

Introduction of TensorFlow:

An open source machine learning platform

Chee Wei Tan

Table of contents

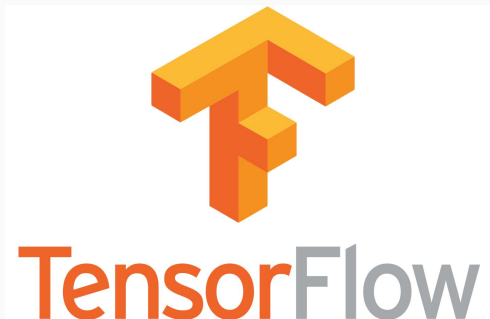
1. Introduction
2. Setup
3. Demo
4. Reference
5. Q&A

Introduction

Introduction

What is TensorFlow?

TensorFlow is an end-to-end open source platform for machine learning (ML). It provides a comprehensive, flexible ecosystem of tools, libraries and community resources that allows researchers to push the state-of-the-art in ML so that developers can easily build and deploy ML powered applications.



Introduction

Why Tensorflow?

- Easy to use



Easy model building

Build and train ML models easily using intuitive high-level APIs like Keras with eager execution, which makes for immediate model iteration and easy debugging.



Robust ML production anywhere

Easily train and deploy models in the cloud, on-prem, in the browser, or on-device no matter what language you use.



Powerful experimentation for research

A simple and flexible architecture to take new ideas from concept to code, to state-of-the-art models, and to publication faster.

Introduction

Why Tensorflow?

- Easy to use
- Widely used



Figure 1: Companies using TensorFlow

Introduction

Why Tensorflow?

- Easy to use
- Widely used



TensorFlow

Learn the foundation of TensorFlow with tutorials for beginners and experts to help you create your next machine learning project.



For JavaScript

Use TensorFlow.js to create new machine learning models and deploy existing models with JavaScript.



For Mobile & IoT

Run inference with TensorFlow Lite on mobile and embedded devices like Android, iOS, Edge TPU, and Raspberry Pi.



For Production

Deploy a production-ready ML pipeline for training and inference using TensorFlow Extended (TFX).

Figure 2: Different platforms of TensorFlow

Introduction

Why Tensorflow?

- Easy to use
- Widely used
- Resources and community support

#PoweredByTF DevPost Challenge



Build something amazing with TF 2.0, share it with the world, and win prizes!

2.0 feature tracker



See details of in-progress, planned, and completed development activities for TensorFlow 2.0.

Google Summer of Code



Get paid to work on an open-source TensorFlow project this summer! Open to undergrad and graduate students.

YouTube



Our YouTube Channel focuses on machine learning and AI with TensorFlow. Explore a number of new shows, including TensorFlow Meets, Ask TensorFlow, and Coding TensorFlow.

Twitter



For up-to-date news and updates from the community and the TensorFlow team, follow @tensorflow on Twitter.

TensorFlow announcements



Join the TensorFlow announcement mailing list to learn about the latest release updates, security advisories, and other important information from the TensorFlow team.

What is TensorFlow.js?

TensorFlow.js is a library for developing ML models in JavaScript such that it can be run in the browser or under Node.js.



A WebGL accelerated, browser based JavaScript library for training and deploying ML models.

Setup

Setup

Two main ways to run TensorFlow.js in the browser:

- Adding the `<script>` tag in the .html file



```
<script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@1.0.0"></script>
```

- Installation from NPM and using a build tool like Parcel, WebPack, or Rollup.



```
yarn add @tensorflow/tfjs  
npm install @tensorflow/tfjs
```

Demo

Linear regression:

- Objective function:

$$h_{\theta}(x^{(i)}) = \theta_0 + \theta_1 x^{(i)},$$

where θ_0 and θ_1 are the parameters that we want to get.

- Loss function, **Mean Squared Error** (MSE):

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2,$$

we want to find a θ that minimize this loss function.

Linear regression:

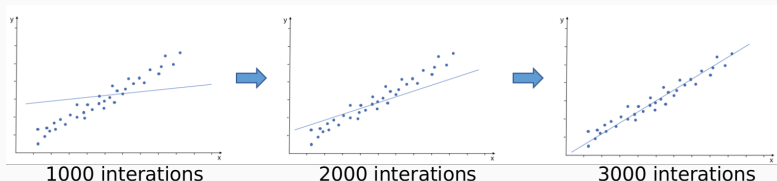
- Gradient descent:

$$\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

where α is the learning rate (e.g. 0.001). We repeat this process until θ converges.

- Learning process:

We randomly initialize θ and update it using the gradient descent method.

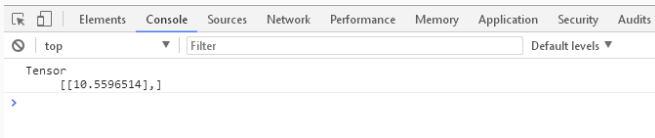


Linear regression:

```
1 <html>
2   <head>
3     <!-- Load TensorFlow.js -->
4     <script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@1.0.0"> </script>
5
6     <!-- Place your code in the script tag below. You can also use an external .js file -->
7     <script>
8       // Notice there is no 'import' statement. 'tf' is available on the index-page
9       // because of the script tag above.
10
11       // Define a model for linear regression.
12       const model = tf.sequential();
13       model.add(tf.layers.dense({units: 1, inputShape: [1]}));
14
15       // Prepare the model for training: Specify the loss and the optimizer.
16       model.compile({loss: 'meanSquaredError', optimizer: 'sgd'});
17
18       // Generate some synthetic data for training.
19       const xs = tf.tensor2d([1, 2, 3, 4, 5], [5, 1]);
20       const ys = tf.tensor2d([1, 3, 5, 7, 9], [5, 1]);
21
22       // Train the model using the data.
23       model.fit(xs, ys, {epochs: 100}).then(() => {
24         // Use the model to do inference on a data point the model hasn't seen before:
25         // Open the browser devtools to see the output
26         model.predict(tf.tensor2d([6], [1, 1])).print();
27       });
28     </script>
29   </head>
30
31   <body>
32   </body>
33 </html>
```

Linear regression:

- Save the codes above in a .html file (e.g. LRdemo.html), and open it with a browser. You can open devtools (F12-Console) to see the output:



The codes are available at:

https://ymq115599.github.io/Demo_LR.html

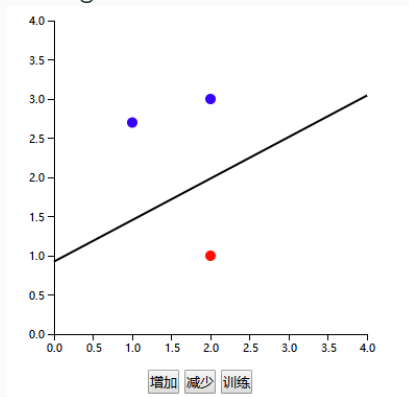
- Change the dataset (xs and ys) to see the difference:

```
18      // Generate some synthetic data for training.
19      const xs = tf.tensor2d([1, 2, 3, 4, 5], [5, 1]);
20      const ys = tf.tensor2d([1, 3, 5, 7, 9], [5, 1]);
```


Linear regression:

- *Can you visualize the result in browser? (hint: add some codes inside the `<body>` tag)

A demo visualizing Support Vector Machine can be found at:
https://ymq115599.github.io/SVM_Demo.html



Reference

<https://tensorflow.google.cn/>

<https://www.tensorflow.org/js/tutorials/setup>

<https://js.tensorflow.org/api/latest/>

<https://www.tensorflow.org/js/guide>

Training a deep learning model using TensorFlow

Table of contents

1. Introduction
2. Setup
3. Code
4. Reference
5. Q&A

Introduction

Usually, we do not retrain the model all the time. Instead, we train and save the model, and load it (in browser) when we need to use it again.

Here, we will introduce how to train the model in local area using TensorFlow (mainly in Python) and use it in browser (website).

Machine learning is like training a program to select the best function.

Machine Learning \approx Looking for a Function

- Speech Recognition

$$f(\text{[audio waveform]}) = \text{"How are you"}$$

- Image Recognition

$$f(\text{[cat image]}) = \text{"Cat"}$$

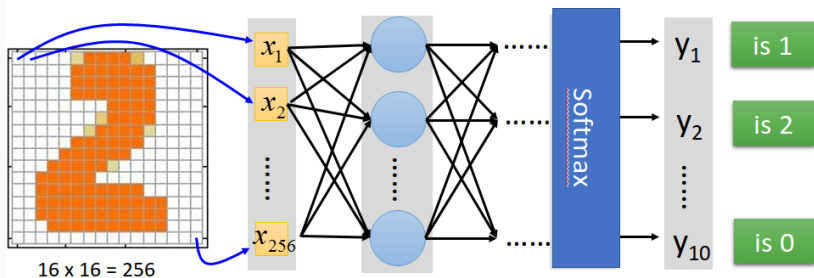
- Playing Go

$$f(\text{[Go board image]}) = \text{"5-5"}_{\text{(next move)}}$$

- Dialogue System

$$f(\text{"Hi"}_{\text{(what the user said)}}) = \text{"Hello"}_{\text{(system response)}}$$

Learning Target





16 x 16 = 256

Ink \rightarrow 1

No ink \rightarrow 0

The learning target is

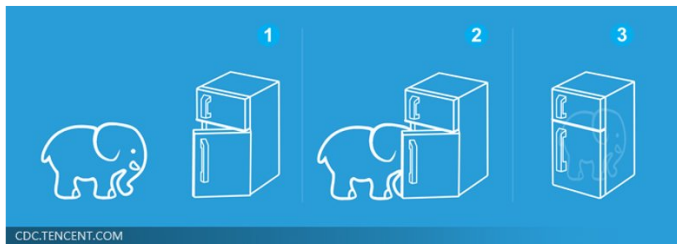
Input:  \rightarrow y_1 has the maximum value

Input:  \rightarrow y_2 has the maximum value

Three Steps for Deep Learning



Deep Learning is so simple



Introduction

We will train a Convolutional Neural Network (CNN) model using TensorFlow for hand-written digit classification.

We use the open source dataset provided by MNIST.



Setup

Environment requirements:

- Python 3 (Anaconda3 is recommended)
- TensorFlow (TensorFlow-GPU if GPU is available)
- TensorFlowjs
- keras

Code

Import necessary packages:



```
import numpy
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from keras.layers import Flatten
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
from keras.optimizers import Adam
from keras.utils import np_utils
```

Load the dataset and data preprocessing:



```
# fix random seed for reproducibility
seed = 7
numpy.random.seed(seed)

# load data
(X_train, y_train), (X_test, y_test) = mnist.load_data()

# Reshaping to format which CNN expects (batch, height, width, channels)
X_train = X_train.reshape(X_train.shape[0], X_train.shape[1], X_train.shape[2], 1).astype('float32')
X_test = X_test.reshape(X_test.shape[0], X_test.shape[1], X_test.shape[2], 1).astype('float32')
```


Data preprocessing:



```
# normalize inputs from 0-255 to 0-1
X_train/=255
X_test/=255

# one hot encode
number_of_classes = 10
y_train = np_utils.to_categorical(y_train, number_of_classes)
y_test = np_utils.to_categorical(y_test, number_of_classes)
```

Build the model:



```
# create model
model = Sequential()
model.add(Conv2D(32, (5, 5), input_shape=(X_train.shape[1], X_train.shape[2], 1), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(32, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.2))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(number_of_classes, activation='softmax'))
```

Train and save the model, evaluate the model using test set:



```
# Compile model
model.compile(loss='categorical_crossentropy', optimizer=Adam(), metrics=['accuracy'])

# Fit the model
model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=10, batch_size=200)

# Save the model
model.save('models/mnistCNN.h5')

# Final evaluation of the model
metrics = model.evaluate(X_test, y_test, verbose=0)
print("Metrics(Test loss & Test Accuracy): ")
print(metrics)
```

Convert the model for future use:



```
import tensorflowjs as tfjs
tfjs.converters.save_keras_model(model, r'D:\Jupyter Program\Machine Learning\Mnist')
```

Several supporting files will be created and named as model.json and group1-shard1of1. These files can help us to load our trained CNN model in the website when we need to use it again in the future.

Reference

[https://medium.com/coinmonks/
handwritten-digit-prediction-using-convolutional-neural-networks](https://medium.com/coinmonks/handwritten-digit-prediction-using-convolutional-neural-networks)
[https:
//github.com/tankala/ai-examples/blob/master/mnistCNN.py](https://github.com/tankala/ai-examples/blob/master/mnistCNN.py)
[https://medium.com/@ashok.tankala/
build-your-first-deep-learning-neural-network-model-using-keras](https://medium.com/@ashok.tankala/build-your-first-deep-learning-neural-network-model-using-keras)