TensorFlow 2 quickstart for beginners

Run in Google Colab (https://colab.research.google.com/github/tensorflow/docs/blob/master/site/

View source on GitHub (https://github.com/tensorflow/docs/blob/master/site/en/tutorials/quicksta

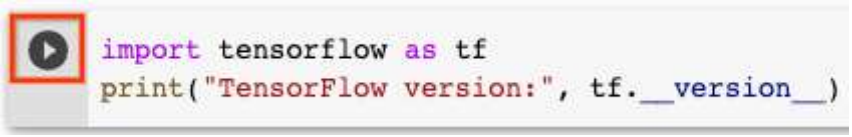
Download notebook (https://storage.googleapis.com/tensorflow_docs/docs/site/en/tutorials/quick

This short introduction uses Keras (https://www.tensorflow.org/guide/keras/overview) to:

1. Load a prebuilt dataset.
2. Build a neural network machine learning model that classifies images.
3. Train this neural network.
4. Evaluate the accuracy of the model.

This tutorial is a Google Colaboratory (https://colab.research.google.com/notebooks/welcome.ipynb) notebook. Python programs are run directly in the browser—a great way to learn and use TensorFlow. To follow this tutorial, run the notebook in Google Colab by clicking the button at the top of this page.

1. In Colab, connect to a Python runtime: At the top-right of the menu bar, select *CONNECT*.
2. To run all the code in the notebook, select **Runtime** > **Run all**. To run the code cells one at a time, hover over each cell and select the **Run cell** icon.



Set up TensorFlow

Import TensorFlow into your program to get started:

```
import tensorflow as tf
print("TensorFlow version:", tf.__version__)
```

```
2024-08-16 07:45:15.387747: E external/local_xla/xla/stream_executor/cuda/cuda
2024-08-16 07:45:15.408731: E external/local_xla/xla/stream_executor/cuda/cuda
2024-08-16 07:45:15.415209: E external/local_xla/xla/stream_executor/cuda/cuda
TensorFlow version: 2.17.0
```

If you are following along in your own development environment, rather than Colab (https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/quickstart/beginner.ipynb), see the install guide (https://www.tensorflow.org/install) for setting up TensorFlow for development.

Note: Make sure you have upgraded to the latest `pip` to install the TensorFlow 2 package if you are using your own development environment. See the install guide (https://www.tensorflow.org/install) for details.

Load a dataset

Load and prepare the MNIST dataset. The pixel values of the images range from 0 through 255. Scale these values to a range of 0 to 1 by dividing the values by `255.0`. This also converts the sample data from integers to floating-point numbers:

```
mnist = tf.keras.datasets.mnist

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

Build a machine learning model

Build a `tf.keras.Sequential` (https://www.tensorflow.org/api_docs/python/tf/keras/Sequential) model:

```
model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
```

```
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10)
])
```

```
/tmpfs/src/tf_docs_env/lib/python3.9/site-packages/keras/src/layers/reshaping
  super().__init__(**kwargs)
WARNING: All log messages before absl::InitializeLog() is called are written
I0000 00:00:1723794318.490455  241277 cuda_executor.cc:1015] successful NUMA
I0000 00:00:1723794318.494342  241277 cuda_executor.cc:1015] successful NUMA
I0000 00:00:1723794318.497584  241277 cuda_executor.cc:1015] successful NUMA
I0000 00:00:1723794318.501312  241277 cuda_executor.cc:1015] successful NUMA
I0000 00:00:1723794318.512702  241277 cuda_executor.cc:1015] successful NUMA
I0000 00:00:1723794318.516197  241277 cuda_executor.cc:1015] successful NUMA
I0000 00:00:1723794318.519187  241277 cuda_executor.cc:1015] successful NUMA
I0000 00:00:1723794318.522647  241277 cuda_executor.cc:1015] successful NUMA
I0000 00:00:1723794318.526047  241277 cuda_executor.cc:1015] successful NUMA
I0000 00:00:1723794318.529503  241277 cuda_executor.cc:1015] successful NUMA
```

`Sequential` (https://www.tensorflow.org/guide/keras/sequential_model) is useful for stacking layers where each layer has one input tensor (https://www.tensorflow.org/guide/tensor) and one output tensor. Layers are functions with a known mathematical structure that can be reused and have trainable variables. Most TensorFlow models are composed of layers. This model uses the `Flatten` (https://www.tensorflow.org/api_docs/python/tf/keras/layers/Flatten), `Dense` (https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dense), and `Dropout` (https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dropout) layers.

For each example, the model returns a vector of logits (https://developers.google.com/machine-learning/glossary#logits) or log-odds (https://developers.google.com/machine-learning/glossary#log-odds) scores, one for each class.

```
predictions = model(x_train[:1]).numpy()
predictions
```

```
array([[ 0.68130803, -0.03935227, -0.53304887,  0.22200397, -0.3079031 ,
        -0.6267688 ,  0.43393654,  0.5691322 ,  0.31098977,  0.32141146]],
      dtype=float32)
```

The **`tf.nn.softmax`** (https://www.tensorflow.org/api_docs/python/tf/nn/softmax) function converts these logits to *probabilities* for each class:

```
tf.nn.softmax(predictions).numpy()
```

```
array([[0.16339162, 0.07947874, 0.04851112, 0.10321827, 0.06076043,
        0.0441712 , 0.12758444, 0.14605366, 0.11282429, 0.11400625]],
      dtype=float32)
```

Note: It is possible to bake the **`tf.nn.softmax`** (https://www.tensorflow.org/api_docs/python/tf/nn/softmax) function into the activation function for the last layer of the network. While this can make the model output more directly interpretable, this approach is discouraged as it's impossible to provide an exact and numerically stable loss calculation for all models when using a softmax output.

Define a loss function for training using **`losses.SparseCategoricalCrossentropy`** (https://www.tensorflow.org/api_docs/python/tf/keras/losses/SparseCategoricalCrossentropy):

```
loss_fn = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)
```

The loss function takes a vector of ground truth values and a vector of logits and returns a scalar loss for each example. This loss is equal to the negative log probability of the true class: The loss is zero if the model is sure of the correct class.

This untrained model gives probabilities close to random (1/10 for each class), so the initial loss should be close to **`-tf.math.log(1/10) ~= 2.3`**.

```
loss_fn(y_train[:1], predictions).numpy()
```

```
3.1196823
```

Before you start training, configure and compile the model using Keras **`Model.compile`** (https://www.tensorflow.org/api_docs/python/tf/keras/Model#compile). Set the **`optimizer`** (https://www.tensorflow.org/api_docs/python/tf/keras/optimizers) class to **`adam`**, set the **`loss`** to the **`loss_fn`** function you defined earlier, and specify a metric to be evaluated for the model by setting the **`metrics`** parameter to **`accuracy`**.

```
model.compile(optimizer='adam',
              loss=loss_fn,
              metrics=['accuracy'])
```

Train and evaluate your model

Use the **`Model.fit`** (https://www.tensorflow.org/api_docs/python/tf/keras/Model#fit) method to adjust your model parameters and minimize the loss:

```
model.fit(x_train, y_train, epochs=5)
```

```
Epoch 1/5
WARNING: All log messages before absl::InitializeLog() is called are written
I0000 00:00:1723794322.305243  241442 service.cc:146] XLA service 0x7effb8008
I0000 00:00:1723794322.305276  241442 service.cc:154]   StreamExecutor device
I0000 00:00:1723794322.305281  241442 service.cc:154]   StreamExecutor device
I0000 00:00:1723794322.305284  241442 service.cc:154]   StreamExecutor device
I0000 00:00:1723794322.305287  241442 service.cc:154]   StreamExecutor device
112/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.6089 - loss: 1.3300
I0000 00:00:1723794323.392324  241442 device_compiler.h:188] Compiled cluster
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 4s 1ms/step - accuracy: 0.8622 - loss: 0.4811
Epoch 2/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.9547 - loss: 0.1539
Epoch 3/5
```

The **`Model.evaluate`** (https://www.tensorflow.org/api_docs/python/tf/keras/Model#evaluate) method checks the model's performance, usually on a validation set (https://developers.google.com/machine-learning/glossary#validation-set) or test set (https://developers.google.com/machine-learning/glossary#test-set).

```
model.evaluate(x_test,  y_test, verbose=2)
```

```
313/313 - 1s - 3ms/step - accuracy: 0.9782 - loss: 0.0729
[0.07293704897165298, 0.9782000184059143]
```

The image classifier is now trained to ~98% accuracy on this dataset. To learn more, read the TensorFlow tutorials (https://www.tensorflow.org/tutorials/).

If you want your model to return a probability, you can wrap the trained model, and attach the softmax to it:

```
probability_model = tf.keras.Sequential([
  model,
  tf.keras.layers.Softmax()
])
```

```
probability_model(x_test[:5])
```

```
<tf.Tensor: shape=(5, 10), dtype=float32, numpy=
array([[1.5427084e-07, 7.5027339e-11, 3.1343968e-06, 4.6326011e-05,
        8.9990645e-13, 1.5266414e-07, 2.0456495e-13, 9.9994934e-01,
        2.1858141e-07, 7.8530559e-07],
       [1.7771253e-08, 8.4947787e-05, 9.9989736e-01, 1.8331458e-06,
        8.3026415e-15, 3.4793761e-08, 6.2480517e-08, 7.9319728e-12,
        1.5733674e-05, 3.5440111e-15],
       [3.3602277e-07, 9.9804592e-01, 5.7737787e-05, 5.8099768e-06,
        6.3599517e-05, 2.3768812e-06, 2.3459031e-06, 1.6781164e-03,
        1.4260423e-04, 1.0617223e-06],
       [9.9997318e-01, 8.7561805e-11, 9.8983969e-07, 9.0878149e-10,
        1.0803159e-07, 3.3033965e-07, 2.3622524e-05, 6.7567669e-07,
        4.7765565e-09, 1.1131582e-06],
```

Conclusion

Congratulations! You have trained a machine learning model using a prebuilt dataset using the Keras (https://www.tensorflow.org/guide/keras/overview) API.

For more examples of using Keras, check out the tutorials (https://www.tensorflow.org/tutorials/keras/). To learn more about building models with Keras, read the guides (https://www.tensorflow.org/guide/keras). If you want learn more about loading and preparing data, see the tutorials on image data loading (https://www.tensorflow.org/tutorials/load_data/images) or CSV data loading (https://www.tensorflow.org/tutorials/load_data/csv).

Last updated 2024-08-16 UTC.