

TensorFlow Tutorial

Open Source Software

Wesley Turner

Rensselaer Polytechnic Institute

Department of Computer Science

Some material adapted from Olivier Poulin

Licensed under: CC-BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0/>

Reading Material

- **TensorFlow Guide - (Reference Only)** – <https://www.tensorflow.org/guide>
 - * Read TensorFlow Basics through "Training Loops"
 - * Keras through "Training and evaluation ..."

Introduction

What is TensorFlow?

TensorFlow™ is:

- Open source
- High performance numerical computation
- Deploys on CPUs, GPUs, TPU
- Desktops, clusters, mobile and edge devices
- Originally developed by Google Brain team within Google's AI organization

TensorFlow™ has:

- Strong support for machine learning
- Strong support for deep learning
- Flexible numerical computation core

Widely Used

Companies:



Companies (continued):



Companies (continued):



By the Numbers

Github page: <https://github.com/tensorflow>

tensorflow

<http://www.tensorflow.org> github-admin@tensorflow.org

Repositories 52 People 160 Projects 0

Pinned repositories

tensorflow
Computation using data flow graphs for scalable machine learning
C++ ★ 107k 📄 66.3k

models
Models and examples built with TensorFlow
Jupyter Notebook ★ 39.8k 📄 23.6k

Search repositories... Type: All Language: All

tensorflow
Computation using data flow graphs for scalable machine learning
python machine-learning deep-neural-networks deep-learning

Top languages
Python TypeScript Jupyter Notebook C++ HTML

Main repository: <https://github.com/tensorflow/tensorflow>

- 53,075 commits (now 132,807)
- 27 branches (now 68)
- 1932 contributors (now 3155)
- 79 releases (now 169)
- 2021 Issues (now 2118 - closed 33,110)
- Permissively licensed under Apache-2.0

tensorflow/tensorflow is licensed under the Apache License 2.0 A permissive license whose main conditions require preservation of copyright and license notices. Contributors provide an express grant of patent rights. Licensed works, modifications, and larger works may be distributed under different terms and without source code.	Permissions <ul style="list-style-type: none">✓ Commercial use✓ Modification✓ Distribution✓ Patent use✓ Private use	Limitations <ul style="list-style-type: none">✗ Trademark use✗ Liability✗ Warranty	Conditions <ul style="list-style-type: none">① License and copyright notice① State changes
This is not legal advice. Learn more about repository licenses.			

Active community

- Community overview: <https://www.tensorflow.org/community/>

TensorFlow™ Install Develop API 1.10 Versions Community Ecosystem Search GITHUB

Community

COMMUNITY ABOUT

Community

Roadmap

Contributing to TensorFlow

Mailing Lists

User Groups

Writing TensorFlow Documentation

TensorFlow Style Guide

Defining and Running Benchmarks

Community

☆☆☆☆☆

Welcome to the TensorFlow community! This page explains where to get help, and different ways to be part of the community. We are committed to fostering an open and welcoming environment, and request that you review our [code of conduct](#).

Get Help

Technical Questions

To ask or answer technical questions about TensorFlow, use [Stack Overflow](#). For example, ask or search about a particular error message you encountered during installation.

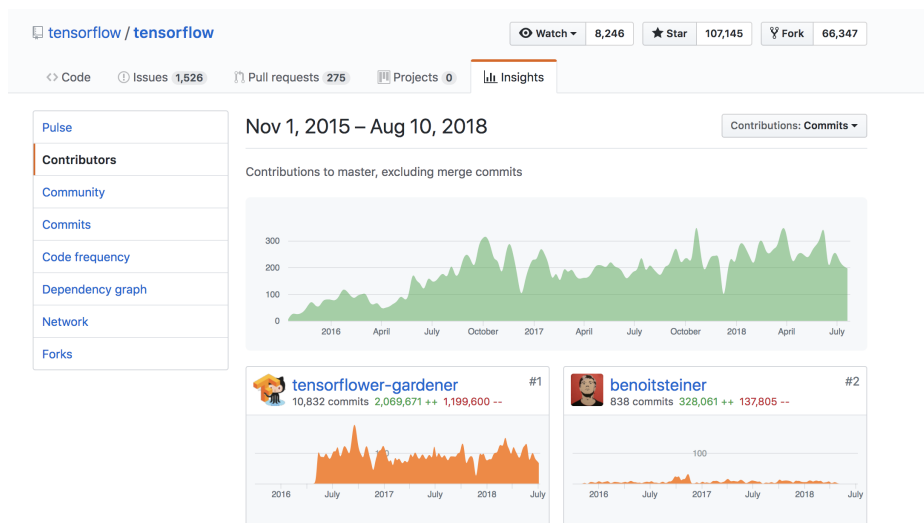
Bugs and Feature Requests

To report bugs or make feature requests, file an issue on GitHub. Please choose the appropriate repository for the project. Major repositories include:

Contents

- Get Help
- Technical Questions
- Bugs and Feature Requests
- Security
- Stay Informed
- Announcements
- Mailing List
- Development Roadmap
- Social Media
- Blog
- YouTube
- Community Support
- Mailing Lists
- User Groups
- Contributing To TensorFlow

- Insights at: <https://github.com/tensorflow/tensorflow/graphs/contributors>



- Blog: <https://blog.tensorflow.org/>

TensorFlow Blog

Search the Blog

Return to TensorFlow Home

All TensorFlow Core TensorFlow.js TensorFlow Lite TFX Community

Featured

TensorFlow Quantum turns one year old

TensorFlow Core

How-to Write a Python Fuzzer for TensorFlow

April 01, 2021 — Posted by Laura Pak

Tags

TensorFlow Core →

TensorFlow.js →

- Issue Tracker: <https://github.com/tensorflow/tensorflow/issues>

tensorflow / tensorflow

Watch 8,246 Star 107,146 Fork 66,347

<> Code Issues 1,526 Pull requests 275 Projects 0 Insights

Want to submit an issue to tensorflow/tensorflow?

If you have a bug or an idea, read the [contributing guidelines](#) before opening an issue.

Dismiss

Filters is:issue is:open Labels Milestones New issue

1,526 Open 11,460 Closed Author Labels Projects Milestones Assignee Sort

OutOfRangeError: read less bytes than requested
#21544 opened 3 hours ago by chaowu2009

ConditionalBijectors not Chainable due to Bijector_Mapping not supporting deep dicts
#21543 opened 4 hours ago by hartikainen

fatal: not a git repository:
'E:/tensorflow/tensorflow/contrib/cmake/build/nsync/src/nsync/.git'
#21542 opened 12 hours ago by cvJle

The nn_test unit test testGradient.L2LossTest fails on AVX512 builds
#21540 opened 12 hours ago by markdown

- Questions: <https://stackoverflow.com/questions/tagged/tensorflow>

- User Models: <https://github.com/tensorflow/models/tree/master/community>



TensorFlow Community Models

This repository provides a curated list of the GitHub repositories with machine learning models and implementations powered by TensorFlow 2.

Note: Contributing companies or individuals are responsible for maintaining their repositories.

Computer Vision

Image Recognition

Model	Paper	Features	Maintainer
DenseNet 169	Densely Connected Convolutional Networks	• FP32 Inference	Intel
Inception V3	Rethinking the Inception Architecture for Computer Vision	• Int8 Inference • FP32 Inference	Intel
Inception V4	Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning	• Int8 Inference • FP32 Inference	Intel
MobileNet V1	MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications	• Int8 Inference • FP32 Inference	Intel
ResNet 101	Deep Residual Learning for Image Recognition	• Int8 Inference • FP32 Inference	Intel

Others:

- Twitter
- YouTube
- Release Notes

The Basics

Much of this section courtesy of Olivier Poulin, one of our previous mentors.

Multiple Installations

- Virtualenv
- "native" pip
- Docker
- Source

For this class, we will use the Docker installation:

```
$ docker run -it -p 8888:8888 -e "DISPLAY=host.docker.internal:0" \
tensorflow/tensorflow:latest
Unable to find image 'tensorflow/tensorflow:latest-devel' locally
latest-devel: Pulling from tensorflow/tensorflow
8ee29e426c26: Pull complete
...
9c2312dbc5d7: Pull complete
Digest: sha256:40844012558fe881ec58faf1627fd4bb3f64fe9d46a2fd8af70f139244cfb538
Status: Downloaded newer image for tensorflow/tensorflow:latest
```

```
-----
--  /-----
-- /  - - \_ -- \_ -- /  -- \_ -- /  /  -- /  -- \_ | / / /
- /  /  _ / / / / ( _ ) / / / /  - _ /  - / / / / _ | / / /
/_/  \_ / / / / / / _ / \_ / / /  /_ /  /_ / \_ / _ / | _ /
```

Docker:

- Runs a TensorFlow Container
 - Bindings to Python
- Maps port 8888 on the Container to port 8888 outside the container
 - Allows you to run Jupyter Notebooks
- Starts up an interactive session
- Allows us to bring up windows on our native OS
 - Requires an X11 Server, but you should all have that from earlier

Validate

From the Docker container:

```
# python
Python 3.8.10 (default, Nov 26 2021, 20:14:08)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.

>>> import tensorflow as tf
>>> a = tf.constant(3)
>>> a
<tf.Tensor: shape=(), dtype=int32, numpy=3>
>>> b = tf.constant(5)
```

```
>>> c = a + b
>>> c
<tf.Tensor: shape=(), dtype=int32, numpy=8>
>>> print(c)
tf.Tensor(8, shape=(), dtype=int32)
```

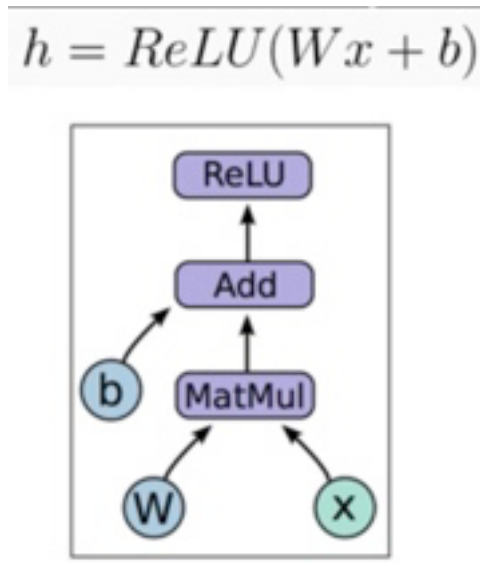
What does TensorFlow do?

- Similar to Numpy, for n-dimensional arrays, but TensorFlow simplifies creation of tensor methods and computes derivatives.

Numpy	TensorFlow
a=np.zeros((2,2)); b=np.ones((2,2))	a=tf.zeros((2,2)); b=tf.ones((2,2))
np.sum(b,axis=1)	tf.reduce_sum(b,axis=[1])
a.shape	a.get_shape()
np.reshape(a, (1,4))	tf.reshape(a, (1,4))
b * 5 + 1	b * 5 + 1
np.dot(a, b)	tf.matmul(a, b)
a[0,0], a[:,0], a[0,:]	a[0,0], a[:,0], a[0,:]

Base usage involves making execution graph

- TensorFlow uses a computation graph that has no numerical value until it's evaluated.
- Program structure has two phases: Construction phase and Execution phase.
- Construction phase assembles the computation graph.
- Execution phase runs the session object to execute all the operations in the graph.



Base usage involves making execution graph

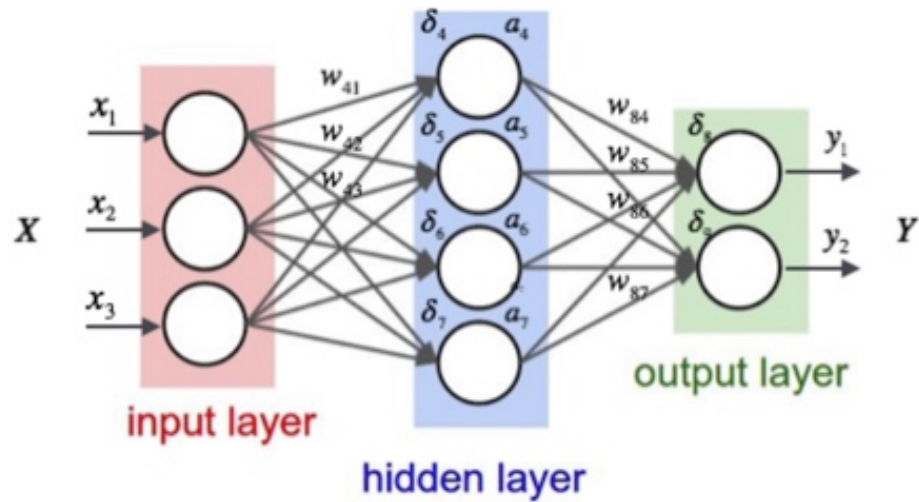
- Okay forget that ...
- This was Tensorflow 1.0
- In Tensorflow 2.0, the graph still exists but you can ignore it ...
 - Unless of course, you want to develop your algorithm in one place, and then run it later
 - Tensorflow allows you to save out the graph in a language independent file that can be migrated to another machine

What is Deep Learning?

- Deep learning is a machine learning method.
- More complex but has broader applications than classic task-specific algorithms.
- It bases the construction of its models on networks observed in biological nervous systems.
- Train Artificial Neural Networks to transform an input into a desired output.

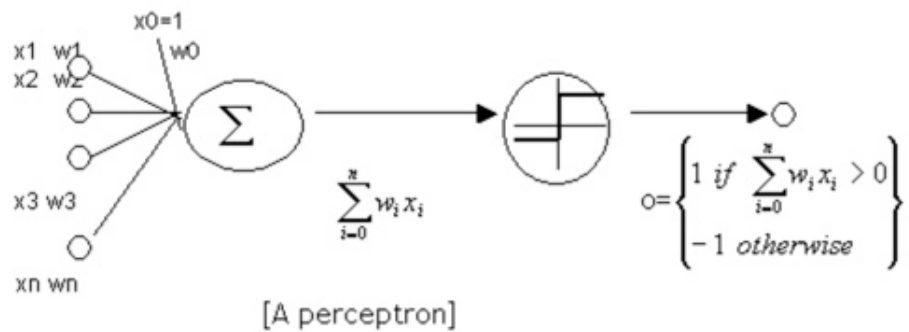
Neural Network

- A collection of units or nodes (artificial neurons, hence neural network)
- Connected in layers to one another. Each node sends data to other nodes
- Train the “weights” and “biases” on each neuron to slowly inch the network towards a specific functionality.



Simplest artificial neural network (ANN): Perceptron

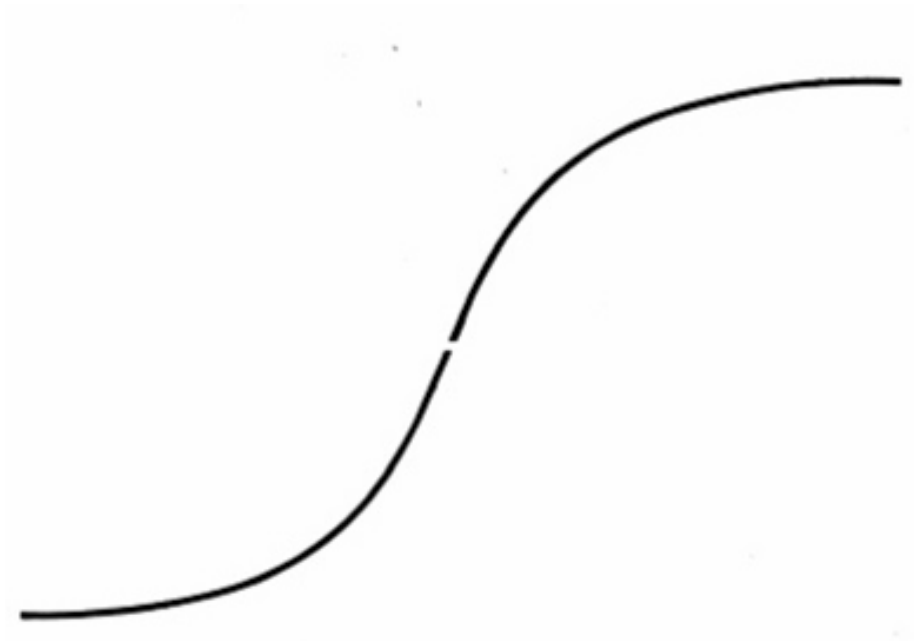
- Only binary inputs/outputs
- Binary output means the signals between neurons can only be binary as well
- Something either is, or isn't
- Limited in its functionality



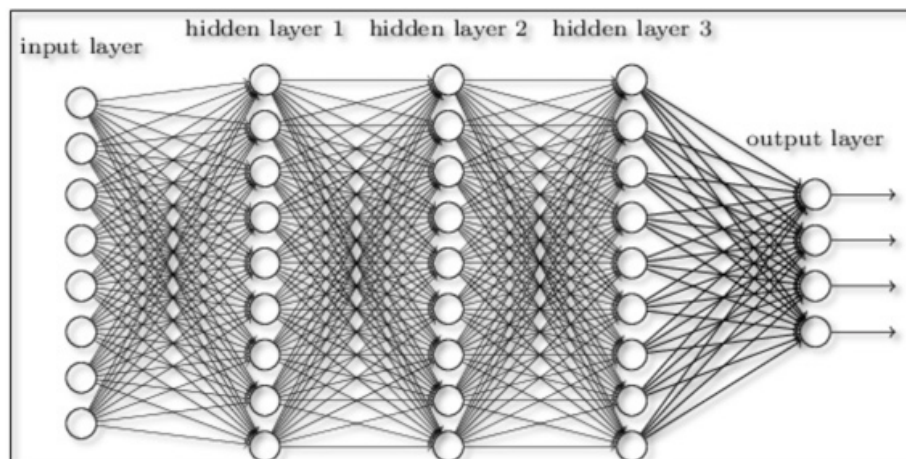
Sigmoid Neurons

- Inputs/outputs are any values between 0 and 1
- Gives us much more nuanced outputs
- Can be used for % matches

$$f(t) = \frac{1}{1+e^{-t}}$$

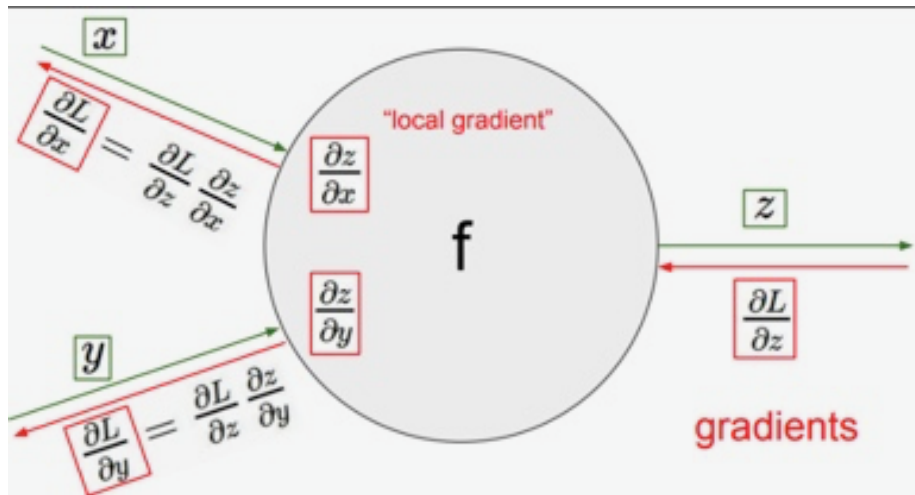


Deep Learning uses multiple layered networks



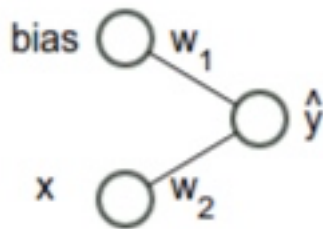
How to train your dragon (network)

- Compare the output with training data
- Get a vector of all the errors and compute the downward slope of the error curve (derivative)
- Change the weights based on this “Gradient Descent”
- Complicated in huge networks



A simple linear example

- Simplifies everything!
- Example: A simple linear regression!
- Linear function $y = ab+c$
- How do we train a simple network to mimic a linear function?
- Where bias = 1. This function becomes $y = xw_2 + w_1$



Get set up

For OSX, you will need to install a few packages on the host first to get the plots at the end to work:

```
brew install socat
socat TCP-LISTEN:6000,reuseaddr,fork UNIX-CLIENT:\"$DISPLAY\" &
brew install --cask xquartz
open -a Xquartz
```

Then set Allow connections from network clients in the pop up if asked.

Run a docker container and update it

```

docker run -it -p 8888:8888 -e "DISPLAY=host.docker.internal:0 \
    tensorflow/tensorflow:latest
apt-get update
apt-get install -y python-tk xterm x11-apps qt5-default
xeyes & # Just a test to make sure our display is working
pip install matplotlib PyQt5

or use your own python installation

pip install tensorflow matplotlib PyQt5

```

Run a simple example

Imports:

```

import tensorflow as tf
from tensorflow.keras import Model
import matplotlib.pyplot as plt

```

Let's define some utilities:

```

def make_noisy_data(m=0.1, b=0.3, n=100):
    x = tf.random.uniform(shape=(n,))
    noise = tf.random.normal(shape=(len(x),), stddev=0.01)
    y = m * x + b + noise
    return x, y

def predict(x):
    y = m * x + b
    return y

def squared_error(y_pred, y_true):
    return tf.reduce_mean(tf.square(y_pred - y_true))

```

Set up the data:

```

x_train, y_train = make_noisy_data()
plt.plot(x_train, y_train, 'b.')
plt.show()

m = tf.Variable(0.)
b = tf.Variable(0.)

loss = squared_error(predict(x_train), y_train)
loss_vec = []
print("Starting loss {:.6f}".format(loss.numpy()))

```

Training parameters:

```

learning_rate = 0.05
steps = 200

Execute the gradient descent:

for i in range(steps):

    with tf.GradientTape() as tape:
        predictions = predict(x_train)
        loss = squared_error(predictions, y_train)

    loss_vec.append(loss)
    gradients = tape.gradient(loss, [m, b])

    m.assign_sub(gradients[0] * learning_rate)
    b.assign_sub(gradients[1] * learning_rate)

    if i % 20 == 0:
        print("Step {:d}, Loss {:.6f}".format(i, loss.numpy()))

Report:

print("Solution: y = {:.6f} * x + {:.6f}".format(m.numpy(), b.numpy()))
plt.plot(list(range(steps)), loss_vec)
plt.show()

plt.plot(x_train, y_train, 'b.')
plt.plot(x_train, predict(x_train))
plt.show()

```

Using TensorFlow

Same Example in Keras

```

# MIT License
#
# This example is partially derived from the fashion example in Tensorflow.
# New code is copywritten by Wesley Turner (c) 2022
#
# The original code is copywritten below.
#
# Copyright (c) 2017 François Chollet
#

# TensorFlow and tf.keras
import tensorflow as tf

```

```

# Helper libraries
import numpy as np
import matplotlib.pyplot as plt

def make_noisy_data(m=0.1, b=0.3, n=100):
    x = 4*tf.random.uniform(shape=(n,1,1)) - 2
    noise = tf.random.normal(shape=(len(x),1,1), stddev=0.01)
    y = m * x + b + noise
    return x, y

def predict(m, b, x):
    y = m * x + b
    return y

def squared_error(y_pred, y_true):
    return tf.reduce_mean(tf.square(y_pred - y_true))

# Loading data
m = 0.1
b = 0.3
(train_x, train_y) = make_noisy_data(m=m, b=b, n=20000)
(test_x, test_y) = make_noisy_data(m=m, b=b, n=10000)

# 1. Create a model with a single neuron in 1 dense layer and 'relu' activation
# 2. Your model should use 'RMSprop' optimization, and mean squared error for
# both the loss function and the metric
# 3. Train your model on the 'train_x' and 'train_y' data from above.
model = tf.keras.Sequential([
    tf.keras.layers.Dense(1, activation='relu'),
])

model.compile(optimizer=tf.keras.optimizers.SGD(learning_rate=0.2),
              loss=tf.keras.losses.mean_squared_error,
              metrics=tf.keras.metrics.mean_squared_error)

model.fit(train_x, train_y, epochs=10)

# 4. Then evaluate the model against the test_x and test_y
test_loss, test_acc = model.evaluate(test_x, test_y, verbose=2)

#5 Finally, use your model to predict the correct output from the 'test_x'
predictions = model.predict(test_x)

# Calculate the actual values
actual = predict(m, b, test_x)

# Report

```

```

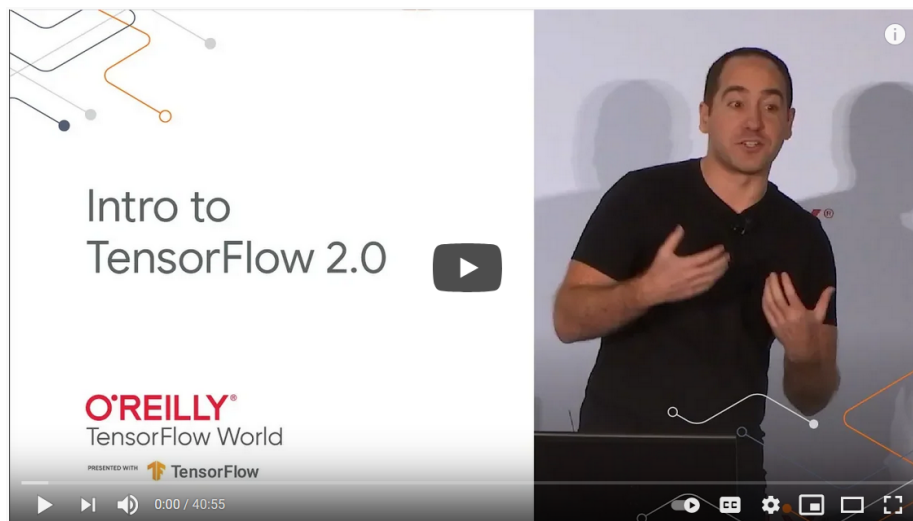
print('\nTest accuracy: ', test_acc, '\nTest Loss: ', test_loss)
print("Accuracy (MeanSquaredError): ", squared_error(predictions, actual).numpy())

# Plot
plt.plot(np.squeeze(train_x), np.squeeze(train_y), 'b.')
plt.plot(np.squeeze(test_x), np.squeeze(predictions), 'g*')
plt.plot(np.squeeze(test_x), np.squeeze(actual), 'r-')
plt.show()

```

Tutorial

Of course, Google has us covered: <https://www.youtube.com/watch?v=5ECD8J3dvDQ>



Website from the video: <https://github.com/tensorflow/workshops>

tensorflow / workshops

Watch 111 Star 720 Fork 334

Code Issues 0 Pull requests 1 Projects 0 Wiki Insights

A few exercises for use at events. <https://tensorflow.org>

81 commits 3 branches 0 releases 8 contributors Apache-2.0

Branch: master New pull request Create new file Upload files Find file Clone or download

random-forests Merge pull request #29 from sararob/sr-tthub-text Latest commit 20359e7 3 days ago

extras	Merge pull request #29 from sararob/sr-tthub-text	3 days ago
images	Adding images folder	9 months ago
CONTRIBUTING.md	adding readme and contrib guidelines	11 months ago
LICENSE	Initial commit	a year ago
README.md	updated location	a month ago

README.md

TensorFlow workshops

Other links:

- Cats versus Dogs (longer version) <https://bit.ly/2G0bWNe>
- <https://colab.research.google.com/>
- <https://js.tensorflow.org/>
- <https://ai.google/education/>