AI Tutorial

IMP Links

* <https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/quickstart/beginner.ipynb#scrollTo=he5u_okAYS4a>
* <https://www.tensorflow.org/lite/guide/hosted_models> (All Tensor Flow Models)
* <https://www.tensorflow.org/lite/examples> (Examples of Tensorflow Models)
* <https://www.tensorflow.org/lite/guide> (Starting guide for Tensorflow Model Development)
* https://www.tensorflow.org/lite/guide/android (Guide for Tensorflow Model Development on Android)
* <https://www.tensorflow.org/lite>
* [Intro to TensorFlow for Deep Learning - Udacity](https://classroom.udacity.com/courses/ud187/lessons/7b590cdb-0acf-4118-848c-8728ced19bc6/concepts/4e1b5f7e-5094-481e-ae29-284bc2fb847f) (Crash Course for beginners to learn Deep Learning) – added on 04-05-2022
* TensorFlow Lite is a set of tools that enables on-device machine learning by helping developers run their models on mobile, embedded, and edge devices.

**Development workflow**

### **1. Generate a TensorFlow Lite model**

A TensorFlow Lite model is represented in a special efficient portable format known as [FlatBuffers](https://google.github.io/flatbuffers/). This provides several advantages over TensorFlow's protocol buffer model format such as reduced size (small code footprint) and faster inference (data is directly accessed without an extra parsing/unpacking step) that enables TensorFlow Lite to execute efficiently on devices with limited compute and memory resources.

You can generate a TensorFlow Lite model in the following ways:

* **Use an existing TensorFlow Lite model:** Refer to [TensorFlow Lite Examples](https://www.tensorflow.org/lite/examples) to pick an existing model. Models may or may not contain metadata.
* **Create a TensorFlow Lite model:** Use the [TensorFlow Lite Model Maker](https://www.tensorflow.org/lite/guide/model_maker) to create a model with your own custom dataset. By default, all models contain metadata.
* **Convert a TensorFlow model into a TensorFlow Lite model:** Use the [TensorFlow Lite Converter](https://www.tensorflow.org/lite/convert/index) to convert a TensorFlow model into a TensorFlow Lite model. During conversion, you can apply [optimizations](https://www.tensorflow.org/lite/performance/model_optimization) such as [quantization](https://www.tensorflow.org/lite/performance/post_training_quantization) to reduce model size and latency with minimal or no loss in accuracy. By default, all models don't contain metadata.

**Simple ML Code to create a model and to predict Temperature from Celsius to Fahrenheit**

Link for the code : https://colab.research.google.com/drive/1MUg0qYcnmmGV6Iw3926WTSjL9HTDXVe6#scrollTo=mp7NnuK5jzAO

import tensorflow as tf

import numpy as np

#input data which is in Celsius format

x = np.array([0, 10, 20, 30, 38, 32], dtype=float)

y = np.array([32, 50, 68, 86, 100, 90], dtype=float)

#Creating Layers to process the input data and to predict the output

l1 = tf.keras.layers.Dense(units=4, name='l1', input\_shape=[1])

l2 = tf.keras.layers.Dense(units=4, name='l2')

l3 = tf.keras.layers.Dense(units=1, name='l3')

#Creating Model

model = tf.keras.Sequential([l1, l2, l3])

#Compile the model

model.compile(

    loss='mean\_squared\_error',

    optimizer=tf.keras.optimizers.Adam(0.1),

)

#Train the model

history = model.fit(x, y, epochs=500,verbose=False)

#Visualize the loss rate

import matplotlib.pyplot as plt

plt.xlabel("Epochs")

plt.ylabel("Loss")

plt.plot(history.history['loss'])

#predict the model

print(model.predict([100.0]))

# Summary

In this lesson we trained a neural network to classify images of articles of clothing. To do this we used the Fashion MNIST dataset, which contains 70,000 greyscale images of articles of clothing. We used 60,000 of them to train our network and 10,000 of them to test its performance. In order to feed these images into our neural network we had to flatten the 28 × 28 images into 1d vectors with 784 elements. Our network consisted of a fully connected layer with 128 units (neurons) and an output layer with 10 units, corresponding to the 10 output labels. These 10 outputs represent probabilities for each class. The softmax activation function calculated the probability distribution.

We also learned about the differences between regression and classification problems.

* **Regression:** A model that outputs a single value. For example, an estimate of a house’s value.
* **Classification:** A model that outputs a probability distribution across several categories. For example, in Fashion MNIST, the output was 10 probabilities, one for each of the different types of clothing. Remember, we use Softmax as the activation function in our last Dense layer to create this probability distribution.

Table

Description automatically generated