TENSOR FLOW in PYTHON

API Application Program Interface

Tensor Flow uses three main API's that Keras, Lego-like building block for building and defining models, tf.data is an easy input pipeline and Eager execution, which makes TensorFlow feel like regular Python.

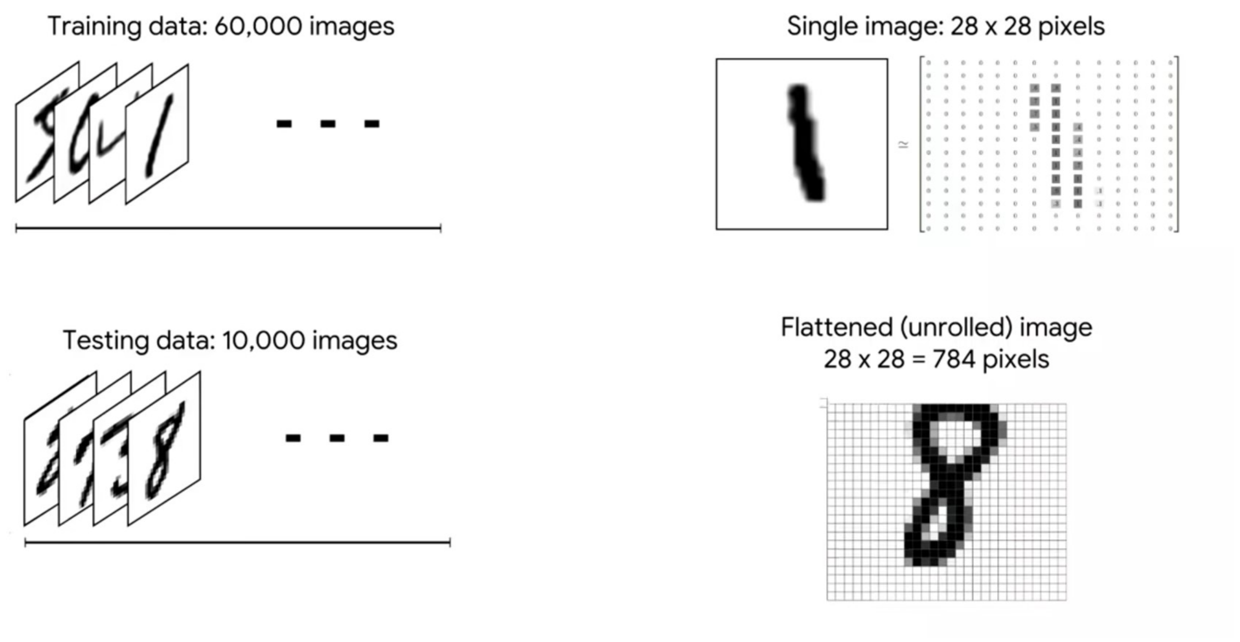
Dr. Randy Davila pointed out that going to Colab.research.google.com which is baiscally a virtural enviorment where you can write your neural network. The advantages is that the site already has the packages, pip's, snippets and access to GPU's.

Thoughtfully designing an experiment is much more important than the accuracy. What are you trying to predict and why; how will it be used in practice; what could go wrong and why; and where did the data come from.

Steps to writing neural network

1. Collect a data set
2. Build your model
3. Train
4. Evaluate
5. Predict

Using MNIST Data Set.


Training data: 60,000 images
Single image: 28 x 28 pixels
Testing data: 10,000 images
Flattened (unrolled) image
28 x 28 = 784 pixels

Step 1 Data

Build the data set.

```
import tensorflow as tf                                     #Step1 import tensorflow
mnist = tf.keras.datasets.mnist                             #importing mnist included into tensorflow
```

Step 2 Building Model

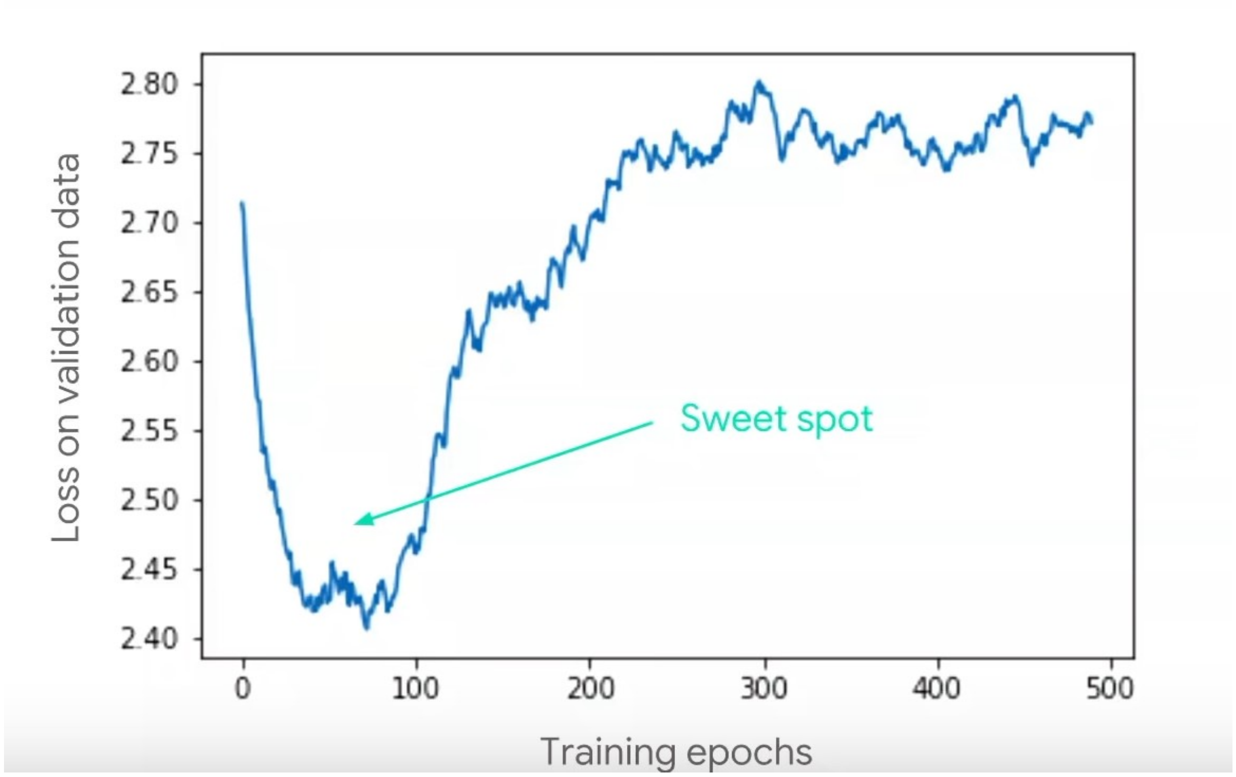
The optimizer used is adam - an optimization algorithm that can used instead of the classical stochastic gradient descent procedure to update network weights iterative based in training data.

```
import tensorflow as tf                                     #Step1 import tensorflow
mnist = tf.keras.datasets.mnist                             #importing mnist included into tensorflow
```

Step 3 Train Data

This one line trains the model. The only parameter that matters is epoch, which means one sweep across the network. Loss is that same as error.

```
model.fit(x_train, y_train, epochs=5)                           #Step3 train data
```



Loss is that same as error. The idea is that the model iterates across the sweet spot and finds the extrema, the smallest loss.

Step 4 Evaluate

Given some new data, classify with network and take a look at accuracy. This will give loss and error. This model has an accuracy of 98 percent classification.

```
model.evaluate(x_test, y_test)                                 #Step4 evaluate
```

Step 5 Predict

Makes a prediction of character to the data set.

```
model.evaluate(x_test, y_test)                                 #Step4 evaluate
```

In [1]:
```
#***** DO NOT RUN CELL UNLESS YOU HAVE TIME TO STAND-BY********

import tensorflow as tf                                       #Step1 import tensorfl
ow
mnist = tf.keras.datasets.mnist                               #importing mnist inclu
ded into tensorflow

(x_train, y_train),(x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([                           #Step 2 Build the mod
el
  tf.keras.layers.Flatten(input_shape=(28, 28)),               #takes input and puts
it into vector
  tf.keras.layers.Dense(512, activation=tf.nn.relu),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10, activation=tf.nn.softmax)
])
model.compile(optimizer='adam',                                #compile the network
compare the thing that the networ
              loss='sparse_categorical_crossentropy',          #predicted to the thi
ng you wanted it to predict
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)                          #Step3 train data
model.evaluate(x_test, y_test)                                 #Step4 evaluate
```

```
WARNING:tensorflow:From C:\Users\jpg63\Anaconda3\envs\tensoflow\lib\site-package
s\tensorflow\python\ops\resource_variable_ops.py:435: colocate_with (from tensor
flow.python.framework.ops) is deprecated and will be removed in a future version
.
Instructions for updating:
Colocations handled automatically by placer.
WARNING:tensorflow:From C:\Users\jpg63\Anaconda3\envs\tensoflow\lib\site-package
s\tensorflow\python\keras\layers\core.py:143: calling dropout (from tensorflow.p
ython.ops.nn_ops) with keep_prob is deprecated and will be removed in a future v
ersion.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - keep
_prob`.
Epoch 1/5
60000/60000 [==============================] - 10s 166us/sample - loss: 0.2194 -
acc: 0.9351
Epoch 2/5
60000/60000 [==============================] - 10s 168us/sample - loss: 0.0959 -
acc: 0.9704
Epoch 3/5
60000/60000 [==============================] - 9s 156us/sample - loss: 0.0691 -
acc: 0.9782
Epoch 4/5
60000/60000 [==============================] - 10s 164us/sample - loss: 0.0528 -
acc: 0.9825
Epoch 5/5
60000/60000 [==============================] - 10s 163us/sample - loss: 0.0430 -
acc: 0.9862
10000/10000 [==============================] - 0s 49us/sample - loss: 0.0750 - a
cc: 0.9786
```

Out[1]:
```
[0.0749650876199943, 0.9786]
```

In [2]:
```
predict=model.predict(x_train[0:9])                           #step5 predict
predict[2]
```

Out[2]:
```
array([1.2198483e-08, 6.2375932e-07, 1.9702720e-05, 2.7362552e-07,
       9.9854589e-01, 1.4648179e-08, 2.4020858e-07, 2.2372507e-05,
       1.2853000e-07, 1.4107871e-03], dtype=float32)
```