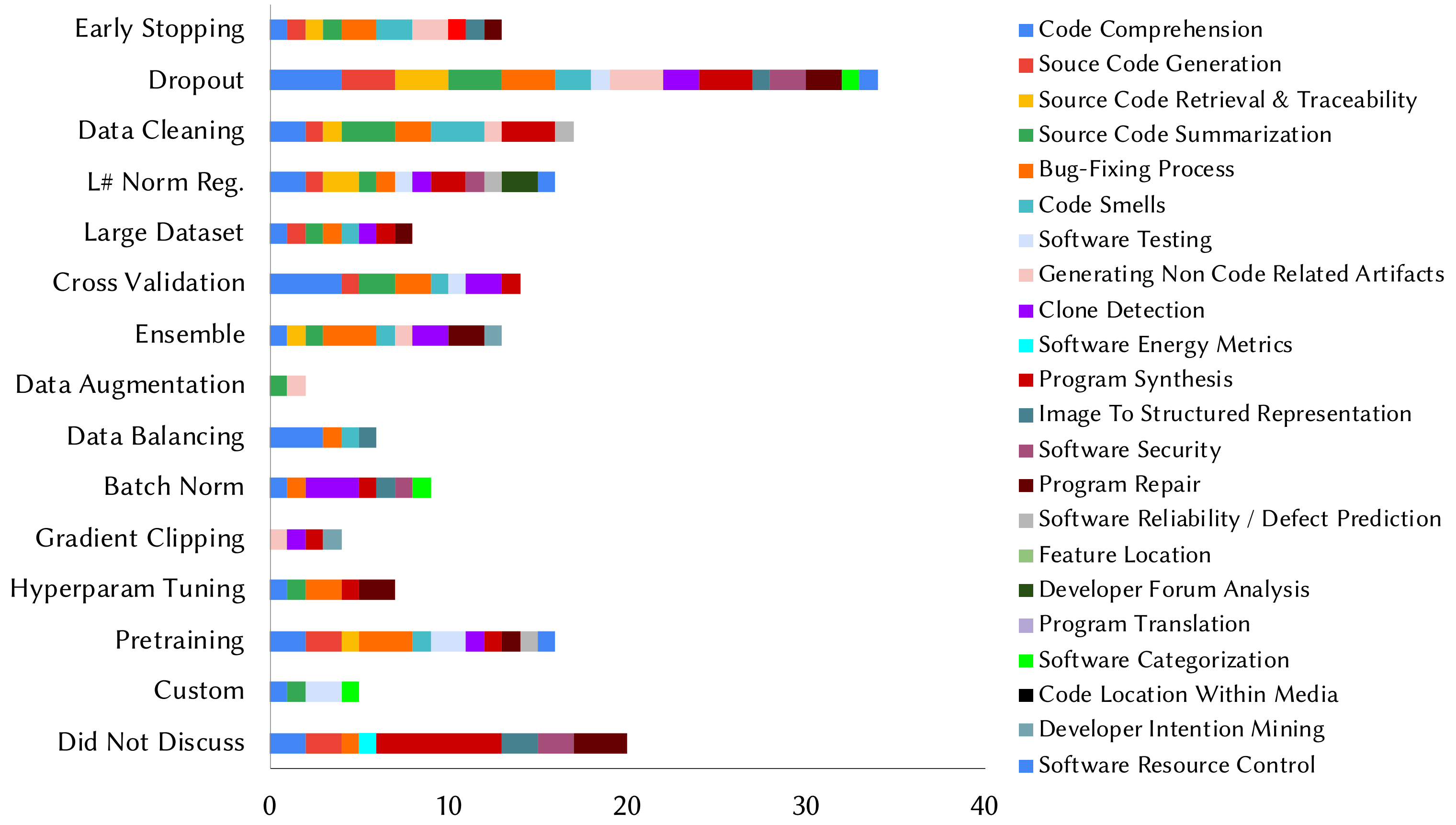
DL4SE Neural Network Architectures



DL4SE Neural Network Architectures

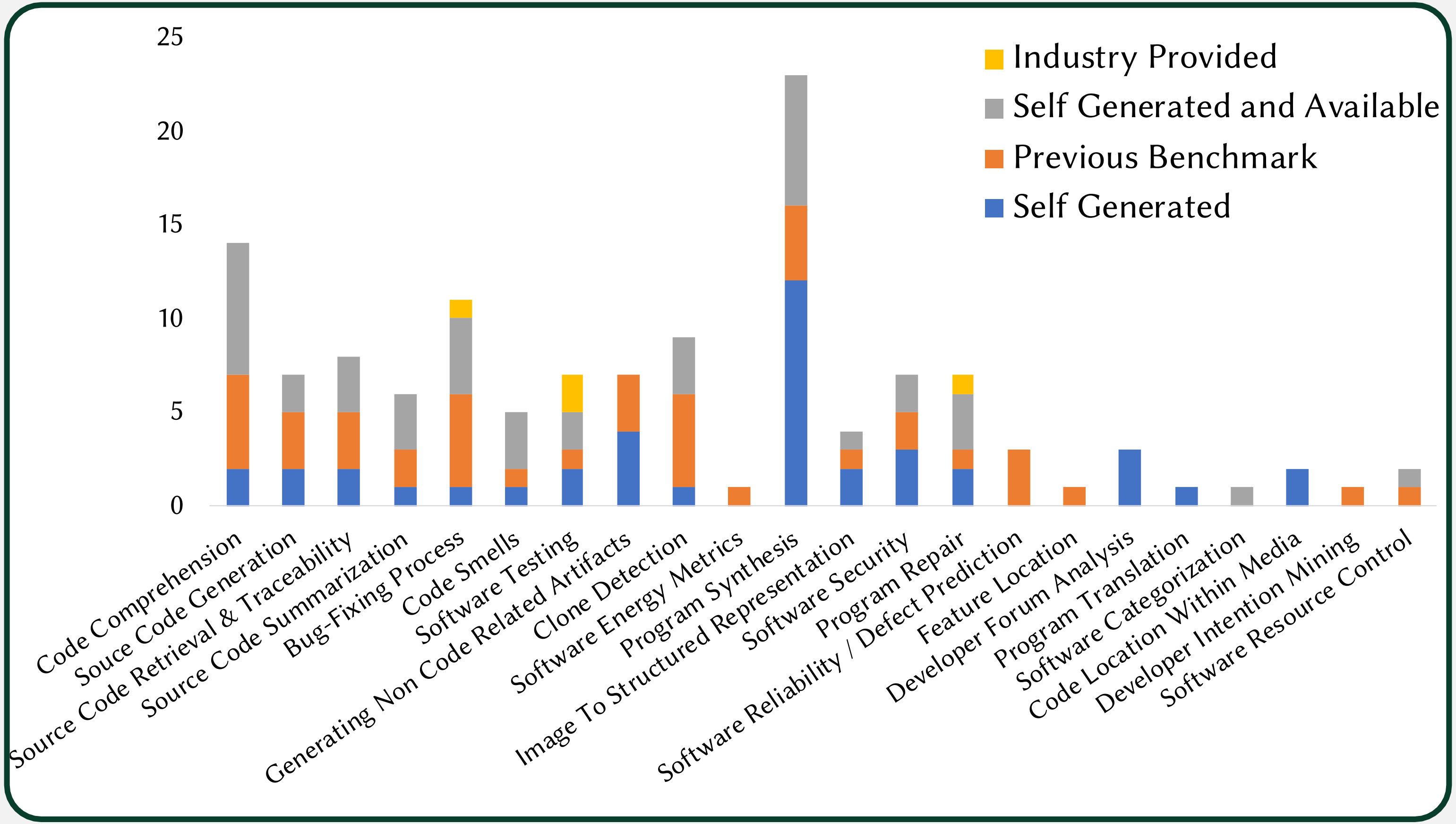


DL4SE Techniques to Combat Overfitting

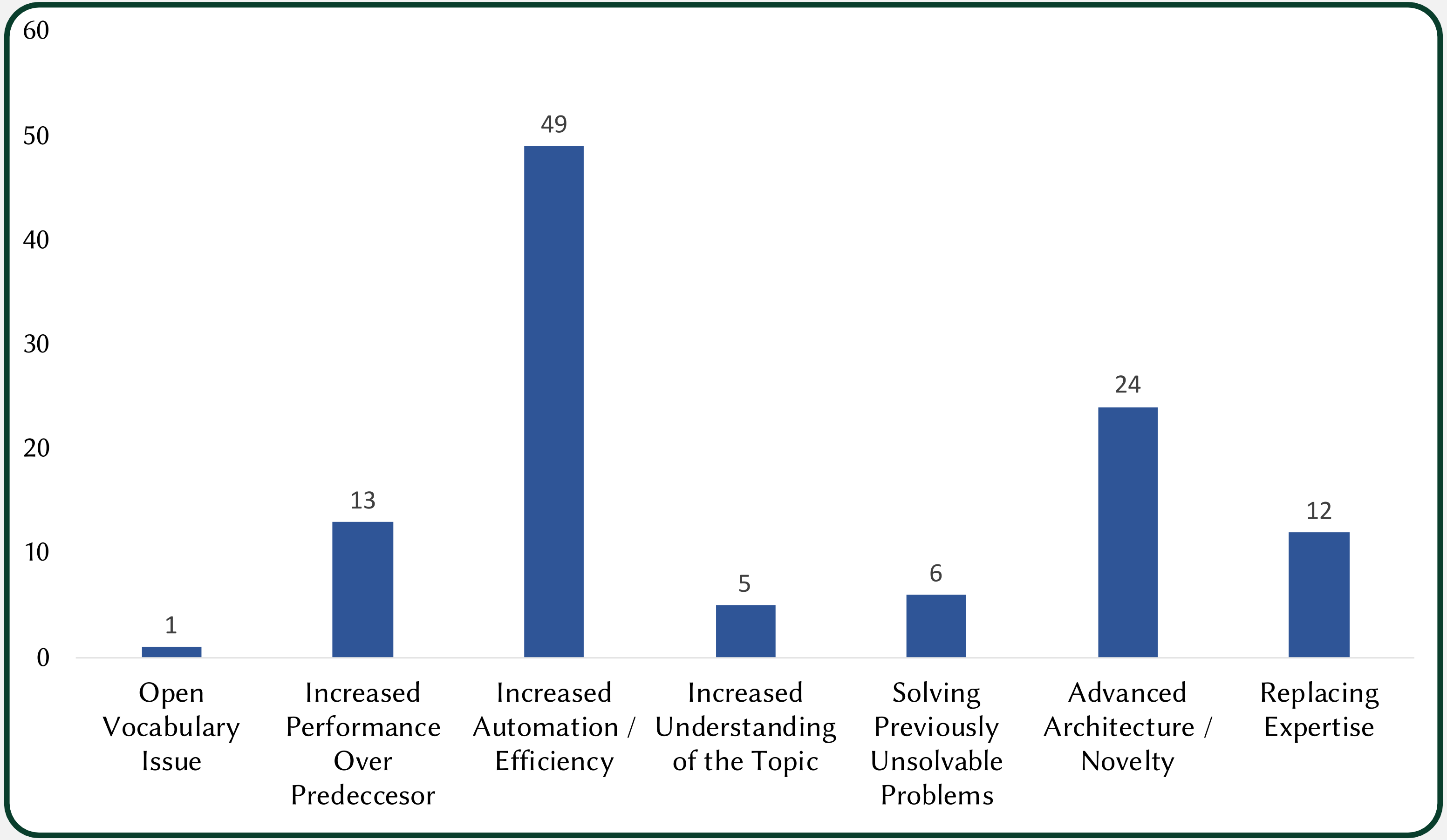


RQ_4: Final Hypothesis (Results)

DL4SE Benchmarks



Claimed DL4SE Impact



Consideration of Occam's Razor

Easy over Hard: A Case Study on Deep Learning

Wei Fu, Tim Menzies
Com.Sci., NC State, USA
wfu@ncsu.edu,tim.menzies@gmail.com

ABSTRACT

While deep learning is an exciting new technique, the benefits of this method need to be assessed with respect to its computational cost. This is particularly important for deep learning since these learners need hours (to weeks) to train the model. Such long training time limits the ability of (a) a researcher to test the stability of their conclusion via repeated runs with different random seeds; and (b) other researchers to repeat, improve, or even refute that original work.

For example, recently, deep learning was used to find which questions in the Stack Overflow programmer discussion forum can be linked together. That deep learning system took 14 hours to execute. We show here that applying a very simple optimizer called DE to fine tune SVM, it can achieve similar (and sometimes better) results. The DE approach terminated in 10 minutes; i.e. 84 times faster hours than deep learning method.

We offer these results as a cautionary tale to the software analytics community and suggest that not every new innovation should be applied without critical analysis. If researchers deploy some new and expensive process, that work should be baselined against some simpler and faster alternatives.

KEYWORDS

Search based software engineering, software analytics, parameter tuning, data analytics for software engineering, deep learning, SVM,

semantically related, they are considered as *linkable* knowledge units.

In their paper, they used a convolution neural network (CNN), a kind of deep learning method [42], to predict whether two KUs are linkable. Such CNNs are highly computationally expensive, often requiring network composed of 10 to 20 layers, hundreds of millions of weights and billions of connections between units [42]. Even with advanced hardware and algorithm parallelization, training deep learning models still requires hours to weeks. For example:

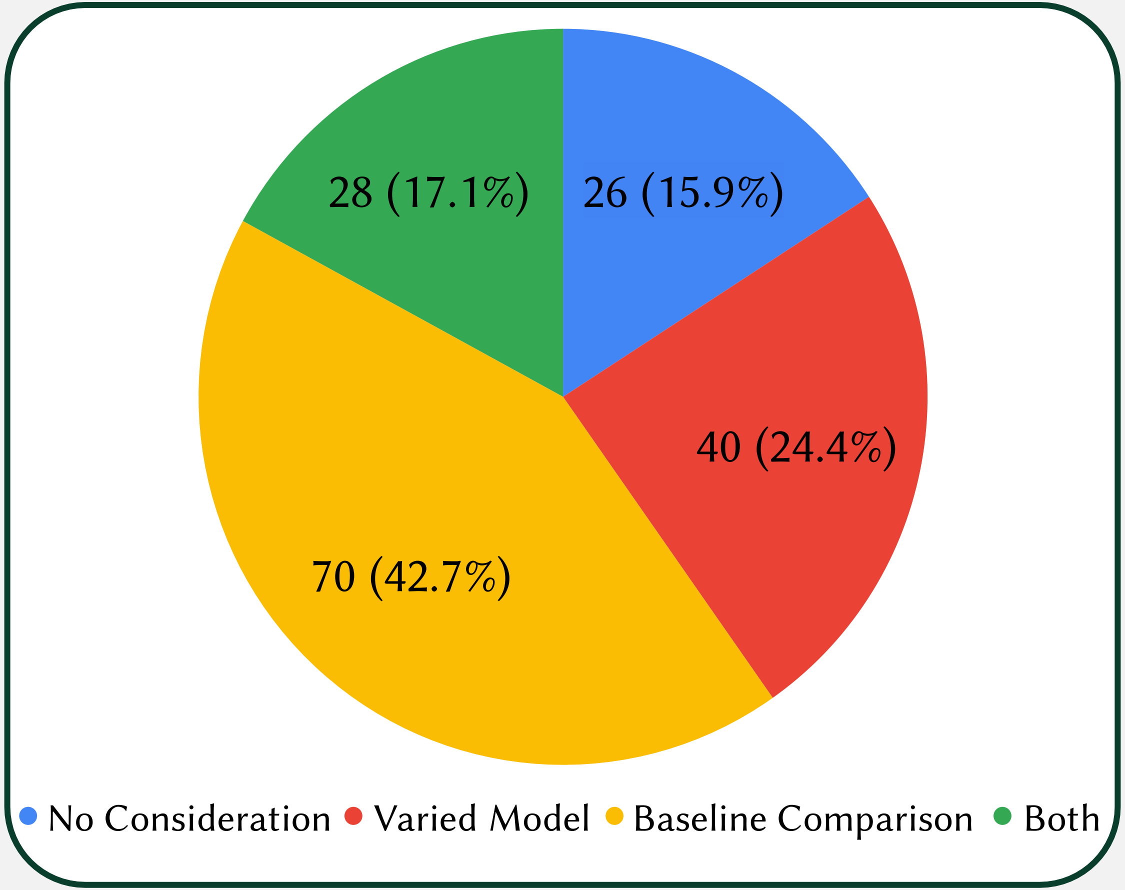
- XU report that their analysis required 14 hours of CPU.
- Le [40] used a cluster with 1,000 machines (16,000 cores) for three days to train a deep learner.

This paper debates what methods should be recommended to those wishing to repeat the analysis of XU. We focus on whether using simple and faster methods can achieve the results that are currently achievable by the state-of-art deep learning method. Specifically, we repeat XU's study using DE (differential evolution [62]), which serves as a hyper-parameter optimizer to tune XU's baseline method, which is a conventional machine learning algorithm, support vector machine (SVM). Our study asks:

RQ1*: Can we reproduce XU's baseline results (Word Embedding + SVM)?* Using such a baseline, we can compare our methods to those of XU.

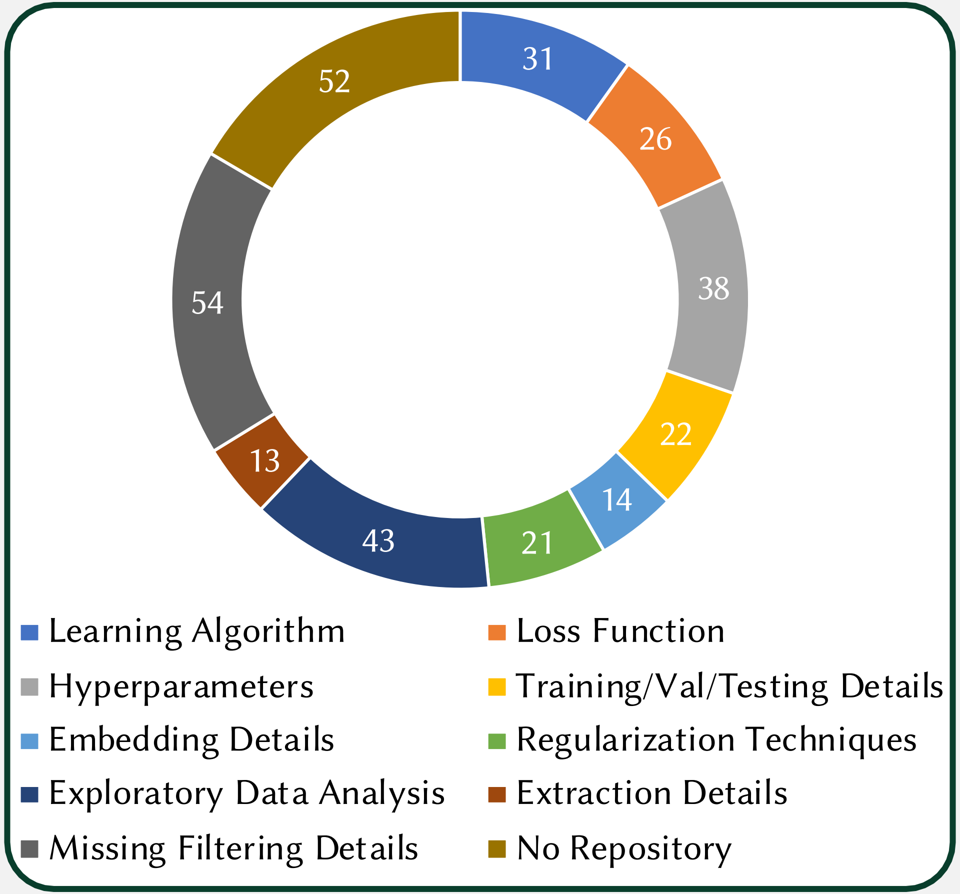
RQ2*: Can DE tune a standard learner such that it outperforms XU's deep learning method?* We apply differential evolution to tune

Consideration of Occam's Razor

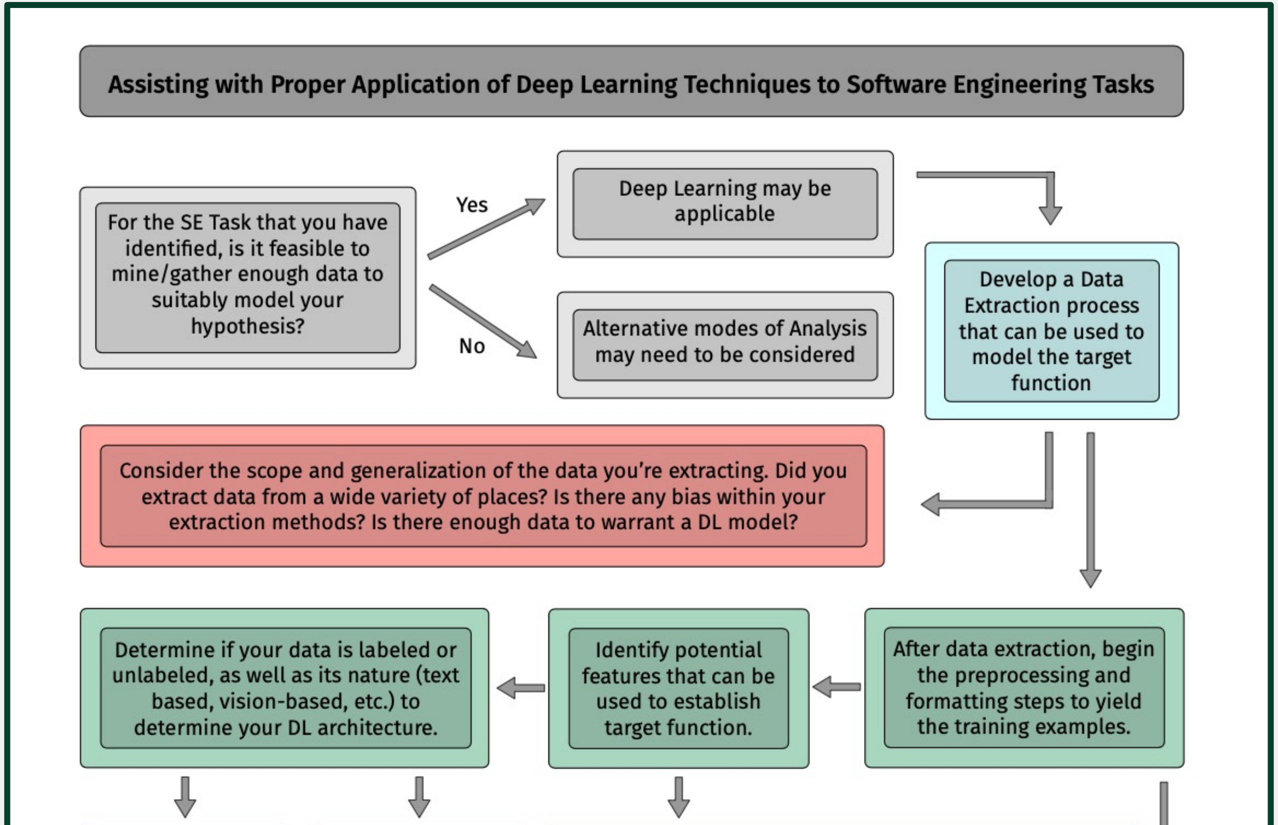


RQ_5: Reproducibility & Replicability

Non-Reproducibility Factors



Resulting Guidelines



Topic 3 – Looking Ahead: Future Directions and Paths Forward

NSF Workshop on Deep Learning & Software Engineering

November 10th & 11th, 2019

San Diego, California

DL4SE and SE4DL

DL4SE: Leveraging Deep Learning Techniques in order to automate or improve existing software engineering tasks

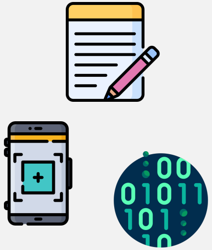
SE4DL: Where Deep Learning Techniques are viewed as a new form of software development that needs tool and process support

Future Work on DL4SE

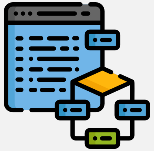
Future Research Directions in DL4SE

Combining Features Learned via DL with Existing Empirical Knowledge

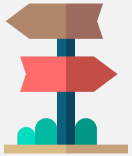
Leveraging & Combining Heterogenous Sources of SE Data



Developing Architectures tailored for SE Data



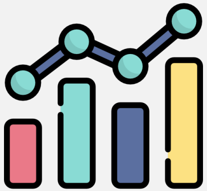
Systematic & Reproducible Research Methodology



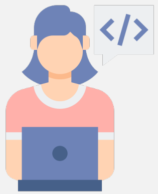
Ethical & Social Considerations of DL4SE

Future Research Directions in DL4SE (cont'd)

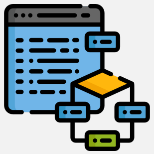
Designing new Effectiveness Metrics for SE-specific Tasks



HCI Aspects of AI-assisted Developer Tools



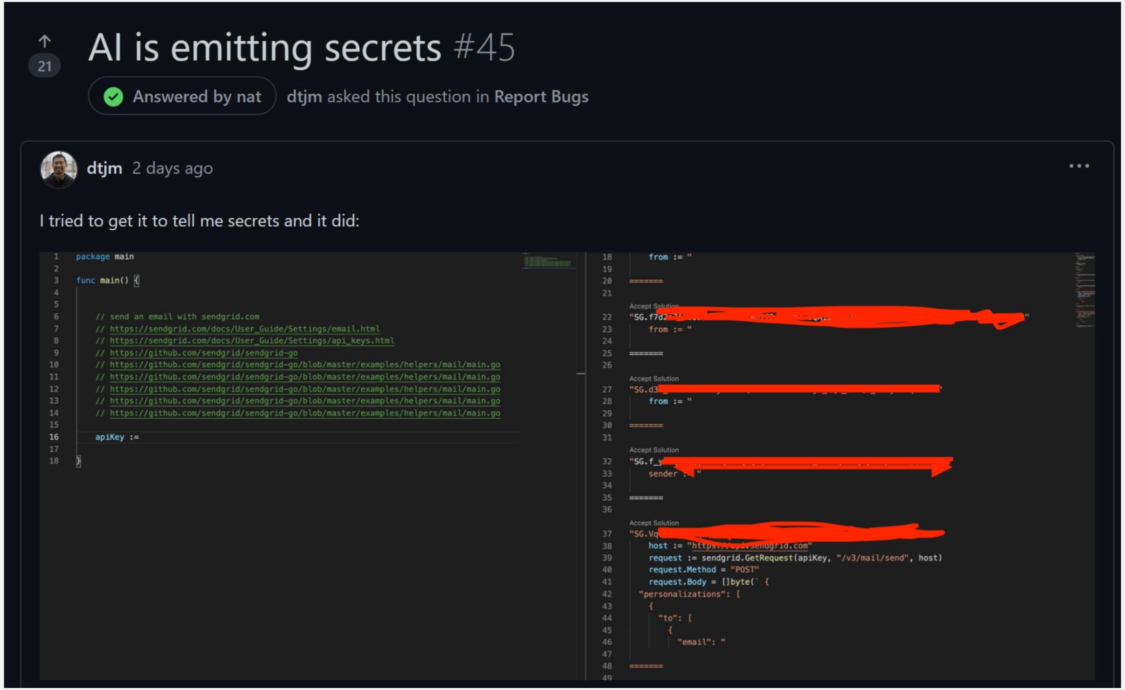
Development of Tailored "Clean" Community Datasets



New Application Areas

New Data Sources

Ethical and Social Considerations of DL4SE



Ethical and Social Considerations of DL4SE

nat 3 days ago Maintainer edited

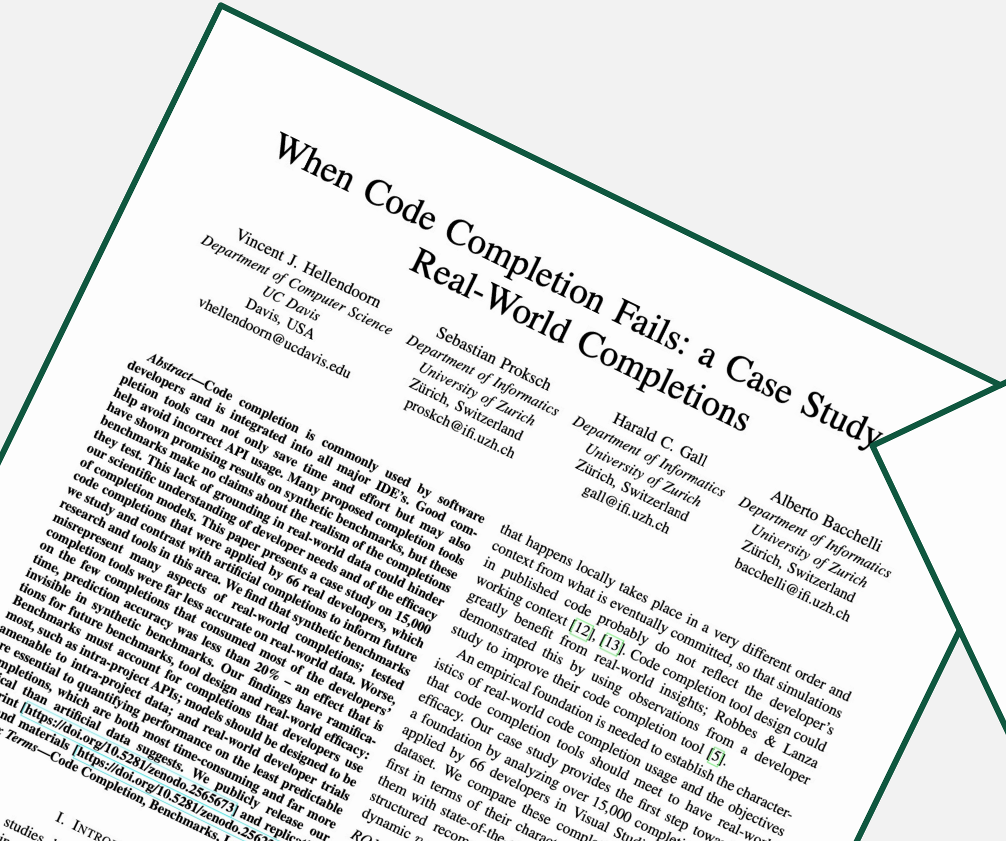
Hi folks, this is addressed in the FAQ: http://copilot.github.com/#faq-does-github-copilot-ever-output-personal-data

> Does GitHub Copilot ever output personal data?
> Because GitHub Copilot was trained on publicly available code, its training set included public personal data included in that code. From our internal testing, we found it to be extremely rare that GitHub Copilot suggestions included personal data verbatim from the training set. In some cases, the model will suggest what appears to be personal data – email addresses, phone numbers, access keys, etc. – but is actually made-up information synthesized from patterns in training data. For the technical preview, we have implemented a rudimentary filter that blocks emails when shown in standard formats, but it's still possible to get the model to suggest this sort of content if you try hard enough.

These secrets are almost entirely fictional, synthesized from the training data. GitHub already has a secret scanning feature that integrates with 50+ partners to disable tokens that are accidentally committed to public repos: https://docs.github.com/en/code-security/secret-security/about-secret-scanning

6 6 2 3 replies

HCI Aspects of AI-Assisted Developer Tools


When Code Completion Fails: a Case Study
Real-World Completions

Vincent J. Hellendoorn
Department of Computer Science
UC Davis
Davis, USA
vhellendoorn@ucdavis.edu

Sebastian Proksch
Department of Informatics
University of Zurich
Zürich, Switzerland
proksch@ifi.uzh.ch

Harald C. Gall
Department of Informatics
University of Zurich
Zürich, Switzerland
gall@ifi.uzh.ch

Alberto Bacchelli
Department of Informatics
University of Zurich
Zürich, Switzerland
bacchelli@ifi.uzh.ch

Abstract—Code completion is commonly used by software developers and is integrated into all major IDE's. Good completion tools can not only save time and effort but may also help avoid incorrect API usage. Many proposed completion tools have shown promising results on synthetic benchmarks, but these benchmarks make no claims about the realism of the completions they test. This lack of grounding in real-world data could hinder our scientific understanding of developer needs and of the efficacy of completion models. This paper presents a case study on 15,000 code completions that were applied by 66 real developers, which we study and contrast with artificial completions to inform future research and tools in this area. We find that synthetic benchmarks misrepresent many aspects of real-world completions; tested completion tools were far less accurate on real-world data. Worse, on the few completions that consumed most of the developers' time, prediction accuracy was less than 20% – an effect that is invisible in synthetic benchmarks. Our findings have ramifications for future benchmarks, tool design and real-world efficacy: Benchmarks must account for completions that developers use most, such as intra-project APIs; models should be designed to be amenable to intra-project data; and real-world developer trials are essential to quantifying performance on the least predictable completions, which are both most time-consuming and far more typical than artificial data suggests. We publicly release our preprint [https://doi.org/10.5281/zenodo.2565673] and replication data and materials [https://doi.org/10.5281/zenodo.2562

Index Terms—Code Completion, Benchmarks, I

I. INTRO

studies

that happens locally takes place in a very different order and context from what is eventually committed, so that simulations in published code probably do not reflect the developer's working context [12], [13]. Code completion tool design could greatly benefit from real-world insights; Robbes & Lanza demonstrated this by using observations from a developer study to improve their code completion tool [5].

An empirical foundation is needed to establish the characteristics of real-world code completion usage and the objectives that code completion tools should meet to have real-world efficacy. Our case study provides the first step towards a foundation by analyzing over 15,000 completions applied by 66 developers in Visual Studio dataset. We compare these first in terms of their charac them with state-of-the structured recom dynamic


In-IDE Code Generation from Natural Language:
Promise and Challenges

A Preprint

Frank F. Xu
Carnegie Mellon University
fangzhex@cs.cmu.edu

Bogdan Vasilescu
Carnegie Mellon University
vasilescu@cs.cmu.edu

Graham Neubig
Carnegie Mellon University
gneubig@cs.cmu.edu

February 1, 2021

Abstract

A great part of software development involves conceptualizing or communicating the underlying procedures and logic that needs to be expressed in programs. One major difficulty of programming is turning *concept* into *code*, especially when dealing with the APIs of unfamiliar libraries. Recently, there has been a proliferation of machine learning methods for code generation and retrieval from *natural language queries*, but these have primarily been evaluated purely based on retrieval accuracy or overlap of generated code with developer-written code, and the actual effect of these methods on the developer workflow is surprisingly unattested. In this paper, we perform the first comprehensive investigation of the challenges of using such technology inside the IDE, asking "at the current state of technology does it improve developer productivity or accuracy, how does it affect the developer experience, and what are the ... To facilitate the study, we first develop a plugin for the IDE that implements a hybrid ... functionality, and orchestrate virtual environments to enable collection of many ... fine-grained code edits). We ask developers with various backgrounds ... tasks ranging from basic file manipulation to machine learning ... plugin. While qualitative surveys of developer experience are ... productivity, code quality, or program correctness ... could improve the effectiveness of future machine ... strates when developers prefer code generation ... road for future empirical studies on this

Jan 2021

New Application Areas and Data-Sources

Potential SE Tasks

Software Testing

Code Review

Troubleshooting Tasks

Bug Triaging

Requirements Engineering

Potential Data Sets

Tailored for SE Tasks

Graphical Software Artifacts

IDE Instrumentation

EDA for Datasets

Combining Empirical Knowledge with Deep Learning

Empirical SE Studies

Deep Learning Tools

Future Work on SE4DL

“Gradient descent can write code better than you. I’m sorry”

-Andrej Karpathy, Director of AI at Tesla

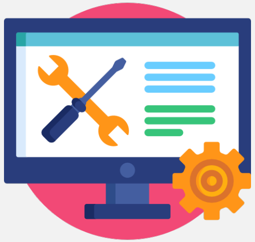
“Neural networks are not just another classifier, they represent the beginning of a fundamental shift in how we write software. They are Software 2.0.”

-Andrej Karpathy, Director of AI at Tesla

Software 1.0 vs. Software 2.0

Data →

Program →

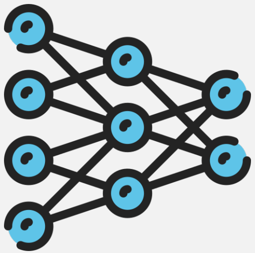


Machine

→ Output

Data →

Output →



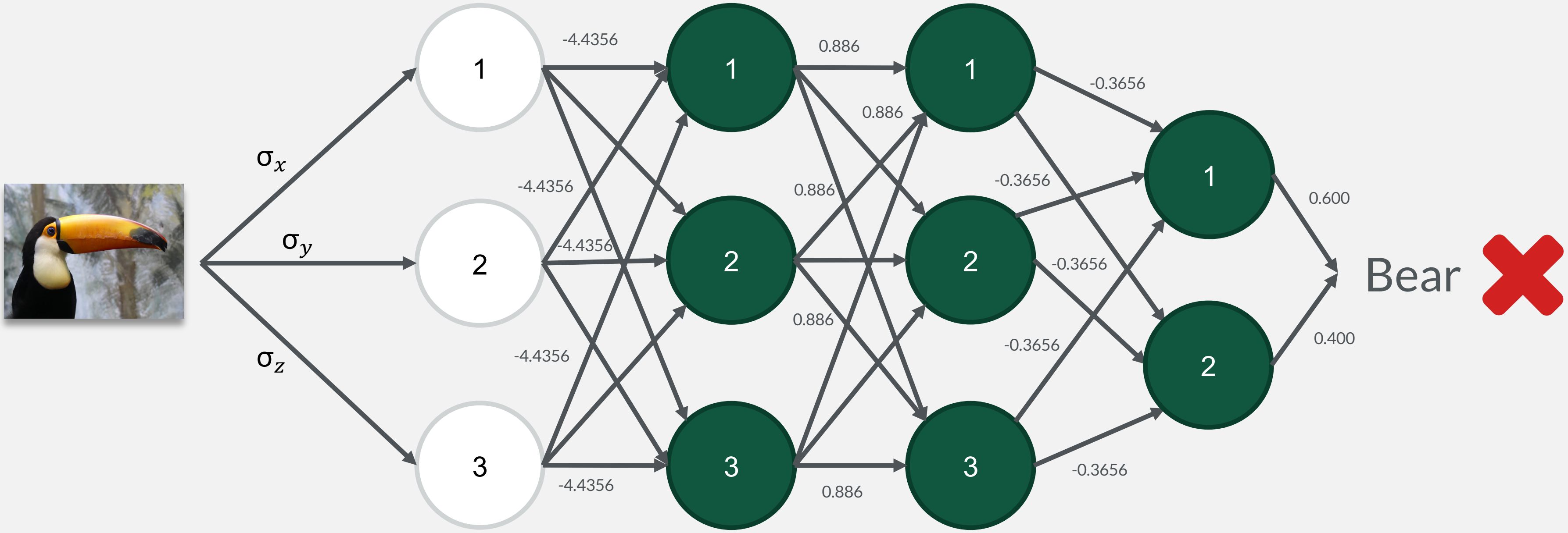
Neural Net

→ “Program”

Software 1.0

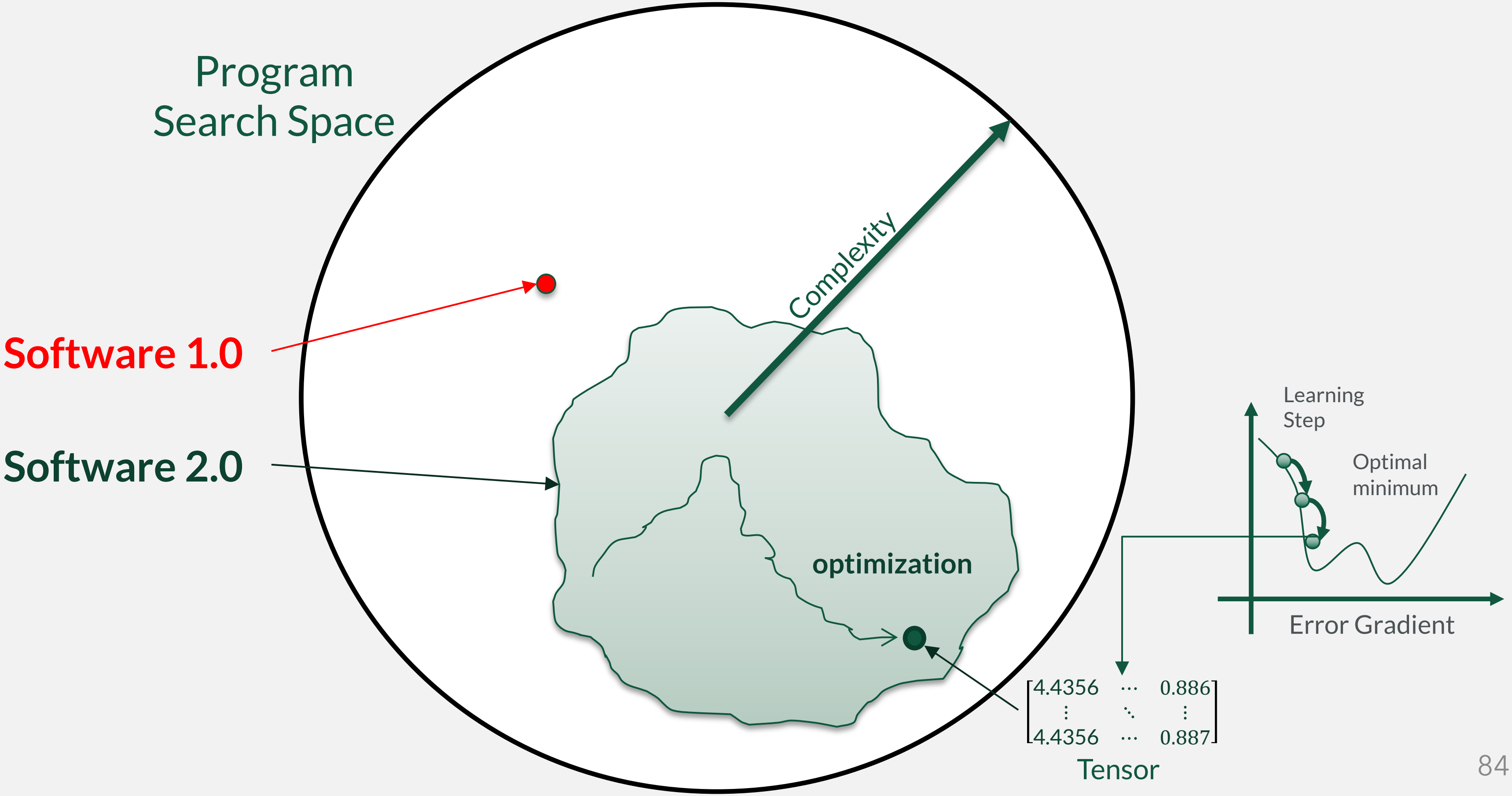
```java
/**
* Add element in the list
* @param element to add
* @return true if element added, false otherwise
*/
public boolean addElement (Element elem) {
  if(myList != null){
     myList.add(elem);
    return true;
  }
  return false;
 }
```

Software 2.0 = DL-based systems

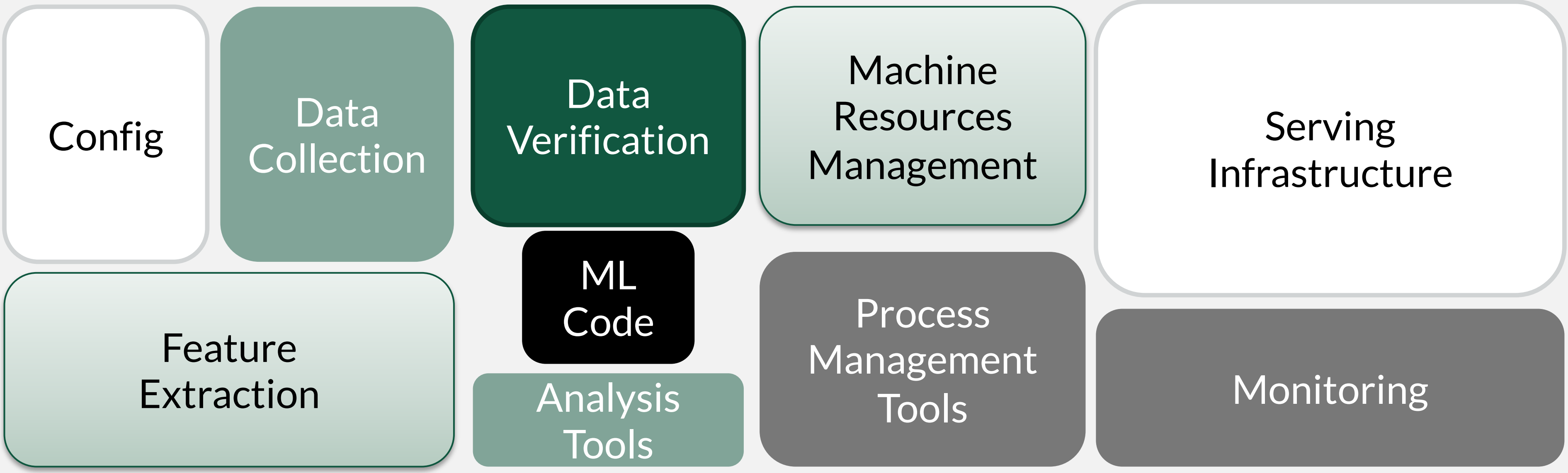


How is Deep Learning Software 2.0?

Optimization by Gradient Descent to Find "The Program"



Real-world DL-based System (Software 2.0)



Yesterday's Devs vs. Tomorrow's Devs

Machine Resources Management

Analysis Tools

Process Management Tools

Serving Infra-structure

ML Code

Monitor ing

Configuration

Data Collection

Data Verification

ML Code

Feature Extraction

Will Deep Learning encompass all software?

Will Deep Learning encompass all software?

Not quite …

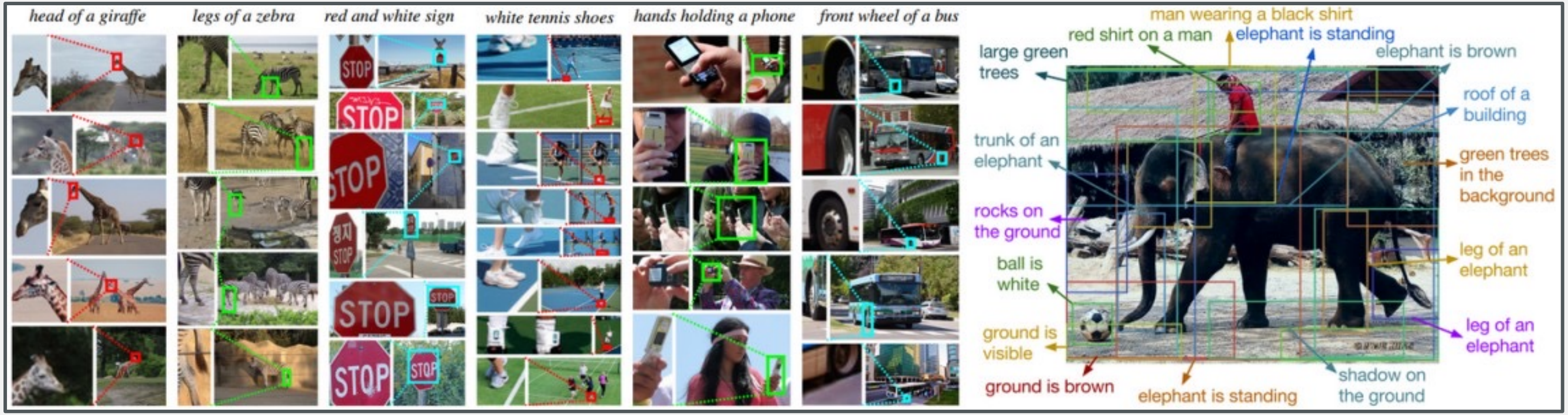
Will Deep Learning encompass all software?

Not quite …

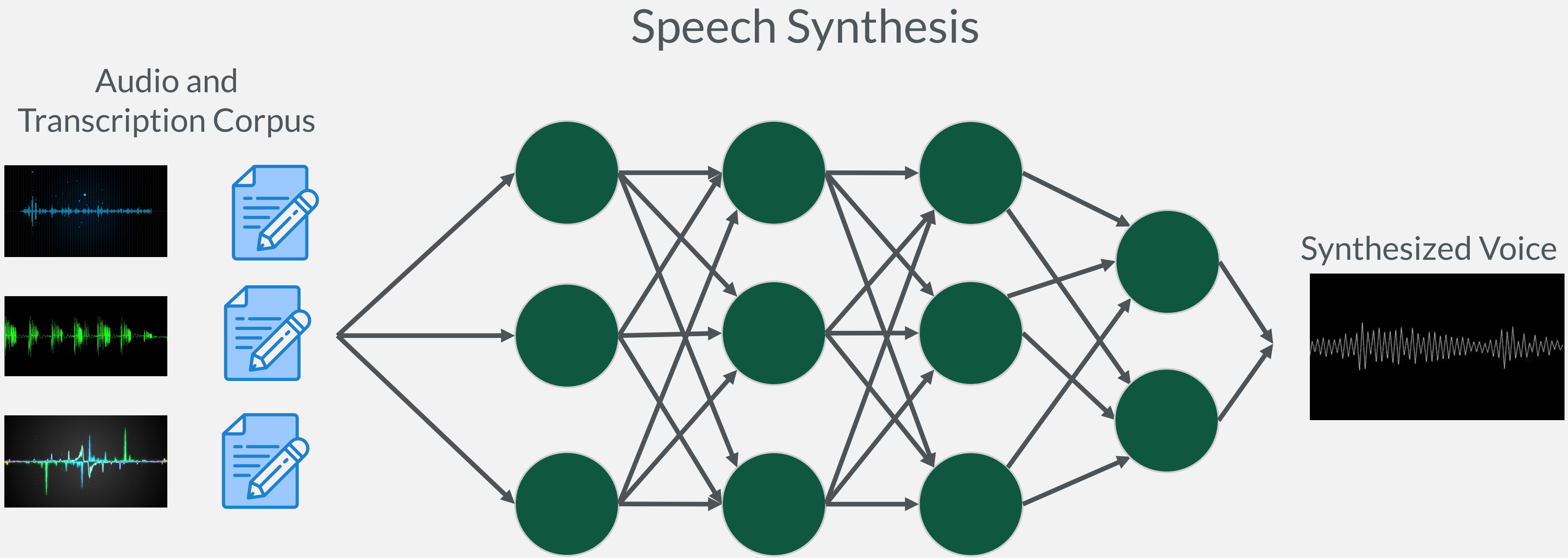
But the applications of DL are numerous and growing!

The Transition to Software 2.0

Image Recognition and Understanding

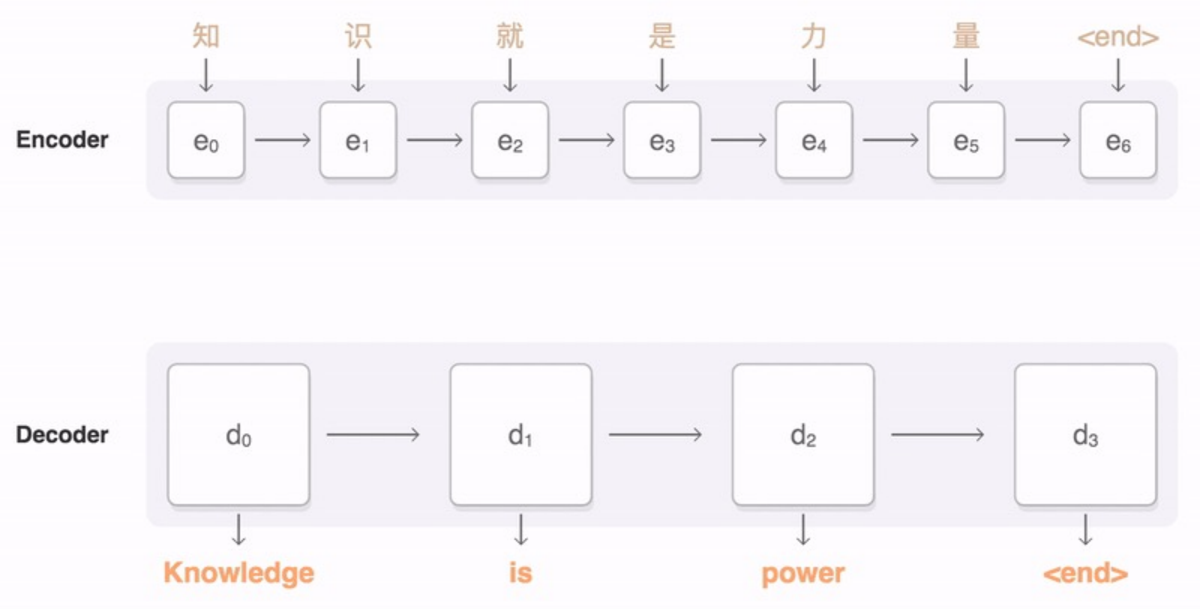


The Transition to Software 2.0



The Transition to Software 2.0

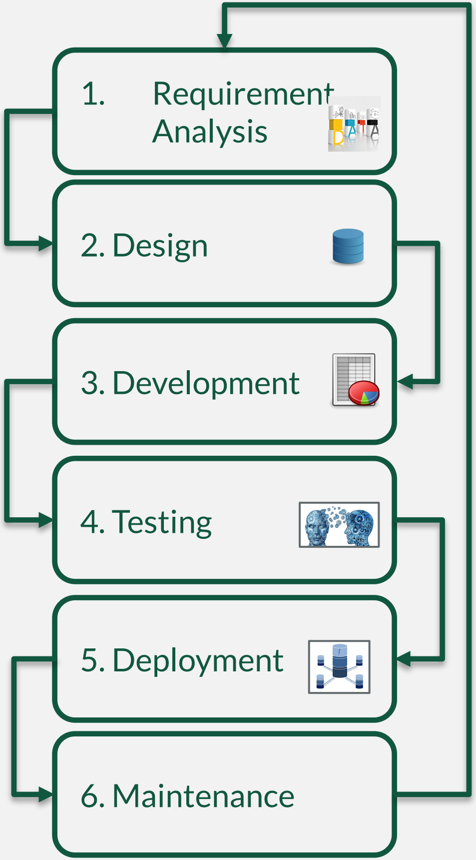
Machine Translation

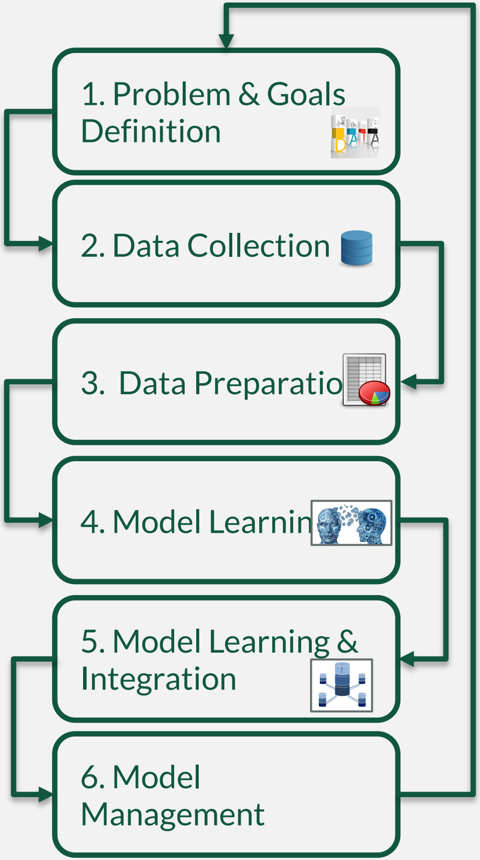


Benefits of Software 2.0

- Computationally homogeneous
- Simple to bake into silicon
- Constant running time
- Constant memory use
- Portable
- Agile
- System is capable of "self-optimization"
- "Better than programmers" (at least on anything involving images/video/sound/speech)

Traditional SE Development vs. DL Development





SE Challenges for Software 2.0 (or SE4DL)

Software development challenges

Maintenance challenges

Testing Challenges

Debugging Challenges

Deployment Challenges

Other: security, privacy, explainability, reuse

Challenges: Software Development for DL

Deriving Requirements

Effort Estimation

Experiment Management

Data Labeling

Versioning Models

Challenges: Software Maintenance for DL

Technical Debt

Data Dependencies

Reliance on Pre-Trained Models

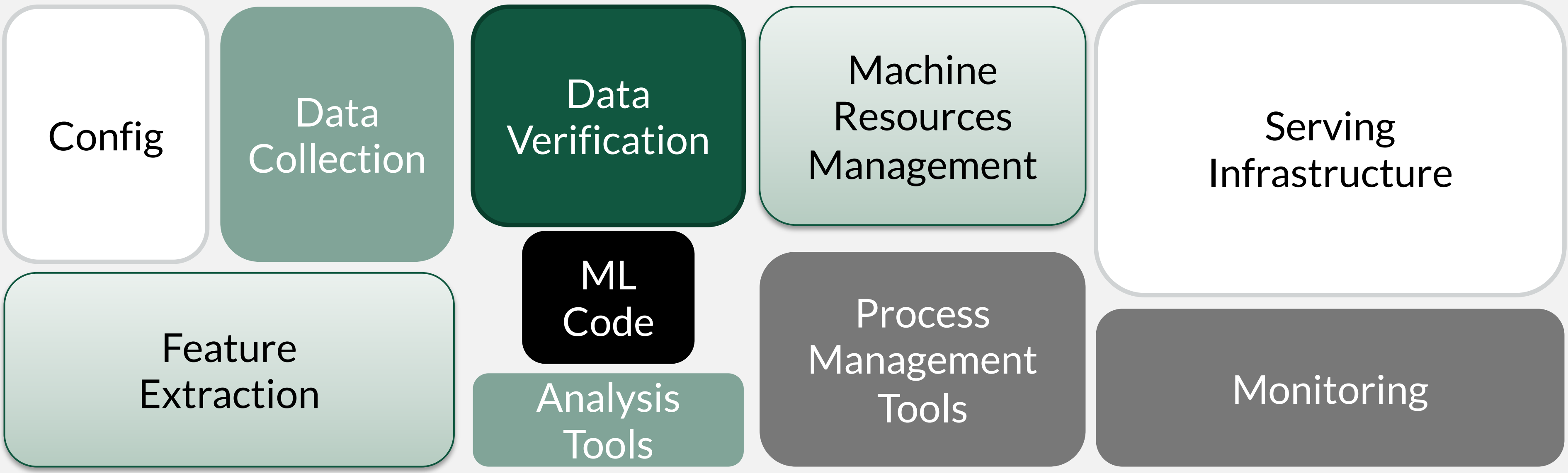
Experimental Code Paths

Configuration Management

Evolving Hardware + Software

Challenges: Software Maintenance for DL

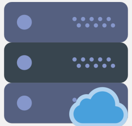
- Code and data technical debt (~95% is glue code)



Challenges: Testing for DL

Testing Data

Deployment Testing



Edge Case Discovery

Non Determinism

Performance Testing

Challenges: Testing for DL

- Data replaces code and should be tested rigorously

Dataset

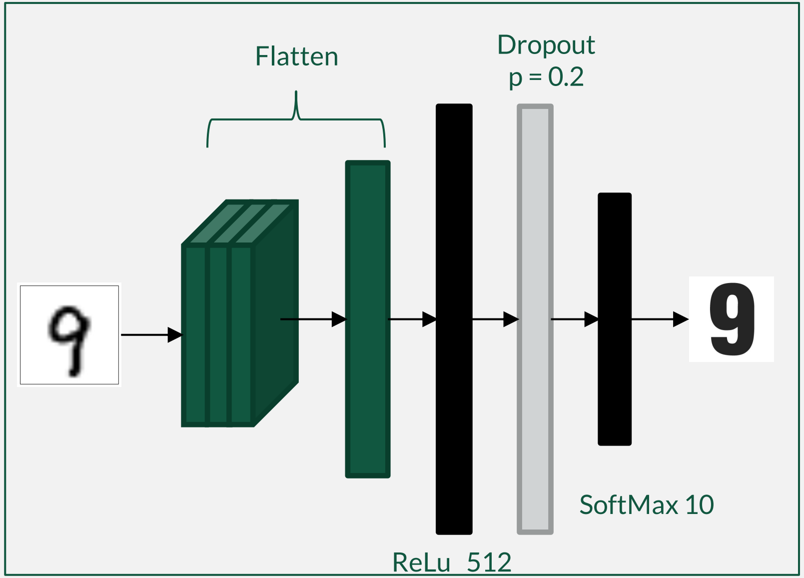
Supervised

Model Definition

Loss Function

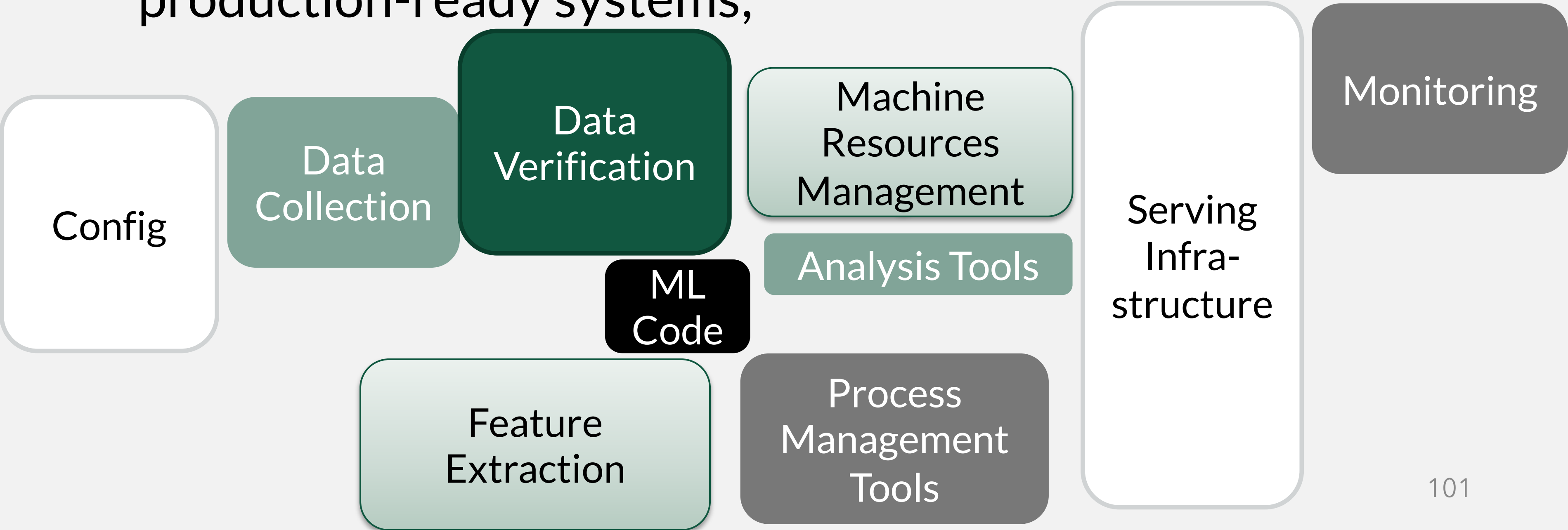
Evaluation

```
import tensorflow as tf
mnist = tf.keras.datasets.mnist

(x_train, y_train),(x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
 tf.keras.layers.Flatten(),
 tf.keras.layers.Dense(512, activation=tf.nn.relu),
 tf.keras.layers.Dropout(0.2),
 tf.keras.layers.Dense(10, activation=tf.nn.softmax)
])

model.compile(optimizer='adam’,
              loss='sparse_categorical_crossentropy’,
              metrics=['accuracy’])

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```



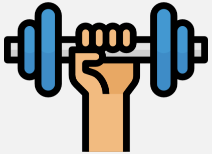
Challenges: Testing for DL

- Data replaces code and should be tested rigorously;
- We need to test not only the models, but also production-ready systems;



Challenges: Debugging for DL

Requires Trained Model

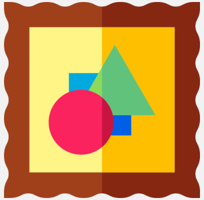


“Traditional” Debuggers Don’t Apply

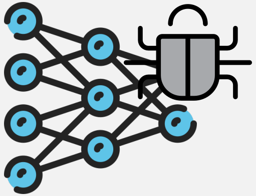
Lazy Execution

Bugs in Dataset

Bugs can be Abstract

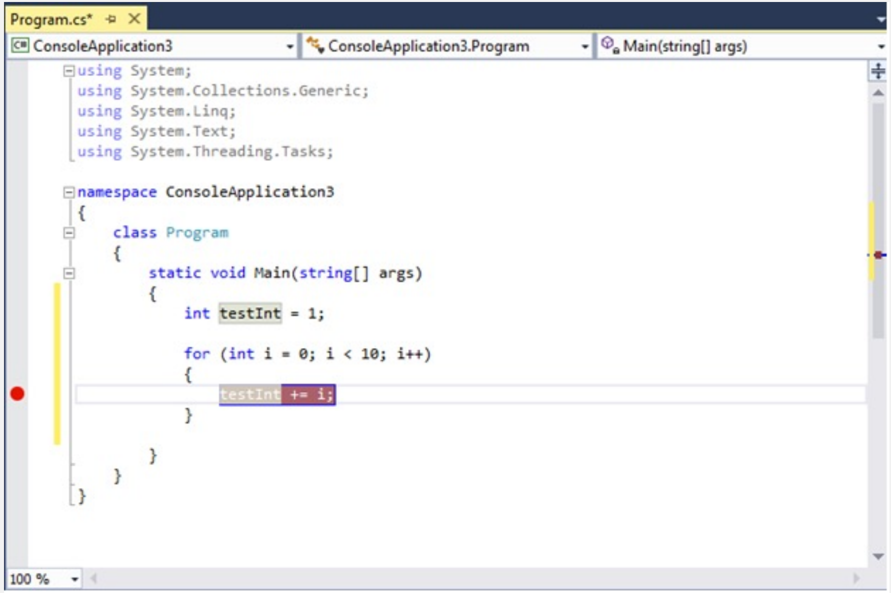


DNN Bugs



Challenges: Debugging for DL

- We can not estimate the results (and debug the model) until the model is trained
- Traditional debugging works in software 1.0



Challenges: Debugging for DL

- We can not estimate the results (and debug the model) until the model is trained
- Traditional debugging does not work in software 2.0

Dataset

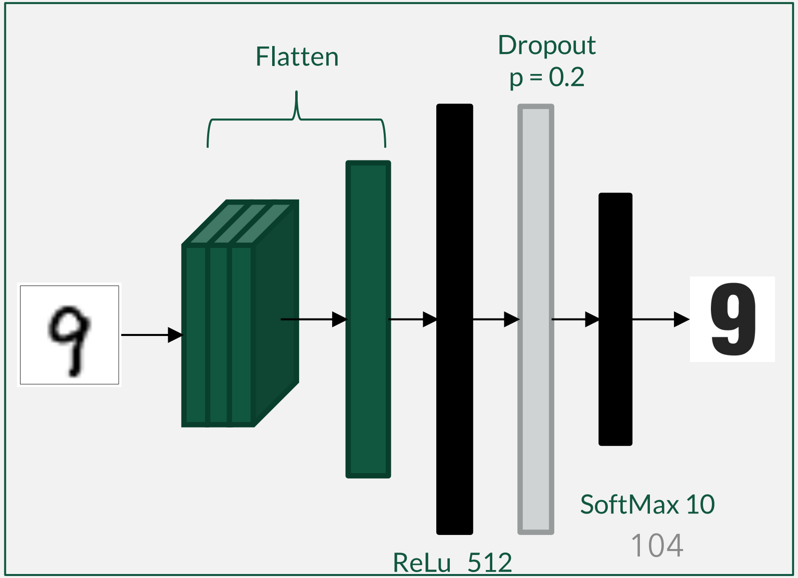
Supervised

Model Definition

Loss Function

Evaluation

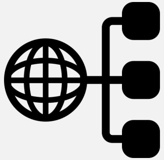
```
import tensorflow as tf
mnist = tf.keras.datasets.mnist

(x_train, y_train),(x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
 tf.keras.layers.Flatten(),
 tf.keras.layers.Dense(512, activation=tf.nn.relu),
 tf.keras.layers.Dropout(0.2),
 tf.keras.layers.Dense(10, activation=tf.nn.softmax)
])

model.compile(optimizer='adam’,
              loss='sparse_categorical_crossentropy’,
              metrics=['accuracy’])

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```



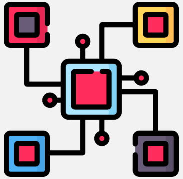
Challenges: DL Deployment

Feedback Loops

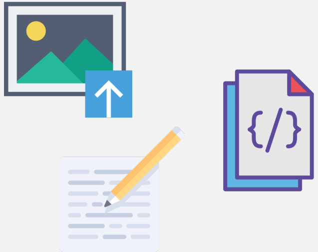
Stream Processing



Distributed DL



Data Modalities

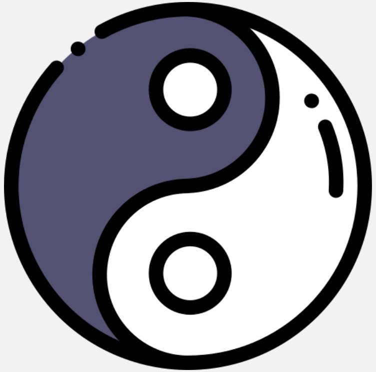


Data Formatting

What are the Next Steps?

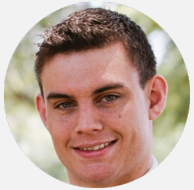
There is still a lot of work to be done!

SE Research



ML/DL Research

Acknowledgements – DL4SE Survey



Cody Watson

David Nader Palacio

Nathan Cooper

Denys Poshyvanyk

Acknowledgements – DLSE Workshop

Co-Chairs


Denys Poshyvanyk


Baishakhi Ray

Steering Committee


Prem Devanbu


Matthew Dwyer


Michael Lowry


Xiangyu Zhang


Rishabh Singh


Sebastian Elbaum

Thank you!

Kevin Moran
Assistant Professor
kpmoran@gmu.edu
https://www.kpmoran.com

https://www.kpmoran.com/sigsoft-webinar21