

Review

DL Libraries

Introduction

Symbolic
Programming
Graphs
Neurons

Fitting Models

Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project
Improvement

Where We Are
Model Tuning

Up Next

Deep Learning with TensorFlow

Deep Learning — Units 7 & 8

Dr. Jon Krohn

`jon@untapt.com`

Slides available at `jonkrohn.com/talks`

November 23, 2019

Outline

- 1 Review Take-Home Exercise
- 2 Comparison of the Leading Deep Learning Libraries
- 3 Introduction to TensorFlow
- 4 Fitting Models
- 5 TF 2.0
- 6 PyTorch
- 7 Deep Learning Project IV: Improving
- 8 Up Next: Advanced Topics

Review

DL Libraries

Introduction

Symbolic
Programming
Graphs
Neurons

Fitting Models

Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project
Improvement

Where We Are
Model Tuning

Up Next

Outline

- 1 Review Take-Home Exercise
- 2 Comparison of the Leading Deep Learning Libraries
- 3 Introduction to TensorFlow
- 4 Fitting Models
- 5 TF 2.0
- 6 PyTorch
- 7 Deep Learning Project IV: Improving
- 8 Up Next: Advanced Topics

Review

DL Libraries

Introduction

Symbolic
Programming
Graphs
Neurons

Fitting Models

Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project
Improvement

Where We Are
Model Tuning

Up Next

Outline

Review

DL Libraries

Introduction

Symbolic
Programming
Graphs
Neurons

Fitting Models

Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project
Improvement

Where We Are
Model Tuning

Up Next

- 1 Review Take-Home Exercise
- 2 Comparison of the Leading Deep Learning Libraries
- 3 Introduction to TensorFlow
- 4 Fitting Models
- 5 TF 2.0
- 6 PyTorch
- 7 Deep Learning Project IV: Improving
- 8 Up Next: Advanced Topics

Outline

Review

DL Libraries

Introduction

Symbolic
Programming
Graphs
Neurons

Fitting Models

Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project
Improvement

Where We Are
Model Tuning

Up Next

- 1 Review Take-Home Exercise
- 2 Comparison of the Leading Deep Learning Libraries
- 3 Introduction to TensorFlow
- 4 Fitting Models
- 5 TF 2.0
- 6 PyTorch
- 7 Deep Learning Project IV: Improving
- 8 Up Next: Advanced Topics

Outline

Review

DL Libraries

Introduction

Symbolic
Programming
Graphs
Neurons

Fitting Models

Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project
Improvement

Where We Are
Model Tuning

Up Next

- 1 Review Take-Home Exercise
- 2 Comparison of the Leading Deep Learning Libraries
- 3 Introduction to TensorFlow
- 4 Fitting Models
- 5 TF 2.0
- 6 PyTorch
- 7 Deep Learning Project IV: Improving
- 8 Up Next: Advanced Topics

Outline

Review

DL Libraries

Introduction

Symbolic
Programming
Graphs
Neurons

Fitting Models

Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project
Improvement

Where We Are
Model Tuning

Up Next

- 1 Review Take-Home Exercise
- 2 Comparison of the Leading Deep Learning Libraries
- 3 Introduction to TensorFlow
- 4 Fitting Models
- 5 TF 2.0
- 6 PyTorch
- 7 Deep Learning Project IV: Improving
- 8 Up Next: Advanced Topics

Outline

Review

DL Libraries

Introduction

Symbolic
Programming
Graphs
Neurons

Fitting Models

Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project
Improvement

Where We Are
Model Tuning

Up Next

- 1 Review Take-Home Exercise
- 2 Comparison of the Leading Deep Learning Libraries
- 3 Introduction to TensorFlow
- 4 Fitting Models
- 5 TF 2.0
- 6 PyTorch
- 7 Deep Learning Project IV: Improving
- 8 Up Next: Advanced Topics

Outline

Review

DL Libraries

Introduction

Symbolic
Programming
Graphs
Neurons

Fitting Models

Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project
Improvement

Where We Are
Model Tuning

Up Next

- 1 Review Take-Home Exercise
- 2 Comparison of the Leading Deep Learning Libraries
- 3 Introduction to TensorFlow
- 4 Fitting Models
- 5 TF 2.0
- 6 PyTorch
- 7 Deep Learning Project IV: Improving
- 8 Up Next: Advanced Topics

Outline

Review

DL Libraries

Introduction

Symbolic
Programming
Graphs
Neurons

Fitting Models

Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project
Improvement

Where We Are
Model Tuning

Up Next

- 1 Review Take-Home Exercise
- 2 Comparison of the Leading Deep Learning Libraries
- 3 Introduction to TensorFlow
- 4 Fitting Models
- 5 TF 2.0
- 6 PyTorch
- 7 Deep Learning Project IV: Improving
- 8 Up Next: Advanced Topics

Review

DL Libraries

Introduction

Symbolic
Programming
Graphs
Neurons

Fitting Models

Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project
Improvement

Where We Are
Model Tuning

Up Next

Assessing Your Deep Learning Project III



Assessing

Your Deep Learning Project III

Where are you at with respect to the following?

1 Splitting your data

- training set (80% — for optimizing parameters)
- validation set (10% — for hyperparameters)
- test set (10% — don't touch yet!)

2 Building and assessing architecture

- get above chance (simplifying problem, if necessary)
- do existing performance benchmarks exist?
- if not, use a simple architecture as benchmark

Assessing

Your Deep Learning Project III

Where are you at with respect to the following?

1 Splitting your data

- training set (80% — for optimizing parameters)
- validation set (10% — for hyperparameters)
- test set (10% — don't touch yet!)

2 Building and assessing architecture

- get above chance (simplifying problem, if necessary)
- do existing performance benchmarks exist?
- if not, use a simple architecture as benchmark

Assessing

Your Deep Learning Project III

Where are you at with respect to the following?

1 Splitting your data

- training set (80% — for optimizing parameters)
- validation set (10% — for hyperparameters)
- test set (10% — don't touch yet!)

2 Building and assessing architecture

- get above chance (simplifying problem, if necessary)
- do existing performance benchmarks exist?
- if not, use a simple architecture as benchmark

Assessing

Your Deep Learning Project III

Where are you at with respect to the following?

1 Splitting your data

- training set (80% — for optimizing parameters)
- validation set (10% — for hyperparameters)
- test set (10% — don't touch yet!)

2 Building and assessing architecture

- get above chance (simplifying problem, if necessary)
- do existing performance benchmarks exist?
- if not, use a simple architecture as benchmark

Assessing

Your Deep Learning Project III

Where are you at with respect to the following?

1 Splitting your data

- training set (80% — for optimizing parameters)
- validation set (10% — for hyperparameters)
- test set (10% — don't touch yet!)

2 Building and assessing architecture

- get above chance (simplifying problem, if necessary)
- do existing performance benchmarks exist?
- if not, use a simple architecture as benchmark

Assessing

Your Deep Learning Project III

Where are you at with respect to the following?

1 Splitting your data

- training set (80% — for optimizing parameters)
- validation set (10% — for hyperparameters)
- test set (10% — don't touch yet!)

2 Building and assessing architecture

- get above chance (simplifying problem, if necessary)
- do existing performance benchmarks exist?
- if not, use a simple architecture as benchmark

Assessing

Your Deep Learning Project III

Where are you at with respect to the following?

1 Splitting your data

- training set (80% — for optimizing parameters)
- validation set (10% — for hyperparameters)
- test set (10% — don't touch yet!)

2 Building and assessing architecture

- get above chance (simplifying problem, if necessary)
- do existing performance benchmarks exist?
- if not, use a simple architecture as benchmark

Assessing

Your Deep Learning Project III

Where are you at with respect to the following?

1 Splitting your data

- training set (80% — for optimizing parameters)
- validation set (10% — for hyperparameters)
- test set (10% — don't touch yet!)

2 Building and assessing architecture

- get above chance (simplifying problem, if necessary)
- do existing performance benchmarks exist?
- if not, use a simple architecture as benchmark

Outline

Review

DL Libraries

Introduction

Symbolic
Programming
Graphs
Neurons

Fitting Models

Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project
Improvement

Where We Are
Model Tuning

Up Next

- 1 Review Take-Home Exercise
- 2 Comparison of the Leading Deep Learning Libraries**
- 3 Introduction to TensorFlow
- 4 Fitting Models
- 5 TF 2.0
- 6 PyTorch
- 7 Deep Learning Project IV: Improving
- 8 Up Next: Advanced Topics

Leading DL Libraries

Review

DL Libraries

Introduction

Symbolic
Programming
Graphs
Neurons

Fitting Models

Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project
Improvement

Where We Are
Model Tuning

Up Next

	Caffe	Torch	MXNet	TensorFlow
<i>Language</i>	Python, Matlab	Lua, C	Python, R, C++ Julia, Matlab JavaScript, Go Scala, Perl	Python, C, C++ Java, Go, JS, Swift (<i>Haskell, Julia, R, Scala, Rust, C#</i>)
<i>Programming Style</i>	Symbolic	Imperative	Imperative	Imperative (<i>in 2.0</i>)
<i>Parallel GPUs: Data</i>	Yes	Yes	Yes	Yes
<i>Parallel GPUs: Model</i>		Yes	Yes	Yes
<i>Pre-Trained Models</i>	Model Zoo	Model Zoo	Model Zoo	github.com/tensorflow/ models
<i>High-Level APIs</i>		PyTorch	in-built	Keras
<i>Particular Strength</i>	CNNs	interactivity		production deployment

Leading DL Libraries

Review

DL Libraries

Introduction

Symbolic
Programming
Graphs
Neurons

Fitting Models

Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project
Improvement

Where We Are
Model Tuning

Up Next

PyTorch

TensorFlow

“NumPy”, optimized for GPUs

ported to Python from C++

dynamic auto-differentiation (autodiff)

static computational graph

debugging is easier

fast.ai API

Keras API

more widely adopted

TorchScript Just-In-Time compilation

TensorFlow Serving, .js, Lite, tf.data, tf.io

better for interactively building models

better for production deployments

Review

DL Libraries

Introduction

Symbolic
Programming
Graphs
Neurons

Fitting Models

Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project
Improvement

Where We Are
Model Tuning

Up Next

- 1 Review Take-Home Exercise
- 2 Comparison of the Leading Deep Learning Libraries
- 3 Introduction to TensorFlow**
Symbolic Programming
Programming TensorFlow Graphs
Neurons in TensorFlow
- 4 Fitting Models
- 5 TF 2.0
- 6 PyTorch
- 7 Deep Learning Project IV: Improving

Outline

Review

DL Libraries

Introduction

Symbolic
Programming

Graphs

Neurons

Fitting Models

Eight Data Points

Eight Million Points

Dense Nets

ConvNets

TF 2.0

PyTorch

Project

Improvement

Where We Are

Model Tuning

Up Next

- 1 Review Take-Home Exercise
- 2 Comparison of the Leading Deep Learning Libraries
- 3 Introduction to TensorFlow**
Symbolic Programming
Programming TensorFlow Graphs
Neurons in TensorFlow
- 4 Fitting Models
- 5 TF 2.0
- 6 PyTorch
- 7 Deep Learning Project IV: Improving

TensorFlow Graphs

Review

DL Libraries

Introduction

Symbolic
Programming

Graphs

Neurons

Fitting Models

Eight Data Points

Eight Million Points

Dense Nets

ConvNets

TF 2.0

PyTorch

Project
Improvement

Where We Are

Model Tuning

Up Next

- 1 build graph
- 2 initialize session
- 3 fetch and feed data

TensorFlow Graphs

Review

DL Libraries

Introduction

Symbolic
Programming

Graphs
Neurons

Fitting Models

Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project
Improvement

Where We Are
Model Tuning

Up Next

- 1 build graph
- 2 initialize session
- 3 fetch and feed data

TensorFlow Graphs

Review

DL Libraries

Introduction

Symbolic
Programming

Graphs

Neurons

Fitting Models

Eight Data Points

Eight Million Points

Dense Nets

ConvNets

TF 2.0

PyTorch

Project
Improvement

Where We Are

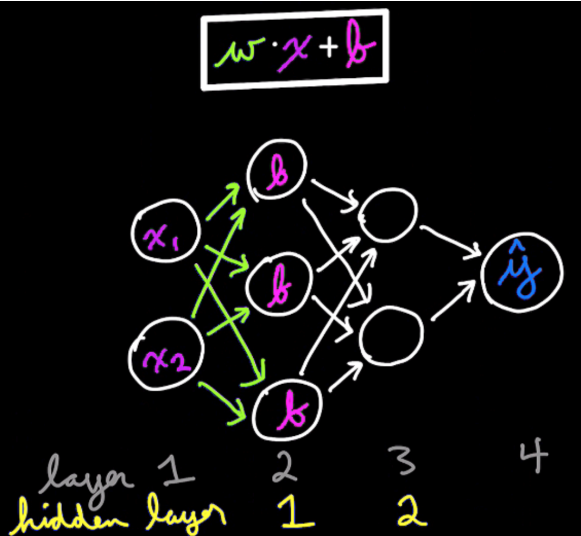
Model Tuning

Up Next

- 1 build graph
- 2 initialize session
- 3 fetch and feed data

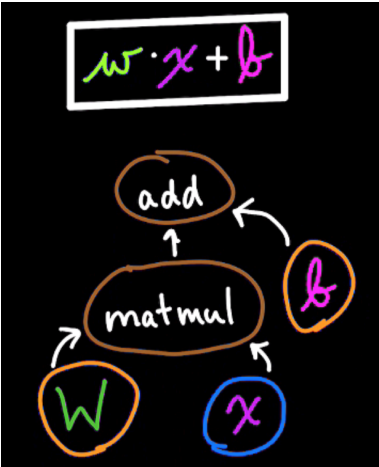
A Familiar Equation

$$w \cdot x + b$$



TensorFlow Graphs

- Review
- DL Libraries
- Introduction
 - Symbolic Programming
 - Graphs
 - Neurons
- Fitting Models
 - Eight Data Points
 - Eight Million Points
 - Dense Nets
 - ConvNets
- TF 2.0
- PyTorch
- Project Improvement
 - Where We Are
 - Model Tuning
- Up Next



Review

DL Libraries

Introduction

Symbolic
Programming

Graphs

Neurons

Fitting Models

Eight Data Points

Eight Million Points

Dense Nets

ConvNets

TF 2.0

PyTorch

Project

Improvement

Where We Are

Model Tuning

Up Next

- 1 Review Take-Home Exercise
- 2 Comparison of the Leading Deep Learning Libraries
- 3 Introduction to TensorFlow**
Symbolic Programming
Programming TensorFlow Graphs
Neurons in TensorFlow
- 4 Fitting Models
- 5 TF 2.0
- 6 PyTorch
- 7 Deep Learning Project IV: Improving

Review

DL Libraries

Introduction

Symbolic
Programming

Graphs

Neurons

Fitting Models

Eight Data Points

Eight Million Points

Dense Nets

ConvNets

TF 2.0

PyTorch

Project

Improvement

Where We Are

Model Tuning

Up Next

TensorFlow Graph

Programming

[*first TensorFlow graphs notebook*]

Review

DL Libraries

Introduction

Symbolic
Programming
Graphs
Neurons

Fitting Models

Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project
Improvement

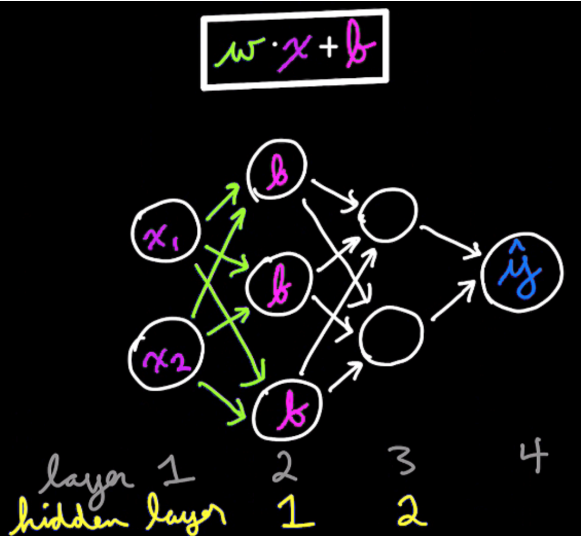
Where We Are
Model Tuning

Up Next

- 1 Review Take-Home Exercise
- 2 Comparison of the Leading Deep Learning Libraries
- 3 Introduction to TensorFlow**
Symbolic Programming
Programming TensorFlow Graphs
Neurons in TensorFlow
- 4 Fitting Models
- 5 TF 2.0
- 6 PyTorch
- 7 Deep Learning Project IV: Improving

A Familiar Equation

$$w \cdot x + b$$



Neurons in TensorFlow

Programming

Review

DL Libraries

Introduction

Symbolic
Programming

Graphs

Neurons

Fitting Models

Eight Data Points

Eight Million Points

Dense Nets

ConvNets

TF 2.0

PyTorch

Project
Improvement

Where We Are

Model Tuning

Up Next

[*first TensorFlow neurons* notebook]

Outline

Review

DL Libraries

Introduction

Symbolic
Programming
Graphs
Neurons

Fitting Models

Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project
Improvement

Where We Are
Model Tuning

Up Next

- 1 Review Take-Home Exercise
- 2 Comparison of the Leading Deep Learning Libraries
- 3 Introduction to TensorFlow
- 4 Fitting Models**
 - Eight Data Points
 - Eight Million Points
 - Dense Nets in TF 1.x
 - ConvNets in TF 1.x
- 5 TF 2.0
- 6 PyTorch

Outline

Review

DL Libraries

Introduction

Symbolic
Programming
Graphs
Neurons

Fitting Models

Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project
Improvement

Where We Are
Model Tuning

Up Next

- 1 Review Take-Home Exercise
- 2 Comparison of the Leading Deep Learning Libraries
- 3 Introduction to TensorFlow
- 4 Fitting Models**
 - Eight Data Points
 - Eight Million Points
 - Dense Nets in TF 1.x
 - ConvNets in TF 1.x
- 5 TF 2.0
- 6 PyTorch

Review

DL Libraries

Introduction

Symbolic
Programming
Graphs
Neurons

Fitting Models

Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project
Improvement

Where We Are
Model Tuning

Up Next

Fitting Eight Points

[*point by point intro to TensorFlow notebook*]

Review

DL Libraries

Introduction

Symbolic
Programming
Graphs
Neurons

Fitting Models

Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project
Improvement

Where We Are
Model Tuning

Up Next

Fitting Eight Points

with Tensors

[*tensor-fied intro to TensorFlow* notebook]

Outline

Review

DL Libraries

Introduction

Symbolic
Programming
Graphs
Neurons

Fitting Models

Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project
Improvement

Where We Are
Model Tuning

Up Next

- 1 Review Take-Home Exercise
- 2 Comparison of the Leading Deep Learning Libraries
- 3 Introduction to TensorFlow
- 4 Fitting Models**
 - Eight Data Points
 - Eight Million Points
 - Dense Nets in TF 1.x
 - ConvNets in TF 1.x
- 5 TF 2.0
- 6 PyTorch

Fitting Eight Million Points

Review

DL Libraries

Introduction

Symbolic
Programming
Graphs
Neurons

Fitting Models

Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project
Improvement

Where We Are
Model Tuning

Up Next

[*intro to TensorFlow times a million notebook*]

Outline

Review

DL Libraries

Introduction

Symbolic
Programming
Graphs
Neurons

Fitting Models

Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project
Improvement

Where We Are
Model Tuning

Up Next

- 1 Review Take-Home Exercise
- 2 Comparison of the Leading Deep Learning Libraries
- 3 Introduction to TensorFlow
- 4 Fitting Models**
 - Eight Data Points
 - Eight Million Points
 - Dense Nets in TF 1.x
 - ConvNets in TF 1.x
- 5 TF 2.0
- 6 PyTorch

Review

DL Libraries

Introduction

Symbolic
Programming
Graphs
Neurons

Fitting Models

Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project
Improvement

Where We Are
Model Tuning

Up Next

Dense Nets

[*intermediate net in TensorFlow notebook*] [*deep net in TensorFlow notebook*]

Review

DL Libraries

Introduction

Symbolic
Programming
Graphs
Neurons

Fitting Models

Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project
Improvement

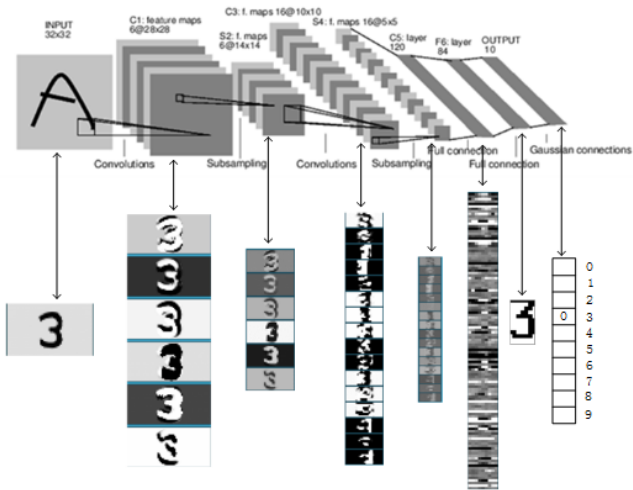
Where We Are
Model Tuning

Up Next

- 1 Review Take-Home Exercise
- 2 Comparison of the Leading Deep Learning Libraries
- 3 Introduction to TensorFlow
- 4 Fitting Models**
 - Eight Data Points
 - Eight Million Points
 - Dense Nets in TF 1.x
 - ConvNets in TF 1.x
- 5 TF 2.0
- 6 PyTorch

LeNet-5

LeCun et al. (1998)



Review

DL Libraries

Introduction

Symbolic
Programming
Graphs
Neurons

Fitting Models

Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project
Improvement

Where We Are
Model Tuning

Up Next

Review

DL Libraries

Introduction

Symbolic
Programming
Graphs
Neurons

Fitting Models

Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project
Improvement

Where We Are
Model Tuning

Up Next

LeNet-5

LeCun et al. (1998)

[*LeNet in TensorFlow* notebook]

Outline

Review

DL Libraries

Introduction

Symbolic
Programming
Graphs
Neurons

Fitting Models

Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project
Improvement

Where We Are
Model Tuning

Up Next

- 1 Review Take-Home Exercise
- 2 Comparison of the Leading Deep Learning Libraries
- 3 Introduction to TensorFlow
- 4 Fitting Models
- 5 TF 2.0**
- 6 PyTorch
- 7 Deep Learning Project IV: Improving
- 8 Up Next: Advanced Topics

Review

DL Libraries

Introduction

Symbolic
Programming
Graphs
Neurons

Fitting Models

Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project
Improvement

Where We Are
Model Tuning

Up Next

- use layers (Keras) by default
- autodiff Eager execution by default
- JIT compiler for optimization, especially across devices
- subclassing for unlimited flexibility, e.g.:
 - custom loss, optimizers, layers, training loops
 - repeating layers, blocks of layers
- data pipelines & processing with tf.data & tf.io
- portability with:
 - *TensorFlow Serving*
 - *TensorFlow Lite* for mobile/embedded
 - *TensorFlow.js*
- [to update from TF 1.x]

Review

DL Libraries

Introduction

Symbolic
Programming
Graphs
Neurons

Fitting Models

Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project
Improvement

Where We Are
Model Tuning

Up Next

- use layers (Keras) by default
- autodiff Eager execution by default
- JIT compiler for optimization, especially across devices
- subclassing for unlimited flexibility, e.g.:
 - custom loss, optimizers, layers, training loops
 - repeating layers, blocks of layers
- data pipelines & processing with tf.data & tf.io
- portability with:
 - *TensorFlow Serving*
 - *TensorFlow Lite* for mobile/embedded
 - *TensorFlow.js*
- [to update from TF 1.x]

Review

DL Libraries

Introduction

Symbolic
Programming
Graphs
Neurons

Fitting Models

Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project
Improvement

Where We Are
Model Tuning

Up Next

- use layers (Keras) by default
- autodiff Eager execution by default
- JIT compiler for optimization, especially across devices
- subclassing for unlimited flexibility, e.g.:
 - custom loss, optimizers, layers, training loops
 - repeating layers, blocks of layers
- data pipelines & processing with tf.data & tf.io
- portability with:
 - *TensorFlow Serving*
 - *TensorFlow Lite* for mobile/embedded
 - *TensorFlow.js*
- [to update from TF 1.x]

Review

DL Libraries

Introduction

Symbolic
Programming
Graphs
Neurons

Fitting Models

Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project
Improvement

Where We Are
Model Tuning

Up Next

- use layers (Keras) by default
- autodiff Eager execution by default
- JIT compiler for optimization, especially across devices
- subclassing for unlimited flexibility, e.g.:
 - custom loss, optimizers, layers, training loops
 - repeating layers, blocks of layers
- data pipelines & processing with `tf.data` & `tf.io`
- portability with:
 - *TensorFlow Serving*
 - *TensorFlow Lite* for mobile/embedded
 - *TensorFlow.js*
- [to update from TF 1.x]

Review

DL Libraries

Introduction

Symbolic
Programming
Graphs
Neurons

Fitting Models

Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project
Improvement

Where We Are
Model Tuning

Up Next

- use layers (Keras) by default
- autodiff Eager execution by default
- JIT compiler for optimization, especially across devices
- subclassing for unlimited flexibility, e.g.:
 - custom loss, optimizers, layers, training loops
 - repeating layers, blocks of layers
- data pipelines & processing with `tf.data` & `tf.io`
- portability with:
 - *TensorFlow Serving*
 - *TensorFlow Lite* for mobile/embedded
 - *TensorFlow.js*
- [to update from TF 1.x]

Review

DL Libraries

Introduction

Symbolic
Programming
Graphs
Neurons

Fitting Models

Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project
Improvement

Where We Are
Model Tuning

Up Next

- use layers (Keras) by default
- autodiff Eager execution by default
- JIT compiler for optimization, especially across devices
- subclassing for unlimited flexibility, e.g.:
 - custom loss, optimizers, layers, training loops
 - repeating layers, blocks of layers
- data pipelines & processing with `tf.data` & `tf.io`
- portability with:
 - *TensorFlow Serving*
 - *TensorFlow Lite* for mobile/embedded
 - *TensorFlow.js*
- [to update from TF 1.x]

Review

DL Libraries

Introduction

Symbolic
Programming
Graphs
Neurons

Fitting Models

Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project
Improvement

Where We Are
Model Tuning

Up Next

- use layers (Keras) by default
- autodiff Eager execution by default
- JIT compiler for optimization, especially across devices
- subclassing for unlimited flexibility, e.g.:
 - custom loss, optimizers, layers, training loops
 - repeating layers, blocks of layers
- data pipelines & processing with `tf.data` & `tf.io`
- portability with:
 - *TensorFlow Serving*
 - *TensorFlow Lite* for mobile/embedded
 - *TensorFlow.js*
- [to update from TF 1.x]

Review

DL Libraries

Introduction

Symbolic
Programming
Graphs
Neurons

Fitting Models

Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project
Improvement

Where We Are
Model Tuning

Up Next

- use layers (Keras) by default
- autodiff Eager execution by default
- JIT compiler for optimization, especially across devices
- subclassing for unlimited flexibility, e.g.:
 - custom loss, optimizers, layers, training loops
 - repeating layers, blocks of layers
- data pipelines & processing with `tf.data` & `tf.io`
- portability with:
 - *TensorFlow Serving*
 - *TensorFlow Lite* for mobile/embedded
 - *TensorFlow.js*
- [to update from TF 1.x]

Review

DL Libraries

Introduction

Symbolic
Programming
Graphs
Neurons

Fitting Models

Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project
Improvement

Where We Are
Model Tuning

Up Next

- use layers (Keras) by default
- autodiff Eager execution by default
- JIT compiler for optimization, especially across devices
- subclassing for unlimited flexibility, e.g.:
 - custom loss, optimizers, layers, training loops
 - repeating layers, blocks of layers
- data pipelines & processing with `tf.data` & `tf.io`
- portability with:
 - *TensorFlow Serving*
 - *TensorFlow Lite* for mobile/embedded
 - *TensorFlow.js*
- [to update from TF 1.x]

Review

DL Libraries

Introduction

Symbolic
Programming
Graphs
Neurons

Fitting Models

Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project
Improvement

Where We Are
Model Tuning

Up Next

- use layers (Keras) by default
- autodiff Eager execution by default
- JIT compiler for optimization, especially across devices
- subclassing for unlimited flexibility, e.g.:
 - custom loss, optimizers, layers, training loops
 - repeating layers, blocks of layers
- data pipelines & processing with `tf.data` & `tf.io`
- portability with:
 - *TensorFlow Serving*
 - *TensorFlow Lite* for mobile/embedded
 - *TensorFlow.js*
- [to update from TF 1.x]

Review

DL Libraries

Introduction

Symbolic
Programming
Graphs
Neurons

Fitting Models

Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project
Improvement

Where We Are
Model Tuning

Up Next

- use layers (Keras) by default
- autodiff Eager execution by default
- JIT compiler for optimization, especially across devices
- subclassing for unlimited flexibility, e.g.:
 - custom loss, optimizers, layers, training loops
 - repeating layers, blocks of layers
- data pipelines & processing with `tf.data` & `tf.io`
- portability with:
 - *TensorFlow Serving*
 - *TensorFlow Lite* for mobile/embedded
 - *TensorFlow.js*
- [to update from TF 1.x]

Review

DL Libraries

Introduction

Symbolic
Programming
Graphs
Neurons

Fitting Models

Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project
Improvement

Where We Are
Model Tuning

Up Next

- use layers (Keras) by default
- autodiff Eager execution by default
- JIT compiler for optimization, especially across devices
- subclassing for unlimited flexibility, e.g.:
 - custom loss, optimizers, layers, training loops
 - repeating layers, blocks of layers
- data pipelines & processing with `tf.data` & `tf.io`
- portability with:
 - *TensorFlow Serving*
 - *TensorFlow Lite* for mobile/embedded
 - *TensorFlow.js*
- [to update from TF 1.x]

Outline

Review

DL Libraries

Introduction

Symbolic
Programming
Graphs
Neurons

Fitting Models

Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project
Improvement

Where We Are
Model Tuning

Up Next

- 1 Review Take-Home Exercise
- 2 Comparison of the Leading Deep Learning Libraries
- 3 Introduction to TensorFlow
- 4 Fitting Models
- 5 TF 2.0
- 6 PyTorch**
- 7 Deep Learning Project IV: Improving
- 8 Up Next: Advanced Topics

Review

DL Libraries

Introduction

Symbolic
Programming
Graphs
Neurons

Fitting Models

Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project
Improvement

Where We Are
Model Tuning

Up Next

PyTorch

“NumPy”, optimized for GPUs

dynamic auto-differentiation (autodiff)

debugging is easier

fast.ai API

TorchScript Just-In-Time compilation

better for interactively building models

TensorFlow

ported to Python from C++

static computational graph

Keras API

more widely adopted

TensorFlow Serving, .js, Lite, tf.data, tf.io

better for production deployments

[shallow net in PyTorch]
[deep net in PyTorch]

Outline

Review

DL Libraries

Introduction

Symbolic
Programming
Graphs
Neurons

Fitting Models

Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

**Project
Improvement**

Where We Are
Model Tuning

Up Next

- 1 Review Take-Home Exercise
- 2 Comparison of the Leading Deep Learning Libraries
- 3 Introduction to TensorFlow
- 4 Fitting Models
- 5 TF 2.0
- 6 PyTorch
- 7 Deep Learning Project IV: Improving
Where We Are
Model Hyperparameter-Tuning Steps**

Outline

Review

DL Libraries

Introduction

Symbolic
Programming
Graphs
Neurons

Fitting Models

Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project
Improvement

Where We Are
Model Tuning

Up Next

- 1 Review Take-Home Exercise
- 2 Comparison of the Leading Deep Learning Libraries
- 3 Introduction to TensorFlow
- 4 Fitting Models
- 5 TF 2.0
- 6 PyTorch
- 7 Deep Learning Project IV: Improving
Where We Are
Model Hyperparameter-Tuning Steps

Review

DL Libraries

Introduction

Symbolic
Programming
Graphs
Neurons

Fitting Models

Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project
Improvement

Where We Are
Model Tuning

Up Next

Improving Your Deep Learning Project IV



Improving

Your Deep Learning Project IV

1 Splitting your data

- training set (80% — for optimizing parameters)
- validation set (10% — for hyperparameters)
- test set (10% — don't touch yet!)

2 Building and assessing architecture

- get above chance (simplifying problem, if necessary)
- do existing performance benchmarks exist?
- if not, use a simple architecture as benchmark

3 Improving performance & tuning hyperparameters...

Improving Your Deep Learning Project IV

- 1 Splitting your data
 - training set (80% — for optimizing parameters)
 - validation set (10% — for hyperparameters)
 - test set (10% — don't touch yet!)
- 2 Building and assessing architecture
 - get above chance (simplifying problem, if necessary)
 - do existing performance benchmarks exist?
 - if not, use a simple architecture as benchmark
- 3 Improving performance & tuning hyperparameters...

Improving

Your Deep Learning Project IV

1 Splitting your data

- training set (80% — for optimizing parameters)
- validation set (10% — for hyperparameters)
- test set (10% — don't touch yet!)

2 Building and assessing architecture

- get above chance (simplifying problem, if necessary)
- do existing performance benchmarks exist?
- if not, use a simple architecture as benchmark

3 Improving performance & tuning hyperparameters...

Improving Your Deep Learning Project IV

1 Splitting your data

- training set (80% — for optimizing parameters)
- validation set (10% — for hyperparameters)
- test set (10% — don't touch yet!)

2 Building and assessing architecture

- get above chance (simplifying problem, if necessary)
- do existing performance benchmarks exist?
- if not, use a simple architecture as benchmark

3 Improving performance & tuning hyperparameters...

Improving

Your Deep Learning Project IV

- 1 Splitting your data
 - training set (80% — for optimizing parameters)
 - validation set (10% — for hyperparameters)
 - test set (10% — don't touch yet!)
- 2 Building and assessing architecture
 - get above chance (simplifying problem, if necessary)
 - do existing performance benchmarks exist?
 - if not, use a simple architecture as benchmark
- 3 Improving performance & tuning hyperparameters...

Improving

Your Deep Learning Project IV

- 1 Splitting your data
 - training set (80% — for optimizing parameters)
 - validation set (10% — for hyperparameters)
 - test set (10% — don't touch yet!)
- 2 Building and assessing architecture
 - get above chance (simplifying problem, if necessary)
 - do existing performance benchmarks exist?
 - if not, use a simple architecture as benchmark
- 3 Improving performance & tuning hyperparameters...

Improving

Your Deep Learning Project IV

- 1 Splitting your data
 - training set (80% — for optimizing parameters)
 - validation set (10% — for hyperparameters)
 - test set (10% — don't touch yet!)
- 2 Building and assessing architecture
 - get above chance (simplifying problem, if necessary)
 - do existing performance benchmarks exist?
 - if not, use a simple architecture as benchmark
- 3 Improving performance & tuning hyperparameters...

Improving

Your Deep Learning Project IV

- 1 Splitting your data
 - training set (80% — for optimizing parameters)
 - validation set (10% — for hyperparameters)
 - test set (10% — don't touch yet!)
- 2 Building and assessing architecture
 - get above chance (simplifying problem, if necessary)
 - do existing performance benchmarks exist?
 - if not, use a simple architecture as benchmark
- 3 Improving performance & tuning hyperparameters...

Improving

Your Deep Learning Project IV

- 1 Splitting your data
 - training set (80% — for optimizing parameters)
 - validation set (10% — for hyperparameters)
 - test set (10% — don't touch yet!)
- 2 Building and assessing architecture
 - get above chance (simplifying problem, if necessary)
 - do existing performance benchmarks exist?
 - if not, use a simple architecture as benchmark
- 3 Improving performance & tuning hyperparameters...

Outline

Review

DL Libraries

Introduction

Symbolic
Programming
Graphs
Neurons

Fitting Models

Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project
Improvement

Where We Are
Model Tuning

Up Next

- 1 Review Take-Home Exercise
- 2 Comparison of the Leading Deep Learning Libraries
- 3 Introduction to TensorFlow
- 4 Fitting Models
- 5 TF 2.0
- 6 PyTorch
- 7 Deep Learning Project IV: Improving
Where We Are
Model Hyperparameter-Tuning Steps

Review

DL Libraries

Introduction

Symbolic
Programming
Graphs
Neurons

Fitting Models

Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project
Improvement

Where We Are
Model Tuning

Up Next

- 1 Parameter initialization
- 2 Cost function selection
- 3 Get above chance
- 4 Layers
 - Type
 - Number
 - Width
- 5 Avoid overfitting
 - Dropout
 - Data augmentation
 - Batch normalization
 - Early stopping / loading low-loss weights
- 6 Learning rate
- 7 Batch size

Review

DL Libraries

Introduction

Symbolic
Programming
Graphs
Neurons

Fitting Models

Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project
Improvement

Where We Are
Model Tuning

Up Next

- 1 Parameter initialization
- 2 Cost function selection
- 3 Get above chance
- 4 Layers
 - Type
 - Number
 - Width
- 5 Avoid overfitting
 - Dropout
 - Data augmentation
 - Batch normalization
 - Early stopping / loading low-loss weights
- 6 Learning rate
- 7 Batch size

Review

DL Libraries

Introduction

Symbolic
Programming
Graphs
Neurons

Fitting Models

Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project
Improvement

Where We Are
Model Tuning

Up Next

- 1 Parameter initialization
- 2 Cost function selection
- 3 Get above chance
- 4 Layers
 - Type
 - Number
 - Width
- 5 Avoid overfitting
 - Dropout
 - Data augmentation
 - Batch normalization
 - Early stopping / loading low-loss weights
- 6 Learning rate
- 7 Batch size

Review

DL Libraries

Introduction

Symbolic
Programming
Graphs
Neurons

Fitting Models

Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project
Improvement

Where We Are
Model Tuning

Up Next

- 1 Parameter initialization
- 2 Cost function selection
- 3 Get above chance
- 4 Layers
 - Type
 - Number
 - Width
- 5 Avoid overfitting
 - Dropout
 - Data augmentation
 - Batch normalization
 - Early stopping / loading low-loss weights
- 6 Learning rate
- 7 Batch size

Review

DL Libraries

Introduction

Symbolic
Programming
Graphs
Neurons

Fitting Models

Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project
Improvement

Where We Are
Model Tuning

Up Next

- 1 Parameter initialization
- 2 Cost function selection
- 3 Get above chance
- 4 Layers
 - Type
 - Number
 - Width
- 5 Avoid overfitting
 - Dropout
 - Data augmentation
 - Batch normalization
 - Early stopping / loading low-loss weights
- 6 Learning rate
- 7 Batch size

Review

DL Libraries

Introduction

Symbolic
Programming
Graphs
Neurons

Fitting Models

Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project
Improvement

Where We Are
Model Tuning

Up Next

- 1 Parameter initialization
- 2 Cost function selection
- 3 Get above chance
- 4 Layers
 - Type
 - Number
 - Width
- 5 Avoid overfitting
 - Dropout
 - Data augmentation
 - Batch normalization
 - Early stopping / loading low-loss weights
- 6 Learning rate
- 7 Batch size

Review

DL Libraries

Introduction

Symbolic
Programming
Graphs
Neurons

Fitting Models

Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project
Improvement

Where We Are
Model Tuning

Up Next

- 1 Parameter initialization
- 2 Cost function selection
- 3 Get above chance
- 4 Layers
 - Type
 - Number
 - Width
- 5 Avoid overfitting
 - Dropout
 - Data augmentation
 - Batch normalization
 - Early stopping / loading low-loss weights
- 6 Learning rate
- 7 Batch size

Review

DL Libraries

Introduction

Symbolic
Programming
Graphs
Neurons

Fitting Models

Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project
Improvement

Where We Are
Model Tuning

Up Next

- 1 Parameter initialization
- 2 Cost function selection
- 3 Get above chance
- 4 Layers
 - Type
 - Number
 - Width
- 5 Avoid overfitting
 - Dropout
 - Data augmentation
 - Batch normalization
 - Early stopping / loading low-loss weights
- 6 Learning rate
- 7 Batch size

Review

DL Libraries

Introduction

Symbolic
Programming
Graphs
Neurons

Fitting Models

Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project
Improvement

Where We Are
Model Tuning

Up Next

- 1 Parameter initialization
- 2 Cost function selection
- 3 Get above chance
- 4 Layers
 - Type
 - Number
 - Width
- 5 Avoid overfitting
 - Dropout
 - Data augmentation
 - Batch normalization
 - Early stopping / loading low-loss weights
- 6 Learning rate
- 7 Batch size

Review

DL Libraries

Introduction

Symbolic
Programming
Graphs
Neurons

Fitting Models

Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project
Improvement

Where We Are
Model Tuning

Up Next

- 1 Parameter initialization
- 2 Cost function selection
- 3 Get above chance
- 4 Layers
 - Type
 - Number
 - Width
- 5 Avoid overfitting
 - Dropout
 - Data augmentation
 - Batch normalization
 - Early stopping / loading low-loss weights
- 6 Learning rate
- 7 Batch size

Review

DL Libraries

Introduction

Symbolic
Programming
Graphs
Neurons

Fitting Models

Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project
Improvement

Where We Are
Model Tuning

Up Next

- 1 Parameter initialization
- 2 Cost function selection
- 3 Get above chance
- 4 Layers
 - Type
 - Number
 - Width
- 5 Avoid overfitting
 - Dropout
 - Data augmentation
 - Batch normalization
 - Early stopping / loading low-loss weights
- 6 Learning rate
- 7 Batch size

Review

DL Libraries

Introduction

Symbolic
Programming
Graphs
Neurons

Fitting Models

Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project
Improvement

Where We Are
Model Tuning

Up Next

- 1 Parameter initialization
- 2 Cost function selection
- 3 Get above chance
- 4 Layers
 - Type
 - Number
 - Width
- 5 Avoid overfitting
 - Dropout
 - Data augmentation
 - Batch normalization
 - Early stopping / loading low-loss weights
- 6 Learning rate
- 7 Batch size

Review

DL Libraries

Introduction

Symbolic
Programming
Graphs
Neurons

Fitting Models

Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project
Improvement

Where We Are
Model Tuning

Up Next

- 1 Parameter initialization
- 2 Cost function selection
- 3 Get above chance
- 4 Layers
 - Type
 - Number
 - Width
- 5 Avoid overfitting
 - Dropout
 - Data augmentation
 - Batch normalization
 - Early stopping / loading low-loss weights
- 6 Learning rate
- 7 Batch size

Review

DL Libraries

Introduction

Symbolic
Programming
Graphs
Neurons

Fitting Models

Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project
Improvement

Where We Are
Model Tuning

Up Next

- 1 Parameter initialization
- 2 Cost function selection
- 3 Get above chance
- 4 Layers
 - Type
 - Number
 - Width
- 5 Avoid overfitting
 - Dropout
 - Data augmentation
 - Batch normalization
 - Early stopping / loading low-loss weights
- 6 Learning rate
- 7 Batch size

Outline

Review

DL Libraries

Introduction

Symbolic
Programming
Graphs
Neurons

Fitting Models

Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project
Improvement

Where We Are
Model Tuning

Up Next

- 1 Review Take-Home Exercise
- 2 Comparison of the Leading Deep Learning Libraries
- 3 Introduction to TensorFlow
- 4 Fitting Models
- 5 TF 2.0
- 6 PyTorch
- 7 Deep Learning Project IV: Improving
- 8 Up Next: Advanced Topics

Review

DL Libraries

Introduction

- Symbolic Programming
- Graphs
- Neurons

Fitting Models

- Eight Data Points
- Eight Million Points
- Dense Nets
- ConvNets

TF 2.0

PyTorch

Project Improvement

- Where We Are
- Model Tuning

Up Next



Deep RL

Review

DL Libraries

Introduction

- Symbolic Programming
- Graphs
- Neurons

Fitting Models

- Eight Data Points
- Eight Million Points
- Dense Nets
- ConvNets

TF 2.0

PyTorch

Project Improvement

- Where We Are
- Model Tuning

Up Next

