Review

DL Librarie

Introduction Symbolic Programming

Fitting Models
Eight Data Points

Eight Million Point Dense Nets ConvNets

TF 2.0

PyTorch

Project Improvement Where We Are

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



Revie

DI Librar

Introductio

Symbolic Programmir Graphs

Fitting Mode Eight Data Points Eight Million Point Dense Nets

TF 2.0

PyTorch

Project Improvemer Where We Are Model Tuning

Up Nex

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



Revie

DI Librari

Introductio

Symbolic Programn Graphs Neurons

Eight Data Points
Eight Million Point
Dense Nets
ConvNets

TF 2.0

PyTorch

Project mprovemen Where We Are Model Tuning

Un Nex

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

Deep Learning Project IV: Improving

8 Up Next: Advanced Topics



Revie

DL Librari

Introduction Symbolic Programming

Symbolic Programn Graphs Neurons

Eight Data Points Eight Million Poin Dense Nets

TF 2.0

PyTorch

Project mprovemen Where We Are Model Tuning

In Nev

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
- Deep Learning Project IV: Improving
- 8 Up Next: Advanced Topics



Revie

DL Librari

Introduction Symbolic Programming

Symbolic Programmii Graphs Neurons

Eight Data Points Eight Million Point Dense Nets

TE 2.0

PyTorch

Project mprovemen Where We Are Model Tuning

In 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
- Deep Learning Project IV: Improving
- 8 Up Next: Advanced Topics



Revie

DL Librari

Introduction Symbolic Programming

Fitting Mode
Eight Data Points
Eight Million Point
Dense Nets

TF 2.0

PyTorch

Project mprovemer Where We Are Model Tuning

In Next

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

Deep Learning Project IV: Improving

8 Up Next: Advanced Topics



Revie

) Librari

Introduction
Symbolic
Programming
Graphs

Fitting Mode
Eight Data Points
Eight Million Point
Dense Nets
ConvNets

TF 2.0

PyTorch

Project mprovemen Where We Are Model Tuning

In 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
- Deep Learning Project IV: Improving
- 8 Up Next: Advanced Topics



Revie

L Librari

Introduction
Symbolic
Programming
Graphs

Fitting Mode
Eight Data Points
Eight Million Point
Dense Nets
ConvNets

TF 2.0

PyTorch

Project mprovemen Where We Are Model Tuning

Jp 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 Take-Home Exercise

2 Comparison of the Leading Deep Learning Libraries 3 Introduction to TensorFlow

Fitting Models

5 TF 2.0

6 PyTorch

Deep Learning Project IV: Improving 8 Up Next: Advanced Topics

Review

Review Take-Home Exercise

Review

Assessing Your Deep Learning Project III





Review

DL Librarı

Introduction Symbolic Programming Graphs

Fitting Model
Eight Data Points

TEON

PvTorc

Project Improvemen Where We Are Model Tuning

Up Next

Assessing Your Deep Learning Project III

- 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



Review

DL Librari

Introductio
Symbolic
Programming
Graphs

Fitting Model
Eight Data Points
Eight Million Points

TF 2.0

PyTorch

Project Improvemen Where We Are Model Tuning

Jp Next

Assessing Your Deep Learning Project III

- 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



Review

DL Librari

Introduction Symbolic Programming Graphs

Fitting Models
Eight Data Points
Eight Million Points

ConvNets

PyTorch

Project Improvemen Where We Are Model Tuning

Jp Next

Assessing Your Deep Learning Project III

- 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



Review

)L Librarie

Introduction Symbolic Programming Graphs

Fitting Models
Eight Data Points
Eight Million Points
Dense Nets

TF 2.0

PyTorch

Project Improvemen Where We Are

Up Nex

Assessing

Your Deep Learning Project III

- 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?
 - do existing performance benchmarks exist?
 - If not, use a simple architecture as benchmark



Review

DL Librari

Introductio
Symbolic
Programming
Graphs

Fitting Models
Eight Data Points
Eight Million Points
Dense Nets

TF 2.0

PyTorc

Project Improvemen Where We Are Model Tuning

Up Nex

Assessing Your Deep Learning Project III

- 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



Fitting Model
Eight Data Points
Eight Million Points
Dense Nets

TF 2.0

PyTorcl

Project Improvemen Where We Are Model Tuning

Up Next

Assessing Your Deep Learning Project III

- 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



Review

DL Librarie

Introduction Symbolic Programming Graphs

Fitting Models
Eight Data Points
Eight Million Points
Dense Nets

TF 2.0

PyTorch

Project Improvemen Where We Are Model Tuning

Up Nex

Assessing Your Deep Learning Project III

- 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



Review

DL Librarie

Introduction Symbolic Programming Graphs

Fitting Models
Eight Data Points
Eight Million Points
Dense Nets

TF 2.0

PyTorcl

Project Improvement Where We Are Model Tuning

Up Next

Assessing Your Deep Learning Project III

- 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



Review

Review Take-Home Exercise

Introduction Symbolic Programming

2 Comparison of the Leading Deep Learning Libraries

Fitting Mode Eight Data Point Eight Million Poi Dense Nets

3 Introduction to TensorFlow

TF 2.0

4 Fitting Models

Project Improveme

ΓF 2.0

Model Tuning

6 PyTorch

7 Deep Learning Project IV: Improving



ced Topics



Review

DL Libraries

Introduction

Symbolic Programming Graphs Neurons

Fitting Models

Eight Data Points

TF20

PyTorch

Project Improvemen Where We Are Model Tuning

Up Next

Leading DL Libraries

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



Review

DL Libraries

Introduction

Symbolic Programming Graphs

Fitting Models
Eight Data Points

Eight Data Points

Eight Million Point

PyTorch

Project Improvemen

Up Next

Leading DL Libraries

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		



Reviev

DI Librari

Introduction

Symbolic Programming Graphs

Fitting Model
Eight Data Points
Eight Million Points
Dense Nets

TF 2.0

PyTorc

Project mprovement Where We Are Model Tuning

Jp 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





Review

DL Librarie

Introduction Symbolic Programming Graphs

Fitting Model
Eight Data Points
Eight Million Point
Dense Nets

TF 2.0

PyTorcl

Project mprovemen Where We Are Model Tuning

In Next

- 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



Review

DI Librarie

Introduction

Symbolic Programming Graphs

Fitting Model

Eight Data Points

Dense N

.. 2.0

PyTorch

Project Improvemen Where We Are

ир мехт

TensorFlow Graphs

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



Review

DL Librarie

Introduction

Symbolic Programming Graphs

Fitting Model

Eight Data Points

Dense N

TF 2.

PyTorch

Project Improvemen Where We Are

Up Next

TensorFlow Graphs

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



Symbolic Programming

TensorFlow Graphs

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



A Familiar Equation

Review

DI Librarie

Introduction

Symbolic Programming Graphs

Fitting Model

Eight Data Points
Eight Million Point
Dense Nets

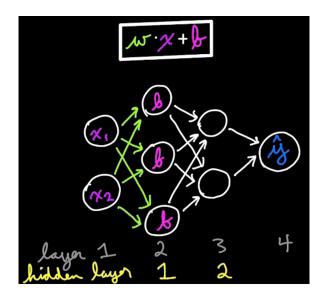
TF 2.0

PyTorch

Duningt

Where We Are

In Nevt





TensorFlow Graphs

Review

DI Libraria

Introductio

Symbolic Programming

Graphs

Fitting Model

Eight Million Point

ConvNe

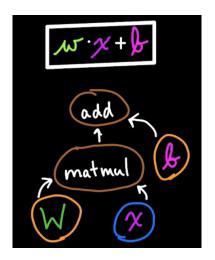
TF 2.0

PyTorch

Broject

Where We Are

Lin Novt





Review

DI Librario

Introductio
Symbolic
Programming
Graphs
Neurons

Fitting Model
Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorcl

Project mprovemen Where We Are Model Tuning

Jp Next

- 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





Graphs

TensorFlow Graph **Programming**

[first TensorFlow graphs notebook]



Reviev

DI Libraria

Introductio
Symbolic
Programming
Graphs
Neurons

Fitting Model
Eight Data Points
Eight Million Point
Dense Nets
ConvNets

TF 2.0

PyTorc

Project mprovemen Where We Are Model Tuning

Jp Next

- 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





A Familiar Equation

Review

DI Libraria

Introduction

Symbolic Programming

Graphs
Neurons

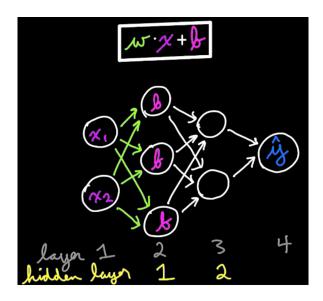
Fitting Models
Eight Data Points
Eight Million Points
Dense Nets

TF 2.0

PyTorch

Project Improvemen Where We Are

In Next





Review

DI Libraria

Introduction

Symbolic Programmi

Neurons

Eight Data Points

Dense Nets ConvNets

17 2.0

PyTorch

Project
Improvement
Where We Are

Up Next

Neurons in TensorFlow Programming

[first TensorFlow neurons notebook]



Review

DI Librarie

Introductio Symbolic Programming Graphs

Fitting Models

Eight Data Points

Dense Ne ConvNets

TF 2.0

PyTorch

Project Improvement Where We Are Model Tuning

Jp Next

- 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 Libraria

Introductio
Symbolic
Programming
Graphs

Fitting Mode
Eight Data Points
Eight Million Point
Dense Nets
ConvNets

TF 2.0

PyTorch

Project Improvemen Where We Are Model Tuning

Jp Next

- 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



D --- : ----

DI Librarie

Introduction

Symbolic Programming Graphs

Fitting Models
Eight Data Points
Eight Million Points
Dense Nets

TF 20

PyTorch

Project Improvement Where We Are Model Tuning

Up Next

Fitting Eight Points

[point by point intro to TensorFlow notebook]



Review

DL Librari

Introduction

Symbolic Programming Graphs

Fitting Models
Eight Data Points
Eight Million Points

TEO

PyTorch

Project Improvement Where We Are

Up Next

Fitting Eight Points with Tensors

[tensor-fied intro to TensorFlow notebook]



Outline

Review

DI Librarie

Introduction
Symbolic
Programming
Graphs
Neurons

Fitting Models
Eight Data Points
Eight Million Points
Dense Nets
ConvNets

17 2.0

PyTorch

Project Improvemen Where We Are Model Tuning

Jp Next

- 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 Librarie

Introduction

Symbolic Programming Graphs

Fitting Mode

Eight Million Points

Dense Nets

TF 2.0

Py Torch

Project Improvemen Where We Are

Up Next

Fitting Eight Million Points

[intro to TensorFlow times a million notebook]



Outline

Review

DL Libraria

Introductio
Symbolic
Programming
Graphs
Neurons

Eight Data Points
Eight Million Point
Dense Nets
ConvNets

TF 2.0

PyTorch

Project Improvemen Where We Are Model Tuning

של Next

- 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 Librarie

Introduction Symbolic

Programmin Graphs Neurons

Fitting Models
Eight Data Points

Dense Nets

TEO

PyTorch

Project Improvemen Where We Are

Up Nex

Dense Nets

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



Outline

Review

DL Libraria

Introductio
Symbolic
Programming
Graphs

Fitting Mode
Eight Data Points
Eight Million Point
Dense Nets
ConvNets

TF 2.0

PyTorcl

Project Improvemen Where We Are Model Tuning

Jp Next

- 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

DI Librarie

Introduction

Symbolic Programming Graphs

Fitting Models
Eight Data Points
Eight Million Points

ConvNets

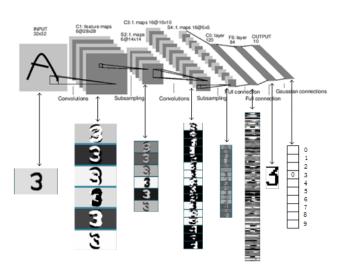
PyTorc

Project Improvement Where We Are

Up Next

LeNet-5

LeCun et al. (1998)





Review

DI Librarie

Introduction

Programmir Graphs

Fitting Models

Eight Data Points
Eight Million Points

ConvNets

D. .T.

ry loici

Improvemen
Where We Are

Up Next

LeNet-5

LeCun et al. (1998)

[LeNet in TensorFlow notebook]





Units 7 and 8

Outline

TF 20

5 TF 2.0

DL Librarie

Introduction

Symbolic Programming Graphs

Fitting Models Eight Data Points Eight Million Points Dense Nets

TF 2.0

PyTorch

Project Improvement Where We Are Model Tuning

Jp 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]



DL Librario

Introduction Symbolic

Symbolic Programming Graphs Neurons

Fitting Models Eight Data Points Eight Million Points Dense Nets

TF 2.0

PyTorch

Project Improvement Where We Are Model Tuning

Jp 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]



DI Libraria

Introduction Symbolic Regramming

Symbolic Programming Graphs Neurons

Eight Data Points Eight Million Points Dense Nets

TF 2.0

PyTorch

Project Improvement Where We Are

- 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]



DL Librario

Introduction Symbolic Programming Graphs

Fitting Models
Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project Improvement Where We Are Model Tuning

Jp 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]



DL Librarie

Introduction Symbolic Programming Graphs

Fitting Models
Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project Improvement Where We Are Model Tuning

- 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]



Reviev

DL Librarie

Introduction Symbolic Programming Graphs

Fitting Models
Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project Improvement Where We Are Model Tuning

- 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]



Reviev

DL Librarie

Introduction Symbolic Programming Graphs

Fitting Models
Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project Improvement Where We Are Model Tuning

- 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]



DL Librarie

Introduction Symbolic Programming Graphs

Fitting Models
Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project Improvement Where We Are Model Tuning · 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]



L Librarie

Introduction Symbolic Programming Graphs

Fitting Models
Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project Improvement Where We Are Model Tuning

- · 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]



L Librarie

Introduction Symbolic Programming Graphs

Fitting Models
Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project Improvement Where We Are Model Tuning

up Nex

- 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]



Reviev

DL Librarie

Introduction Symbolic Programming Graphs

Fitting Models
Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project Improvement Where We Are Model Tuning

- 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]



Reviev

DL Librarie

Introduction Symbolic Programming Graphs

Fitting Models
Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project Improvement Where We Are Model Tuning

- 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

PyTorch

6 PyTorch



DI Librari

DL LIBIAN

Symbolic Programming Graphs

Fitting Models
Eight Data Points
Eight Million Points
Dense Nets

TF 2.

PyTorch

Project Improvemen Where We Are Model Tuning

up Nex

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

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



Outline

Revie

DL Librari

Introduction Symbolic Programming

Fitting Mode
Eight Data Points
Eight Million Point
Dense Nets
ConvNets

TF 2.0

PyTorch

Project Improvement Where We Are Model Tuning

Un Next

- 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
- Deep Learning Project IV: Improving Where We Are Model Hyperparameter-Tuning Steps



Outline

Revie

DL Librari

Introduction Symbolic Programming

Fitting Model
Eight Data Points
Eight Million Point
Dense Nets
ConvNets

TF 2.0

PyTorch

Project Improvemen Where We Are ^{Model} Tuning

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



Review

A Libraria

Introduction
Symbolic
Programming
Graphs

Fitting Models
Eight Data Points
Eight Million Points

TF 2.0

PyTorc

Project Improvemen Where We Are

Un Nex





Review

DI Librarie

Introduction Symbolic

Programmin Graphs Neurons

Fitting Model

ConvNets

Py forci

Project
Improvemen
Where We Are

Up Next

Improving Your Deep Learning Project IV

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
- Improving performance & tuning hyperparameters...



Reviev

DL Librario

Introduction Symbolic Programming Graphs

Fitting Models

Eight Data Points

TEOO

Py Torch

Project Improvement Where We Are Model Tuning

Up Next

- 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...



Review

DL Librario

Introduction Symbolic Programming Graphs

Fitting Models

Eight Data Points

ConvNets

ByTorol

PyTorci

Project Improvement Where We Are Model Tuning

Up Next

- 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
- Improving performance & tuning hyperparameters...



Review

DL Librarie

Introduction Symbolic Programming Graphs

Fitting Model
Eight Data Points

TF 2.0

PyTorch

Project Improvement Where We Are

Up Next

- 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
- Improving performance & tuning hyperparameters...



Review

DL Librarie

Introductio Symbolic Programming Graphs

Fitting Model

TF 2.0

PyTorch

Project
Improvement
Where We Are

Up Next

- 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...



Review

DL Librarie

Introductio
Symbolic
Programming
Graphs

Fitting Model

TF 2.0

PyTorch

Project Improvement Where We Are Model Tuning

Up Next

- 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...



Review

DL Librarie

Introductio
Symbolic
Programming
Graphs

Fitting Model:
Eight Data Points
Eight Million Points

TF 2.0

PyTorcl

Project Improvement Where We Are ^{Model Tuning}

Up Next

- 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...



Review

DL Librarie

Introduction Symbolic Programming Graphs

Fitting Model: Eight Data Points Eight Million Points

TF 2.0

PyTorch

Project Improvement Where We Are Model Tuning

Up Next

- 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...



Review

DL Librarie

Introductio
Symbolic
Programming
Graphs

Fitting Models
Eight Data Points
Eight Million Points

TF 2.0

PyTorch

Project Improvement Where We Are Model Tuning

Jp Next

- 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

Revie

DI Librari

Introductio
Symbolic
Programming

Fitting Model
Eight Data Points
Eight Million Point
Dense Nets
ConvNets

TF 2.0

PyTorch

Project Improvemen 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

Deep Learning Project IV: Improving Where We Are Model Hyperparameter-Tuning Steps



DI Librarie

Introduction Symbolic

Programming Graphs Neurons

Fitting Models
Eight Data Points
Eight Million Points
Dense Nets

TF 2.0

PyTorch

Project
Improvement
Where We Are
Model Tuning

Jp Next

Parameter initialization

- 2 Cost function selection
- 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
- Batch size



DI Librarie

Introductio Symbolic Programming Graphs

Fitting Models
Eight Data Points
Eight Million Points
Dense Nets

TF 2.0

PyTorch

Project Improvement Where We Are Model Tuning

лр ілехт

- Parameter initialization
- 2 Cost function selection
- 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
- Batch size



DI Librarie

Introduction Symbolic Programming Graphs

Fitting Models
Eight Data Points
Eight Million Points
Dense Nets

TF 2.0

PyTorcl

Project Improvement Where We Are Model Tuning

ль ілехі

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



DI Librarie

Introduction Symbolic Programming Graphs

Fitting Models
Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project Improvement Where We Are Model Tuning

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



DI Librarie

Introduction Symbolic Programming Graphs

Fitting Models
Eight Data Points
Eight Million Points
Dense Nets
ConvNets

PyTorch

Project Improvement Where We Are Model Tuning

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



DI Librarie

Introduction Symbolic Programming Graphs

Fitting Models
Eight Data Points
Eight Million Points
Dense Nets

17 2.0

PyTorch

Project Improvement Where We Are Model Tuning

- 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
- Batch size



DI Librarie

Introduction Symbolic Programming Graphs

Fitting Models
Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project Improvement Where We Are Model Tuning

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



DI Librarie

Introduction Symbolic Programming Graphs

Fitting Models
Eight Data Points
Eight Million Points
Dense Nets

TF 2.0

PyTorch

Project Improvement Where We Are Model Tuning

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



DI Librarie

Introduction Symbolic Programming Graphs

Fitting Models
Eight Data Points
Eight Million Points
Dense Nets

17 2.0

PyTorch

Project Improvement Where We Are Model Tuning

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



DI Librarie

Introduction Symbolic Programming Graphs

Fitting Models
Eight Data Points
Eight Million Points
Dense Nets

TF 2.0

PyTorch

Project Improvement Where We Are Model Tuning

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



DL Librarie

Introduction Symbolic Programming Graphs

Fitting Models
Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project Improvement Where We Are Model Tuning

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



Reviev

DL Librarie

Introduction Symbolic Programming Graphs

Fitting Models
Eight Data Points
Eight Million Points
Dense Nets
ConvNets

1F 2.0

PyTorch

Project Improvement Where We Are Model Tuning

ль мехі

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



DL Librarie

Introduction Symbolic Programming Graphs

Fitting Models
Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project Improvement Where We Are Model Tuning

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



DI Librarie

Introductio Symbolic Programming Graphs

Fitting Models
Eight Data Points
Eight Million Points
Dense Nets
ConvNets

TF 2.0

PyTorch

Project Improvemen Where We Are Model Tuning

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





Units 7 and 8

Outline

Up Next

8 Up Next: Advanced Topics

Units 7 and 8 — Tensorflow

GANs

Review

DI Librari

Introduction

Programr Graphs

Fitting Mode

Eight Data Points
Eight Million Points
Dense Nets

TF 2.0

PyTorc

D.

Improvement
Where We Are
Model Tuning

Up Next





Units 7 and 8 — Tensorflow

Up Next

Deep RL



