國立清華大學
National Tsing Hua University

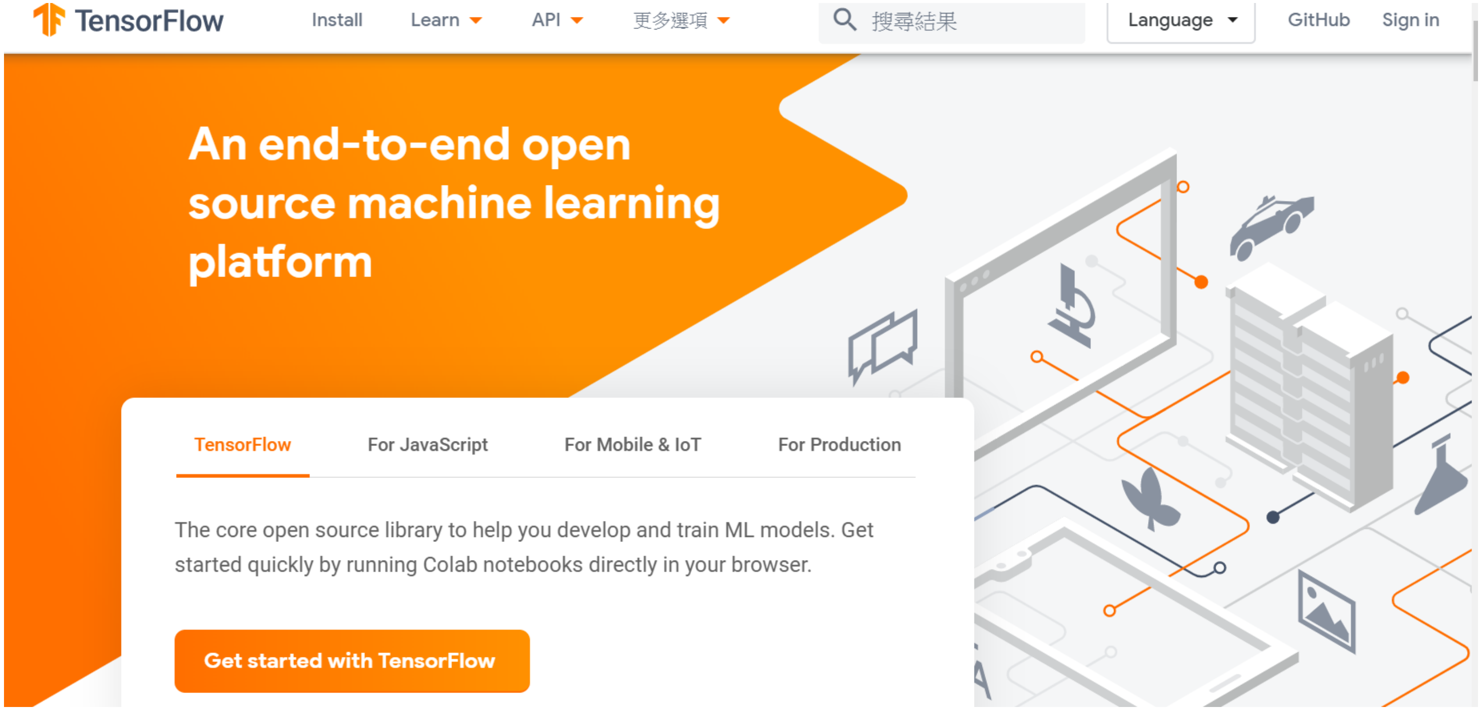
INNOVATION
ENGINEERING
LAB

智慧化企業整合
Intelligent Integration of Enterprise

Introduction of TensorFlow

助教:黃日泓

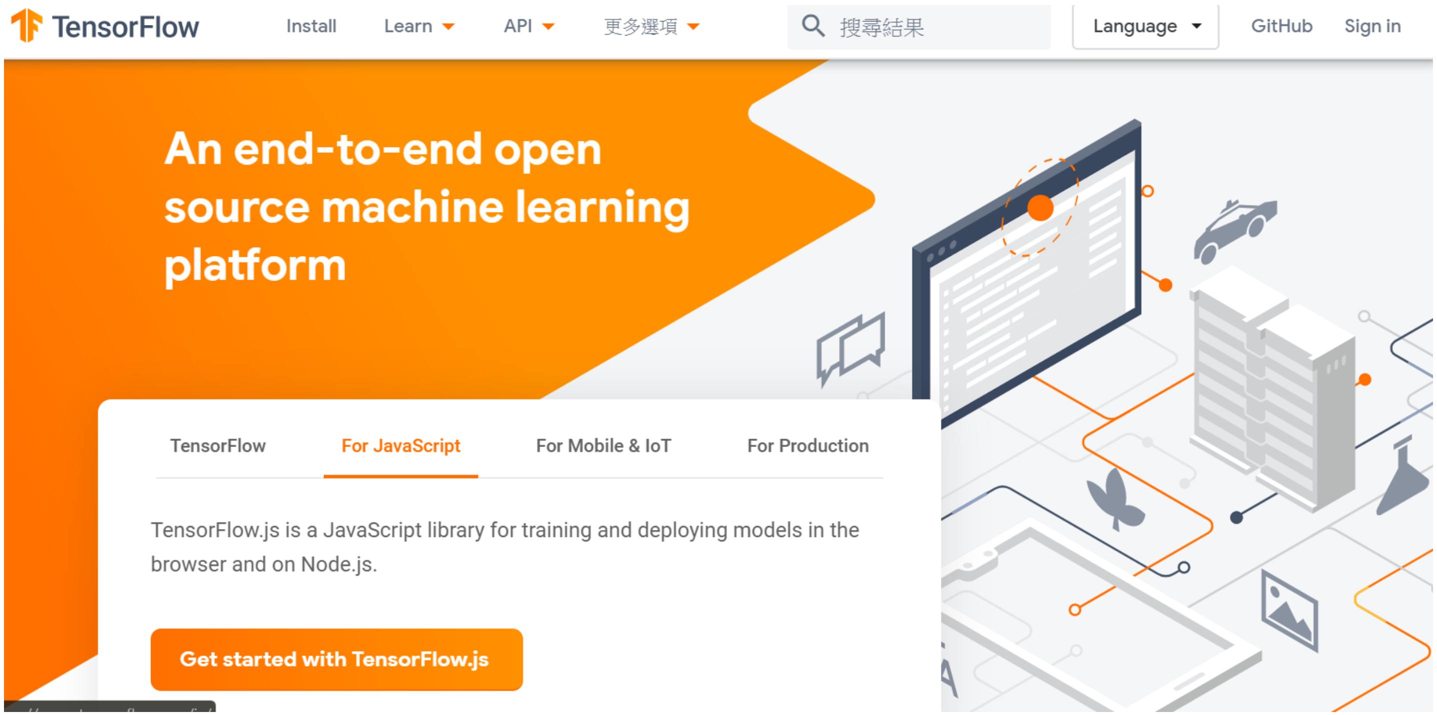
TensorFlow



An end-to-end open source machine learning platform

The core open source library to help you develop and train ML models. Get started quickly by running Colab notebooks directly in your browser.

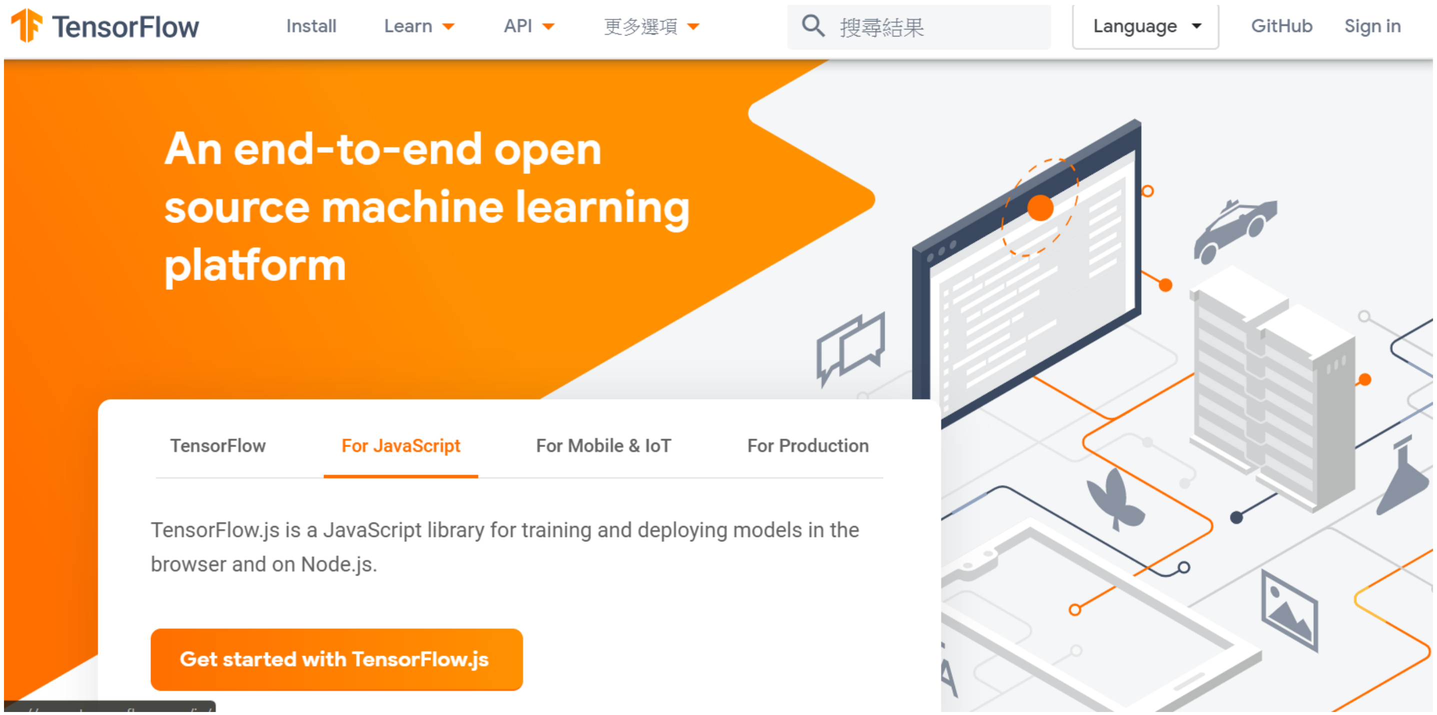
TensorFlow



An end-to-end open source machine learning platform

TensorFlow.js is a JavaScript library for training and deploying models in the browser and on Node.js.

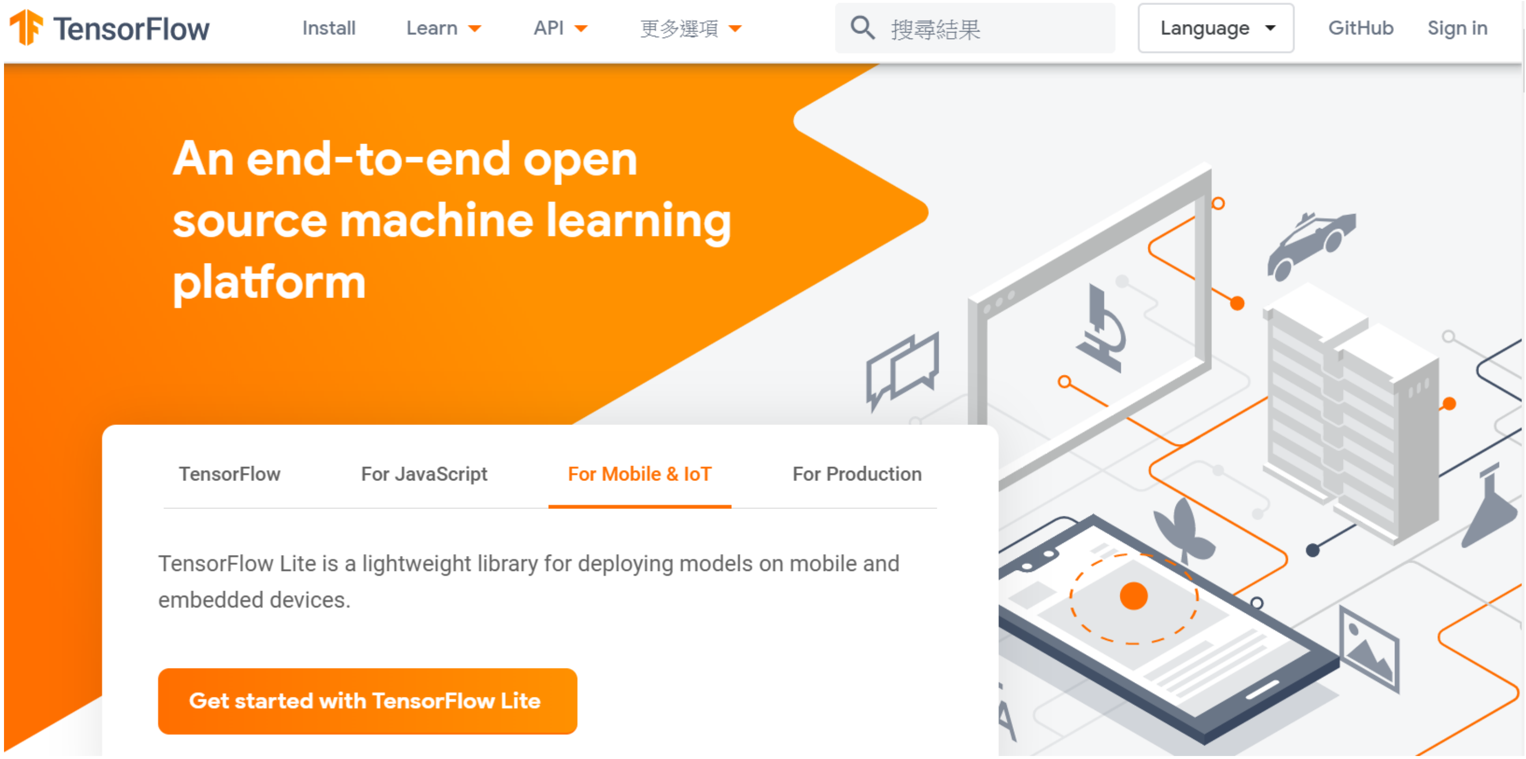
TensorFlow



An end-to-end open source machine learning platform

TensorFlow.js is a JavaScript library for training and deploying models in the browser and on Node.js.

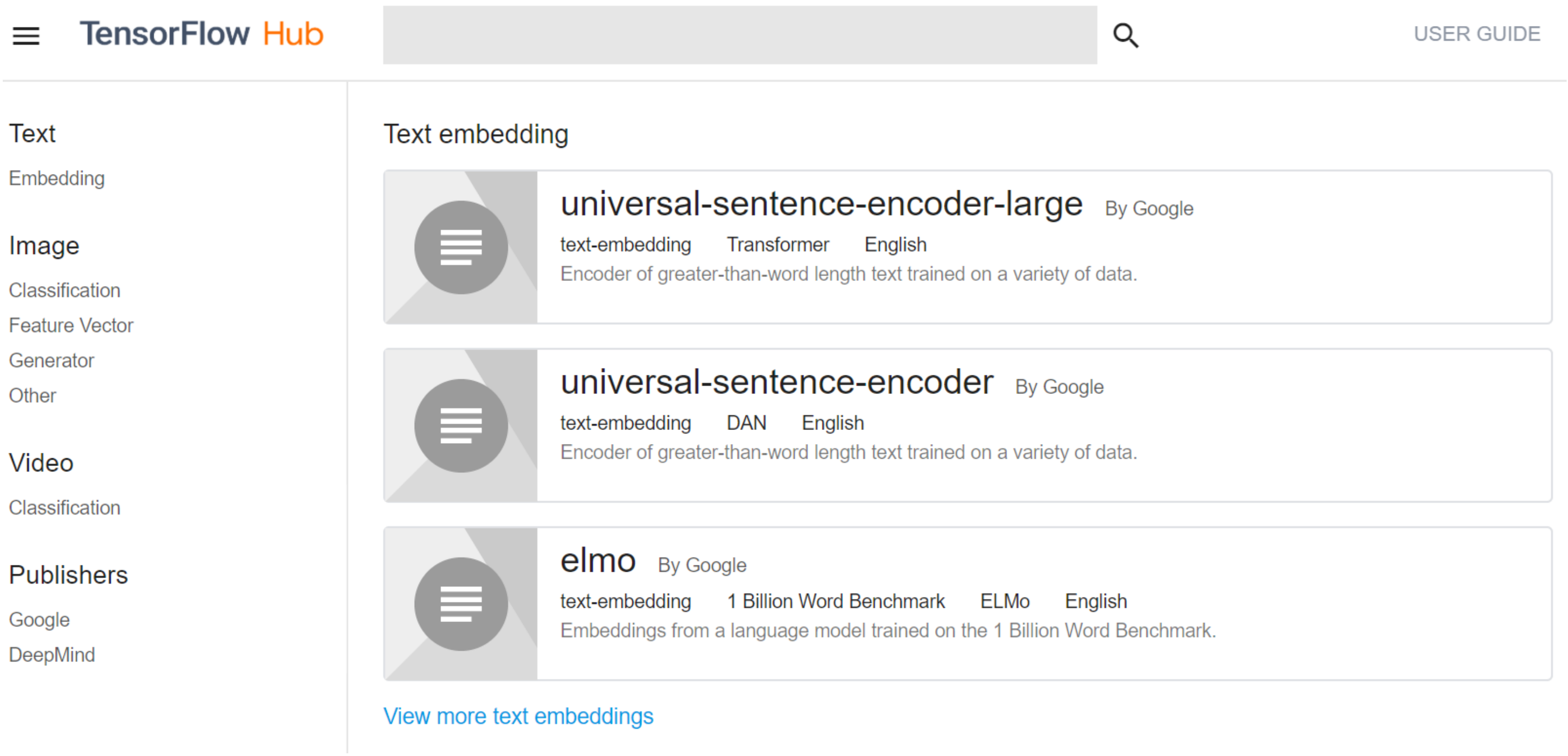
TensorFlow



An end-to-end open source machine learning platform

TensorFlow Lite is a lightweight library for deploying models on mobile and embedded devices.

TensorFlow Hub



TensorFlow Hub

USER GUIDE

Text

Embedding

Image

Classification

Feature Vector

Generator

Other

Video

Classification

Publishers

Google

DeepMind

Text embedding

universal-sentence-encoder-large By Google

text-embedding Transformer English

Encoder of greater-than-word length text trained on a variety of data.

universal-sentence-encoder By Google

text-embedding DAN English

Encoder of greater-than-word length text trained on a variety of data.

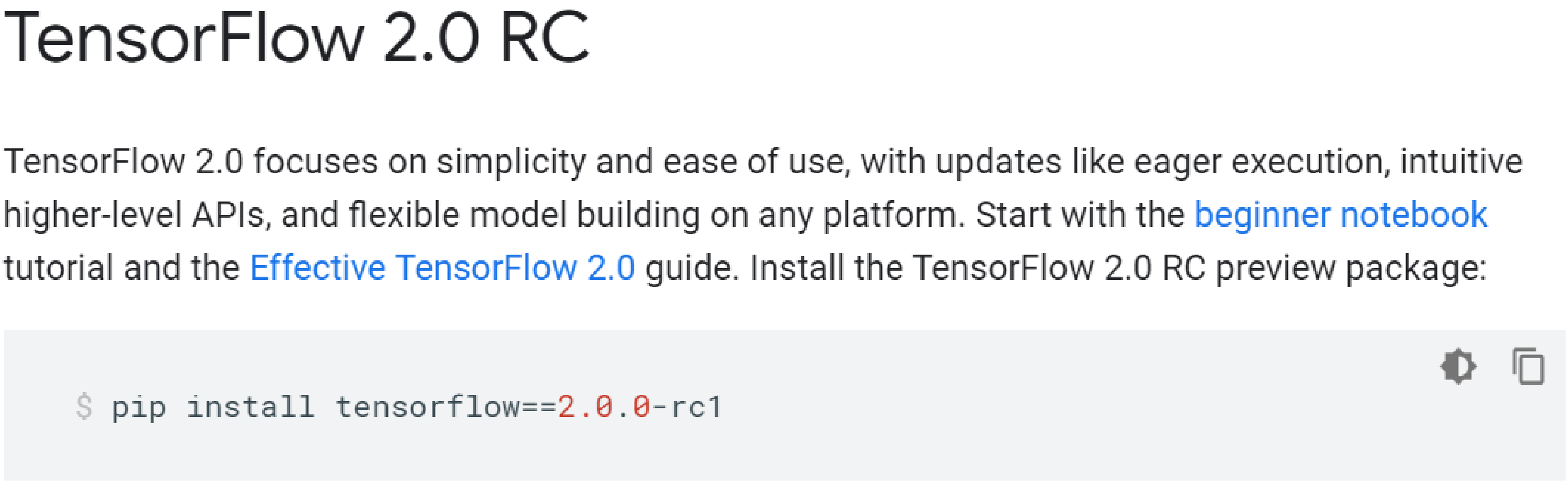
elmo By Google

text-embedding 1 Billion Word Benchmark ELMo English

Embeddings from a language model trained on the 1 Billion Word Benchmark.

View more text embeddings

TensorFlow2.0 installation



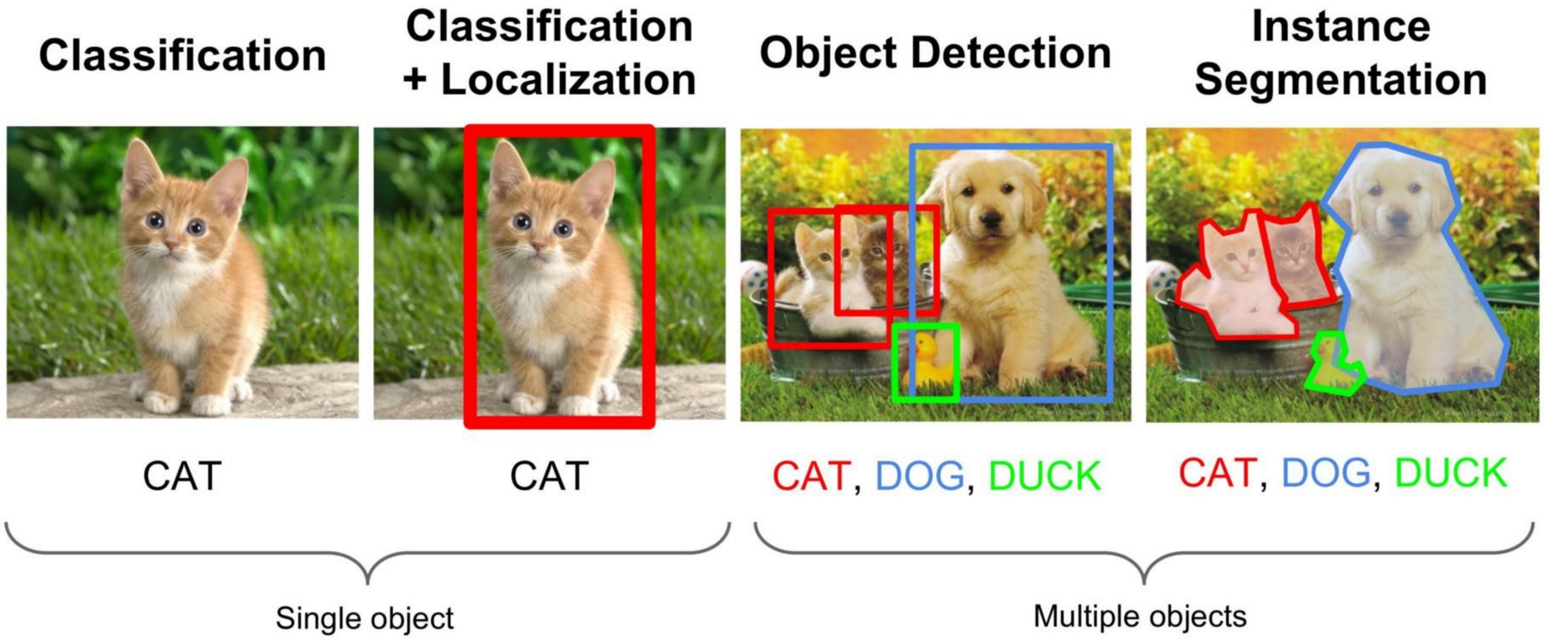
TensorFlow 2.0 RC

TensorFlow 2.0 focuses on simplicity and ease of use, with updates like eager execution, intuitive higher-level APIs, and flexible model building on any platform. Start with the beginner notebook tutorial and the Effective TensorFlow 2.0 guide. Install the TensorFlow 2.0 RC preview package:

```
$ pip install tensorflow==2.0.0-rc1
```

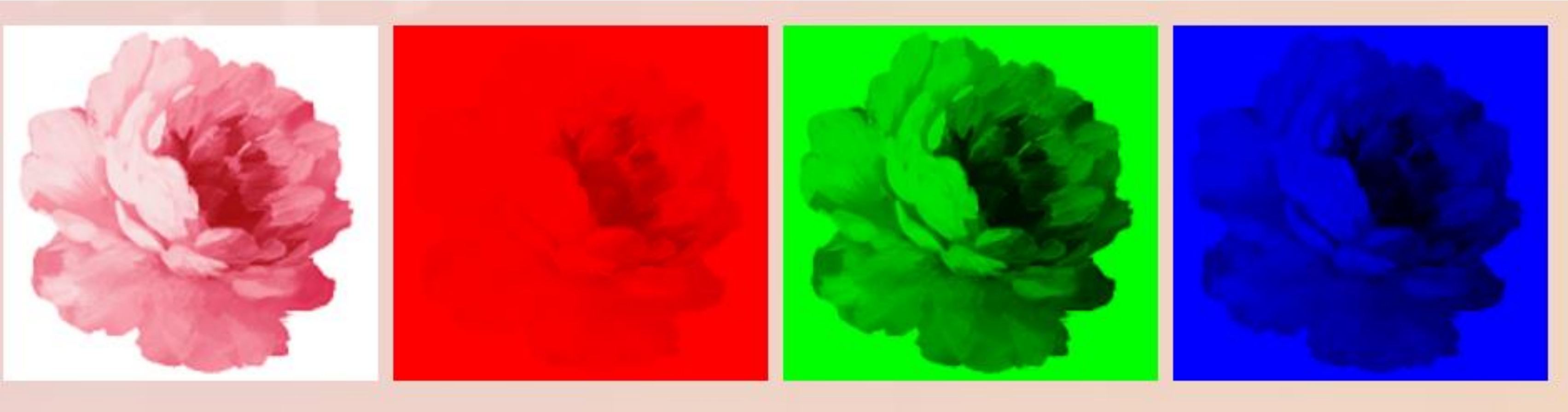
Other reference

Image-related tasks

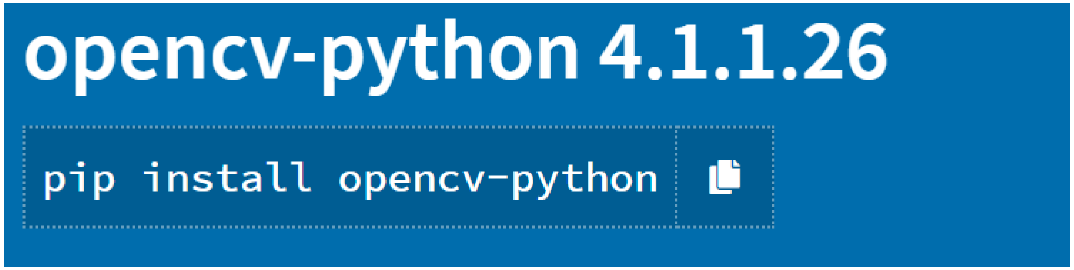


Colorful images

- RGB value : 0~255 (from dark to bright)
- Colorful images: 3 channels
- Grayscale image: 1 channel

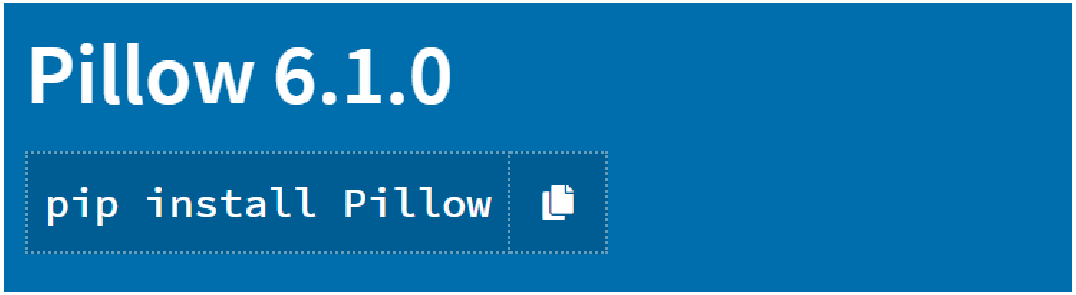


Tools for image processing



opencv-python 4.1.1.26

```
pip install opencv-python
```



Pillow 6.1.0

```
pip install Pillow
```

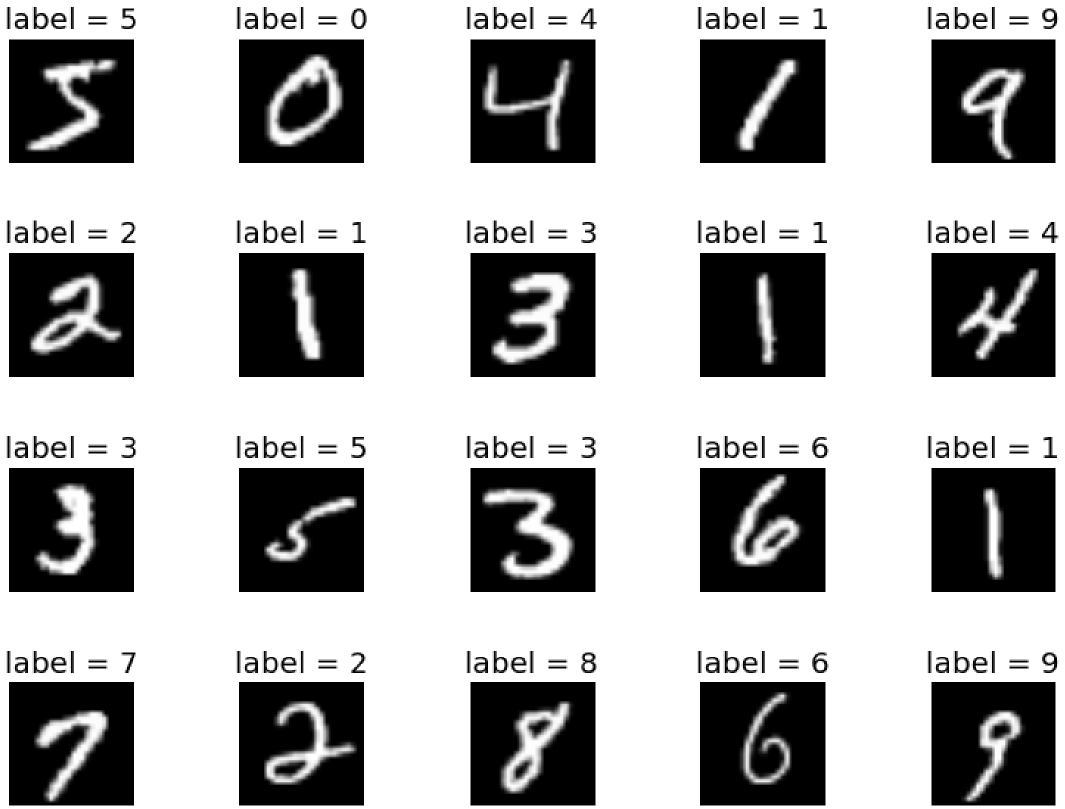
matplotlib

Version 3.1.1

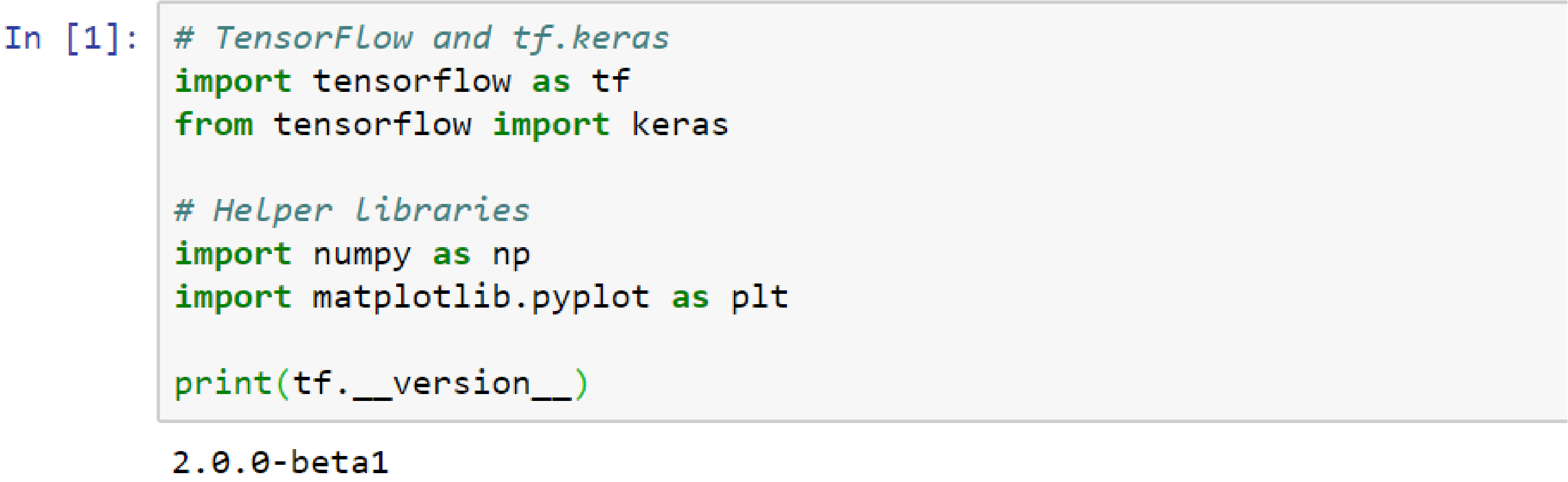
DNN with TensorFlow

Mnist
dataset

Mnist dataset



Import required packages



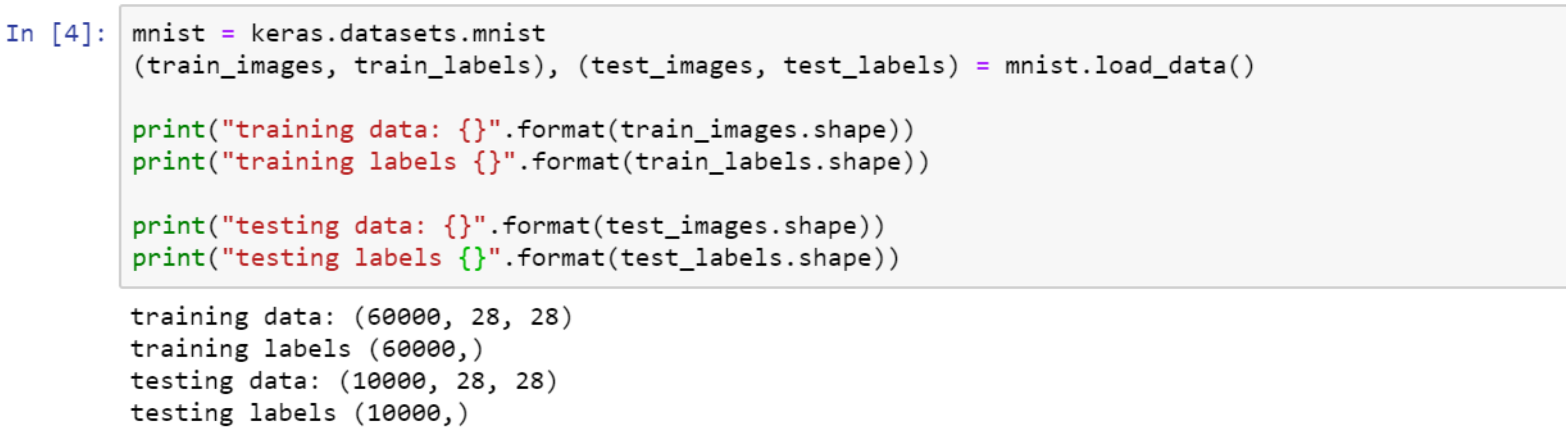
```
In [1]: # TensorFlow and tf.keras
        import tensorflow as tf
        from tensorflow import keras

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

        print(tf.__version__)

        2.0.0-beta1
```

Load data



In [4]:

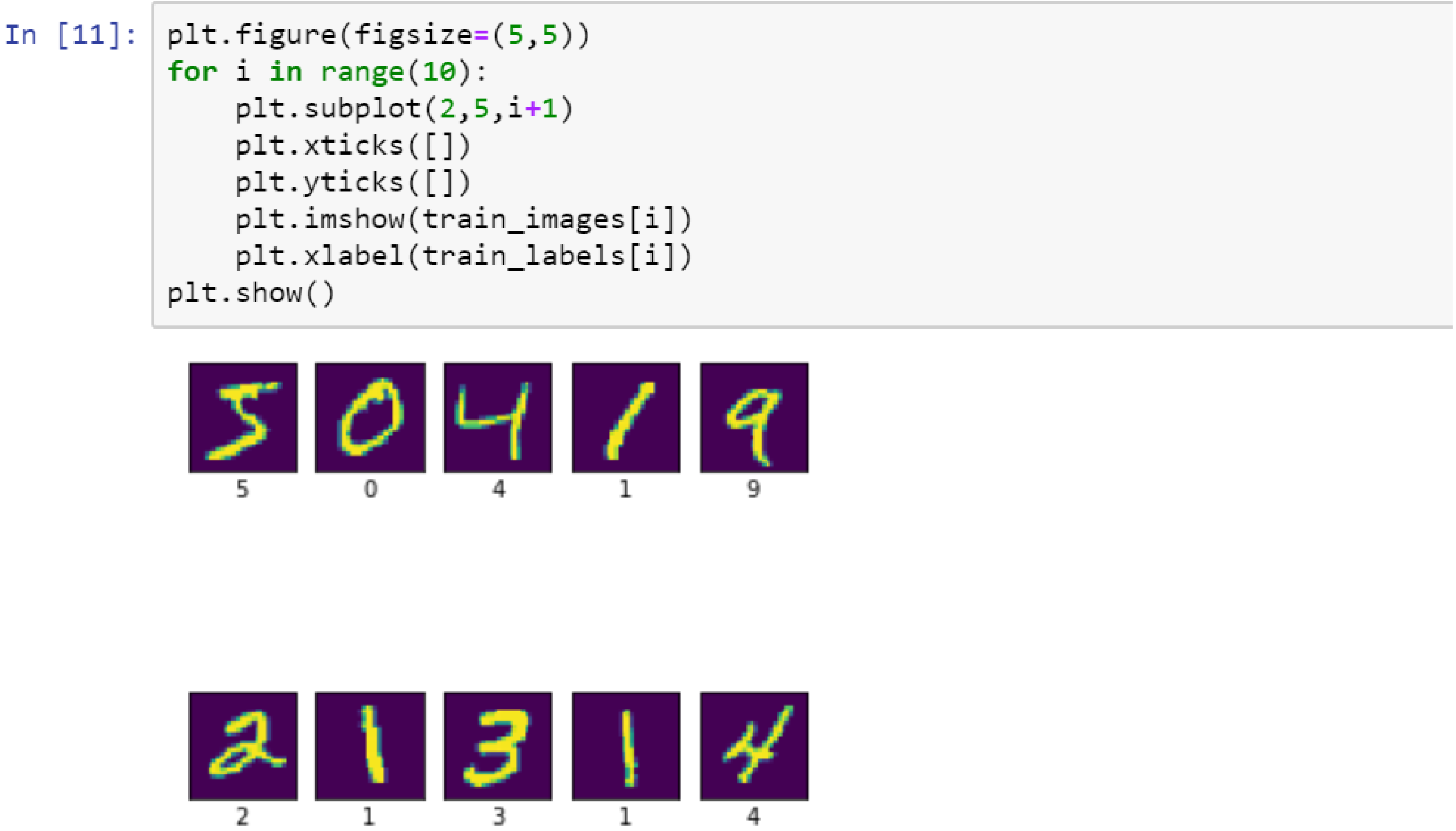
```
mnist = keras.datasets.mnist
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()

print("training data: {}".format(train_images.shape))
print("training labels {}".format(train_labels.shape))

print("testing data: {}".format(test_images.shape))
print("testing labels {}".format(test_labels.shape))
```

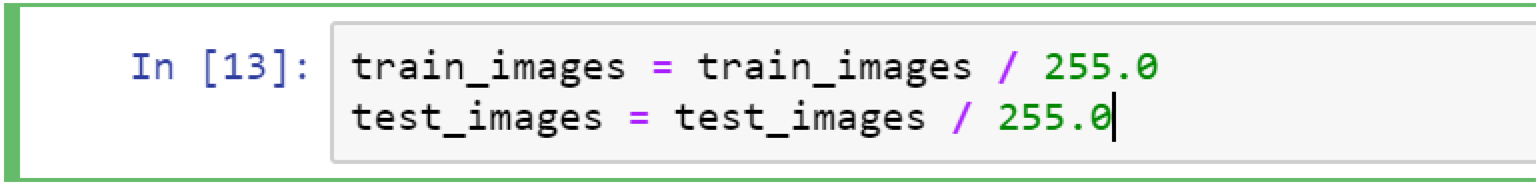
```
training data: (60000, 28, 28)
training labels (60000,)
testing data: (10000, 28, 28)
testing labels (10000,)
```

See data



In [11]:
```
plt.figure(figsize=(5,5))
for i in range(10):
    plt.subplot(2,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.imshow(train_images[i])
    plt.xlabel(train_labels[i])
plt.show()
```

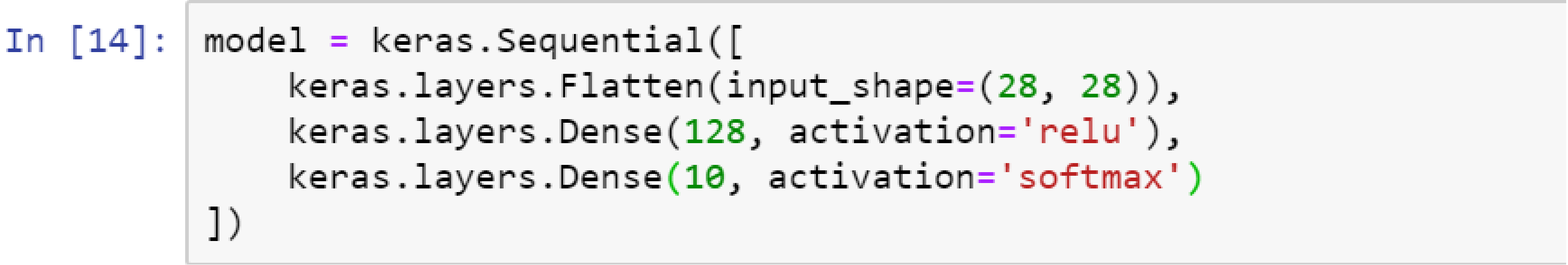
Standardize



In [13]:

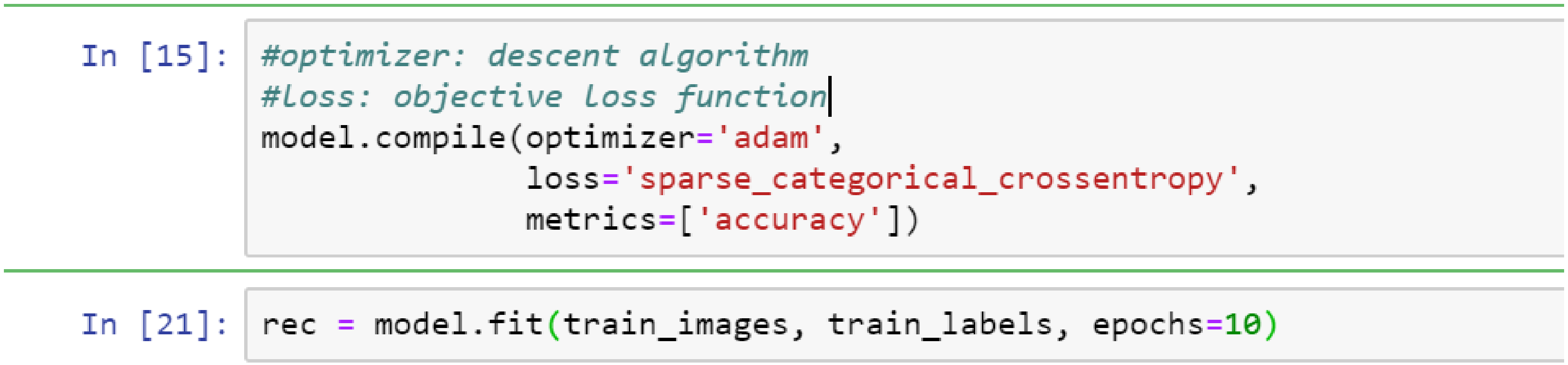
```
train_images = train_images / 255.0
test_images = test_images / 255.0
```

Build model



```
In [14]: model = keras.Sequential([
             keras.layers.Flatten(input_shape=(28, 28)),
             keras.layers.Dense(128, activation='relu'),
             keras.layers.Dense(10, activation='softmax')
         ])
```

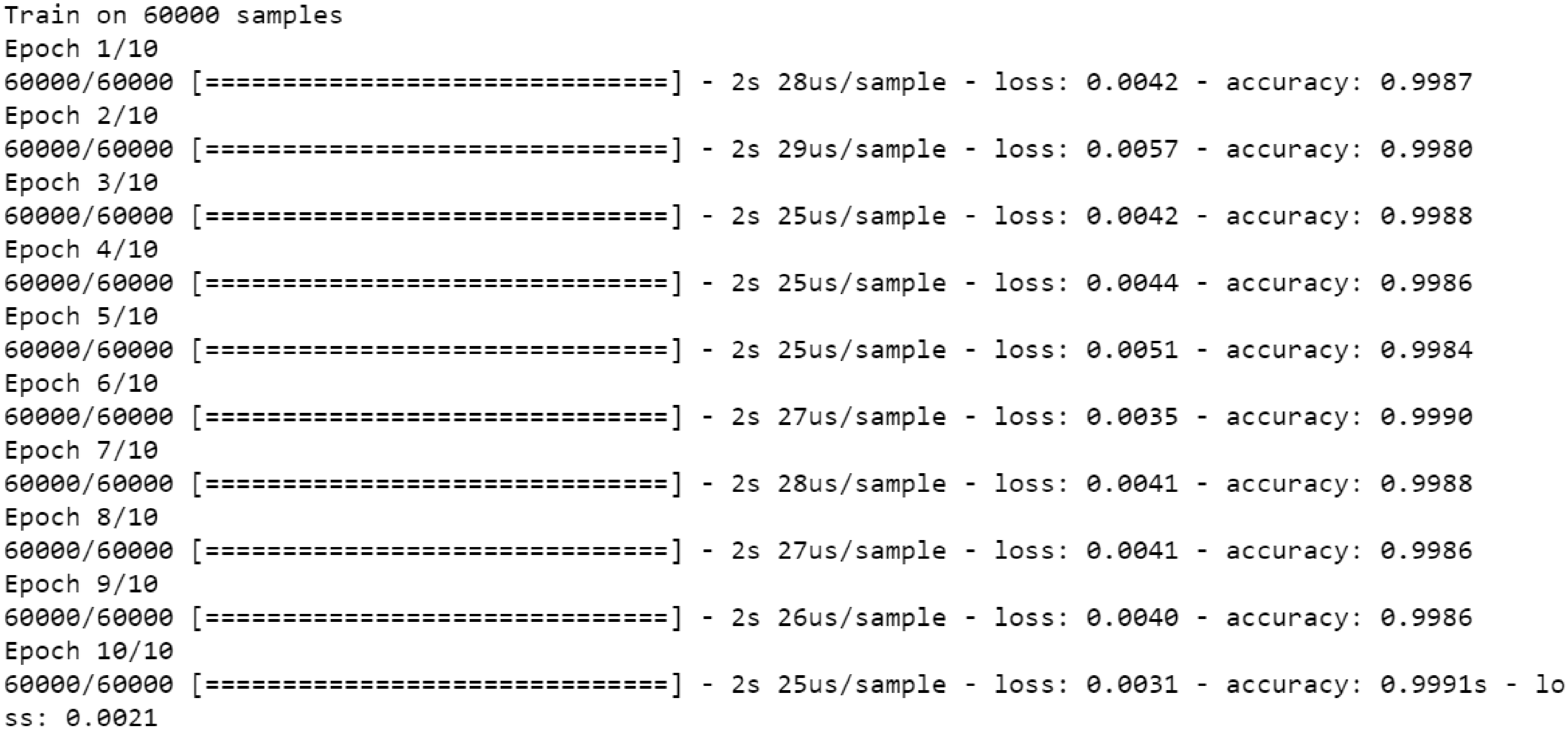
Training



```
In [15]: #optimizer: descent algorithm
         #loss: objective loss function
         model.compile(optimizer='adam',
                       loss='sparse_categorical_crossentropy',
                       metrics=['accuracy'])
```

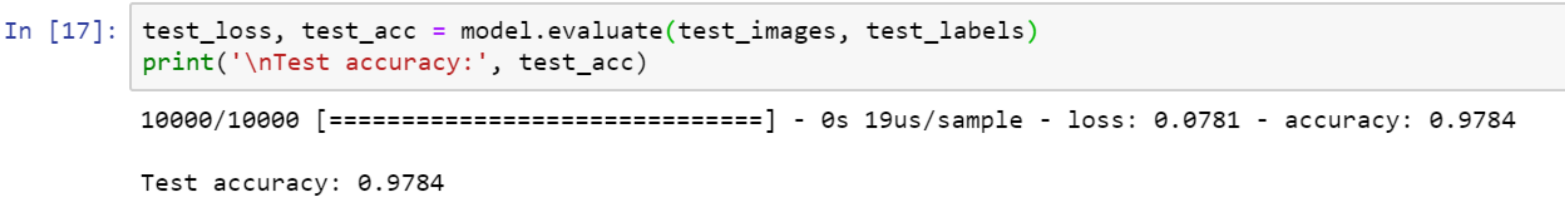
```
In [21]: rec = model.fit(train_images, train_labels, epochs=10)
```

Result



```
Train on 60000 samples
Epoch 1/10
60000/60000 [==============================] - 2s 28us/sample - loss: 0.0042 - accuracy: 0.9987
Epoch 2/10
60000/60000 [==============================] - 2s 29us/sample - loss: 0.0057 - accuracy: 0.9980
Epoch 3/10
60000/60000 [==============================] - 2s 25us/sample - loss: 0.0042 - accuracy: 0.9988
Epoch 4/10
60000/60000 [==============================] - 2s 25us/sample - loss: 0.0044 - accuracy: 0.9986
Epoch 5/10
60000/60000 [==============================] - 2s 25us/sample - loss: 0.0051 - accuracy: 0.9984
Epoch 6/10
60000/60000 [==============================] - 2s 27us/sample - loss: 0.0035 - accuracy: 0.9990
Epoch 7/10
60000/60000 [==============================] - 2s 28us/sample - loss: 0.0041 - accuracy: 0.9988
Epoch 8/10
60000/60000 [==============================] - 2s 27us/sample - loss: 0.0041 - accuracy: 0.9986
Epoch 9/10
60000/60000 [==============================] - 2s 26us/sample - loss: 0.0040 - accuracy: 0.9986
Epoch 10/10
60000/60000 [==============================] - 2s 25us/sample - loss: 0.0031 - accuracy: 0.9991s - lo
ss: 0.0021
```

Evaluate accuracy

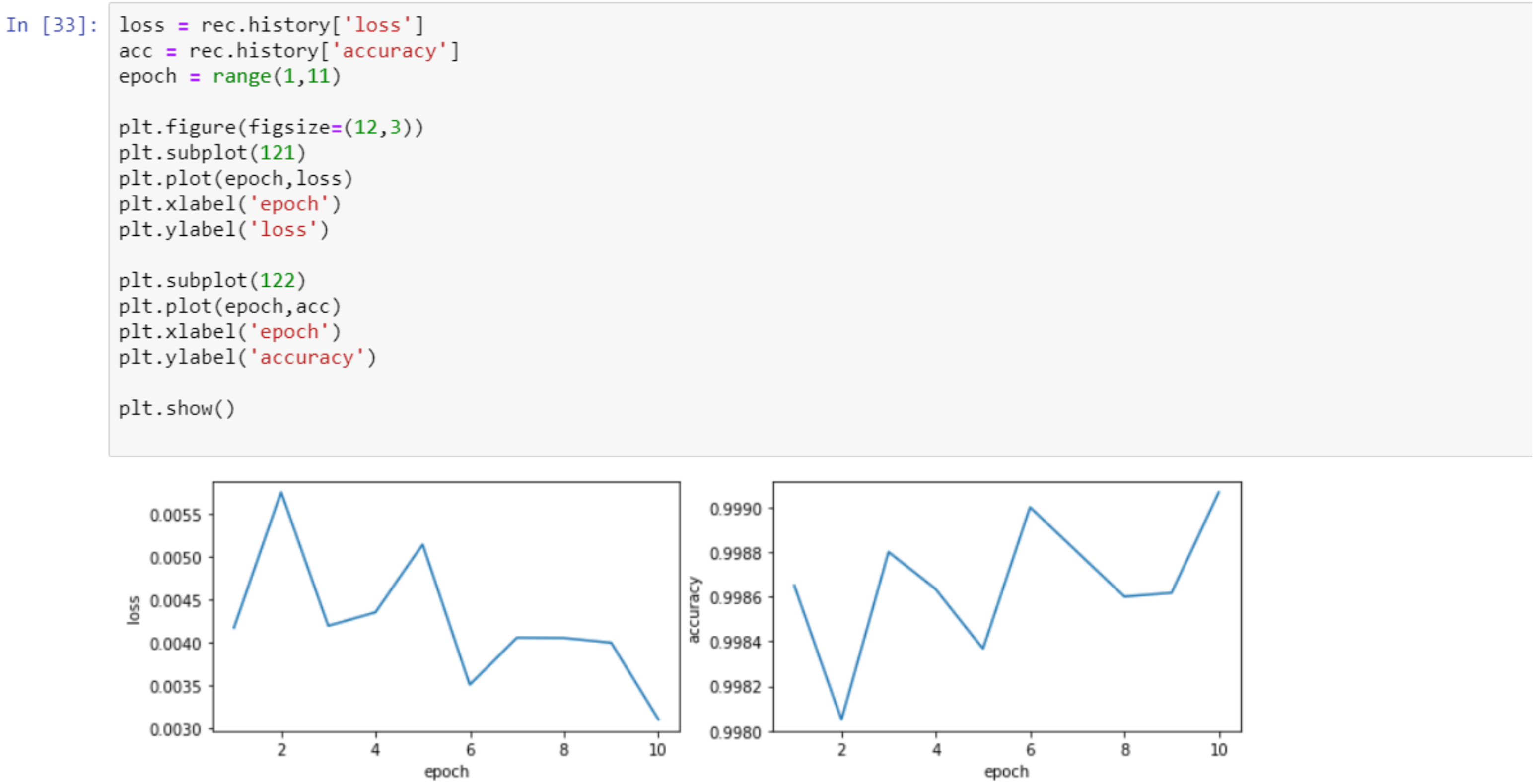


```
In [17]: test_loss, test_acc = model.evaluate(test_images, test_labels)
         print('\nTest accuracy:', test_acc)

         10000/10000 [==============================] - 0s 19us/sample - loss: 0.0781 - accuracy: 0.9784

         Test accuracy: 0.9784
```

Visualize



In [33]:

```
loss = rec.history['loss']
acc = rec.history['accuracy']
epoch = range(1,11)

plt.figure(figsize=(12,3))
plt.subplot(121)
plt.plot(epoch,loss)
plt.xlabel('epoch')
plt.ylabel('loss')

plt.subplot(122)
plt.plot(epoch,acc)
plt.xlabel('epoch')
plt.ylabel('accuracy')

plt.show()
```

References

- TensorFlow
- TensorFlow Hub
- TensorFlow tutorials with tf-2.0

Class assignment

- Please train a MLP (Multi-layer perceptron) to predict the class of input images in **Fashion Mnist dataset**, and the testing accuracy should be at least 85%.
- Turn in your work with the format of .ipynb , and please write some brief comments in your ipynb to illustrate your results.

Homework

- Please use the **Cifar-10 dataset** and what we taught in TA class to train a MLP model, and the testing accuracy should be at least 45%.
- You are encouraged to implement different methods to train your model.

  (EX: dropout or different optimizers)
- Turn in your work with the format of .ipynb , and please write some brief comments in your ipynb to illustrate your results.