Analyzing Software using Deep Learning

Reasoning about Types and Code Changes with Hierarchical Networks (Part 1)

Prof. Dr. Michael Pradel

Software Lab, University of Stuttgart

Summer 2020

Overview

- **Hierarchical neural networks** ←
- **Type prediction**

  Based on “TypeWriter: Neural Type Prediction with Search-based Validation” by Pradel et al., 2020
- **Representing code changes**

  Based on “CC2Vec: Distributed Representations of Code Changes” by Hoang et al., 2020

Motivation

- **What if input to a predictive model consists of multiple parts that**
  - are too many to simply concatenate
  - are not a sequence
  - may each have a different structure?

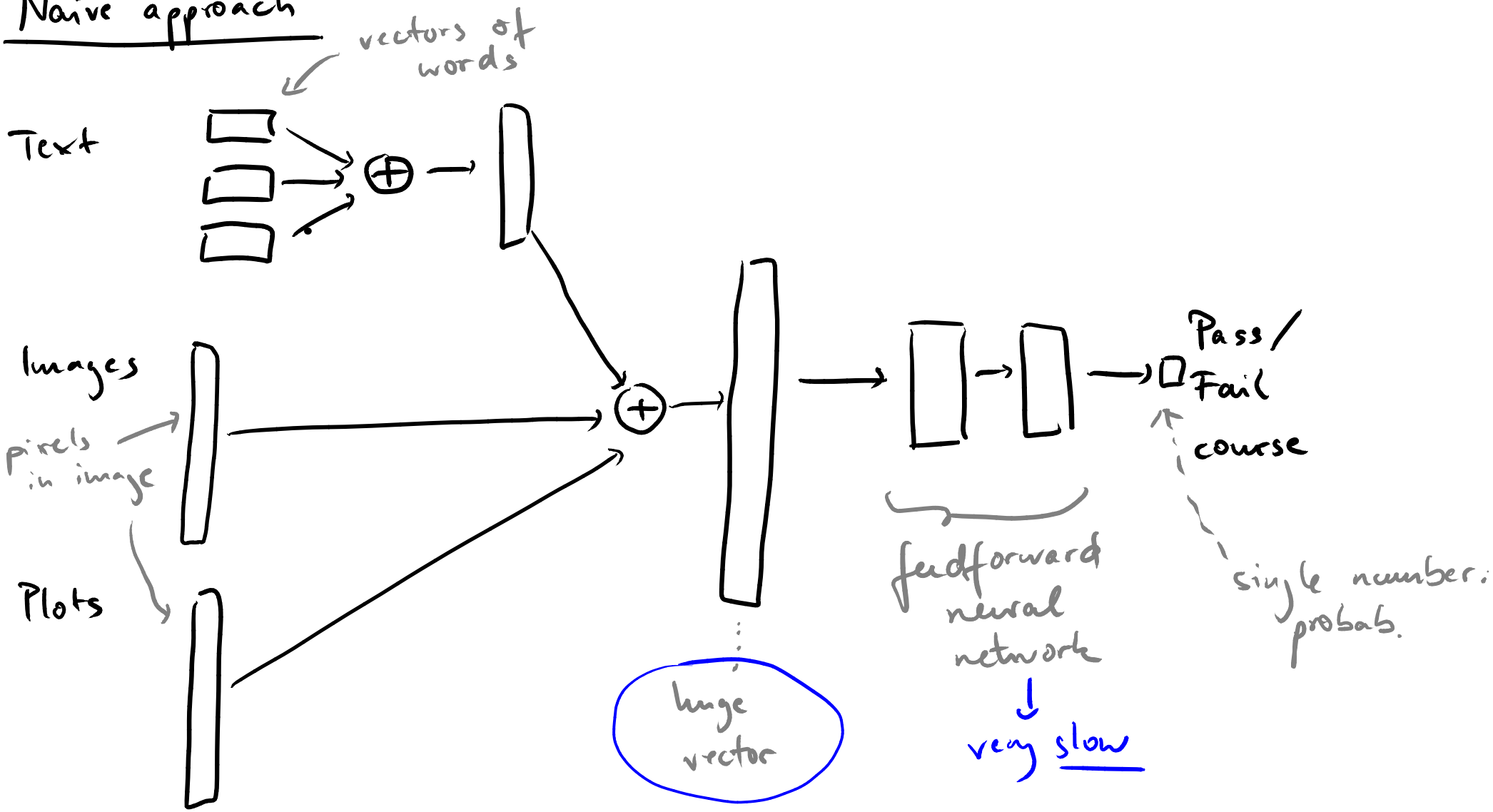
Examples

- **Document**
  - Lines of words
  - Images
  - Plots
- **Evidence of program crash**
  - Stack trace
  - Error message
  - Information about the user (key-value pairs)

Examples (2)

- **Program elements that have a type**
  - Code tokens
  - Identifier names
  - Comments associated with the function
- **Commits to a code repository**
  - Code change
    - Multiple code locations
  - Commit messages

Naive approach



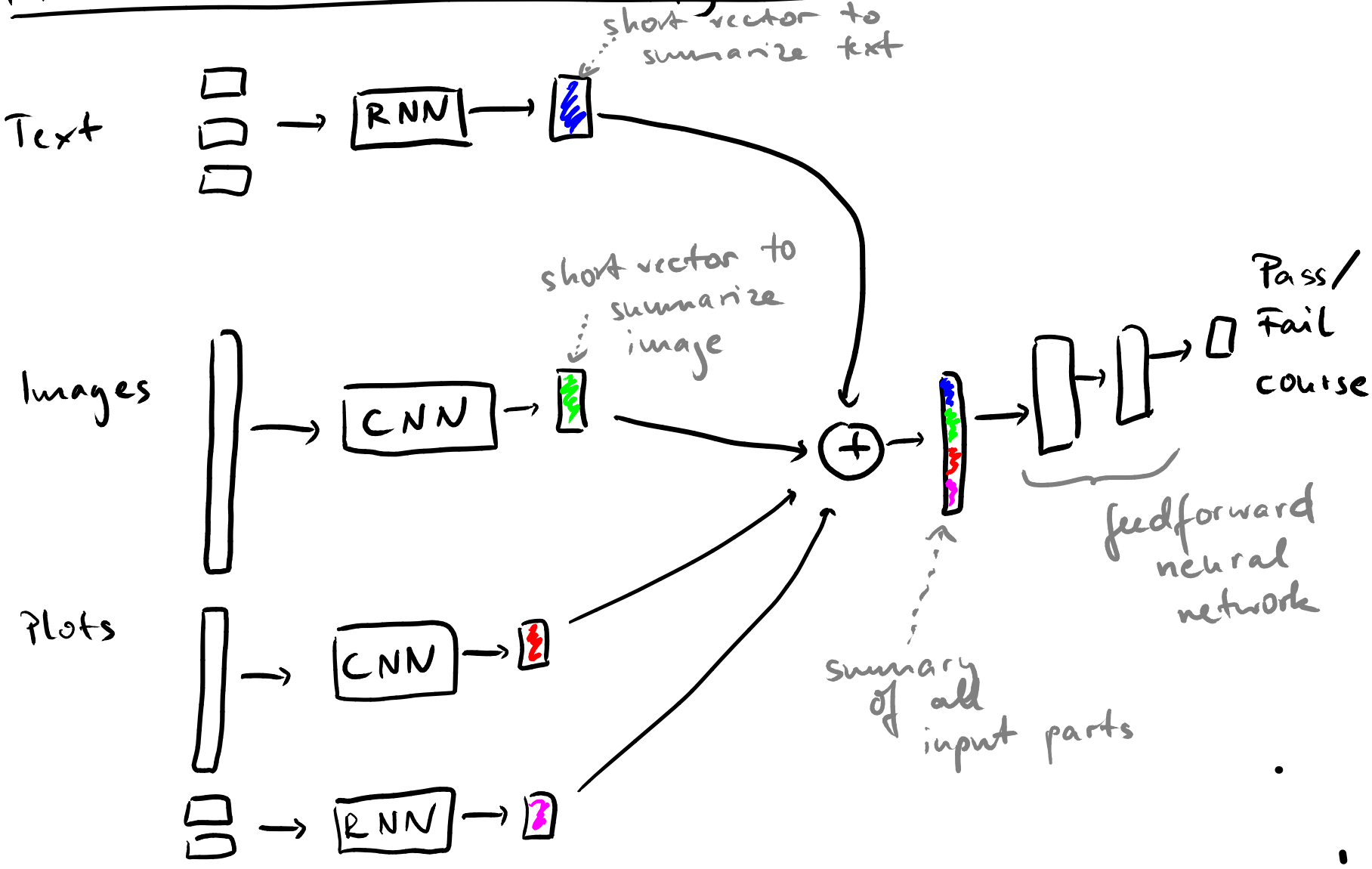
Hierarchical Neural Networks

- **Neural model composed of submodels**
- **Aligned into a hierarchy**
  - E.g., a tree where inputs arrive at leaves
  - Information propagates from leaves to the root
- **Prediction based on summarized information at root**

Submodels

- **Each submodel: Encode specific part of input**
- **Different submodels may be different kinds of neural networks**
  - E.g., feedforward network for some input, RNN for some other input

Hierarchical Model to Classify Documents



Jointly Training the Model

How to train a hierarchical neural network?

Option 1: Train each submodel separately

- ✓ Training focuses on specific model and its input
- ✗ Need training data for each submodel
- ✗ Submodel isn't aware of the overall task

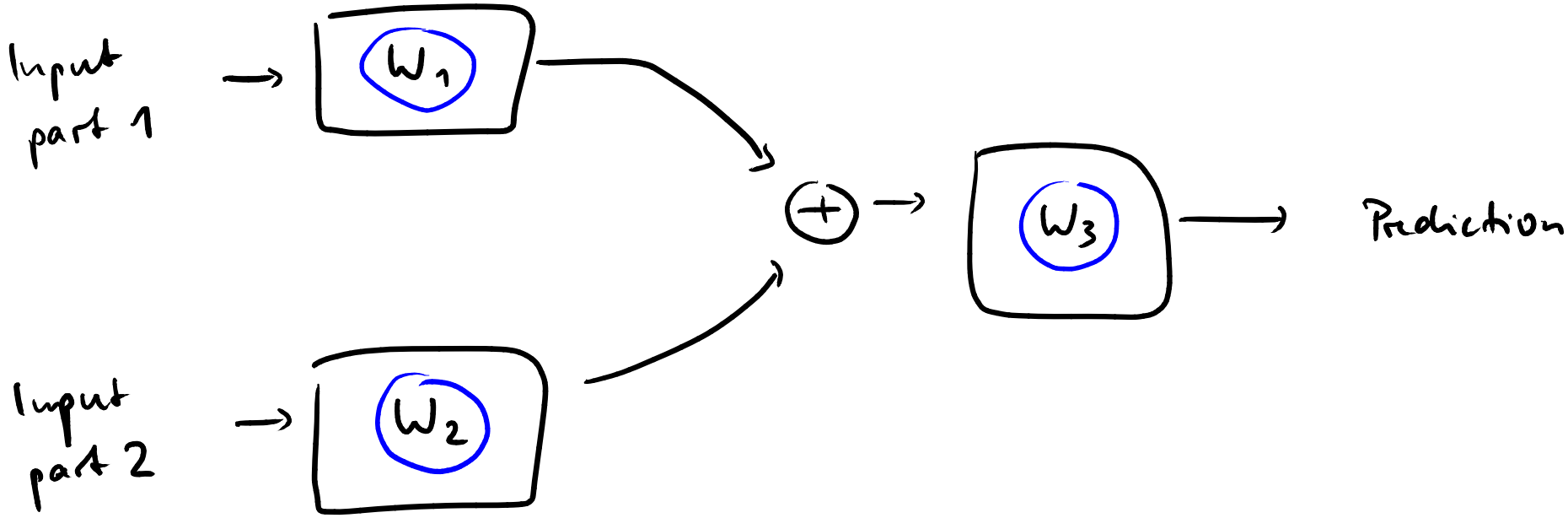
Jointly Training the Model

How to train a hierarchical neural network?

Option 2: Train entire model jointly

- ✓ Need training data only for the overall task
- ✓ Submodels get optimized for the overall task
- ✗ For large models, feedback from final prediction may get lost (vanishing gradient problem)

Example: Joint Training



i_1
i_2
$\vdots$

o_1
o_2
$\vdots$