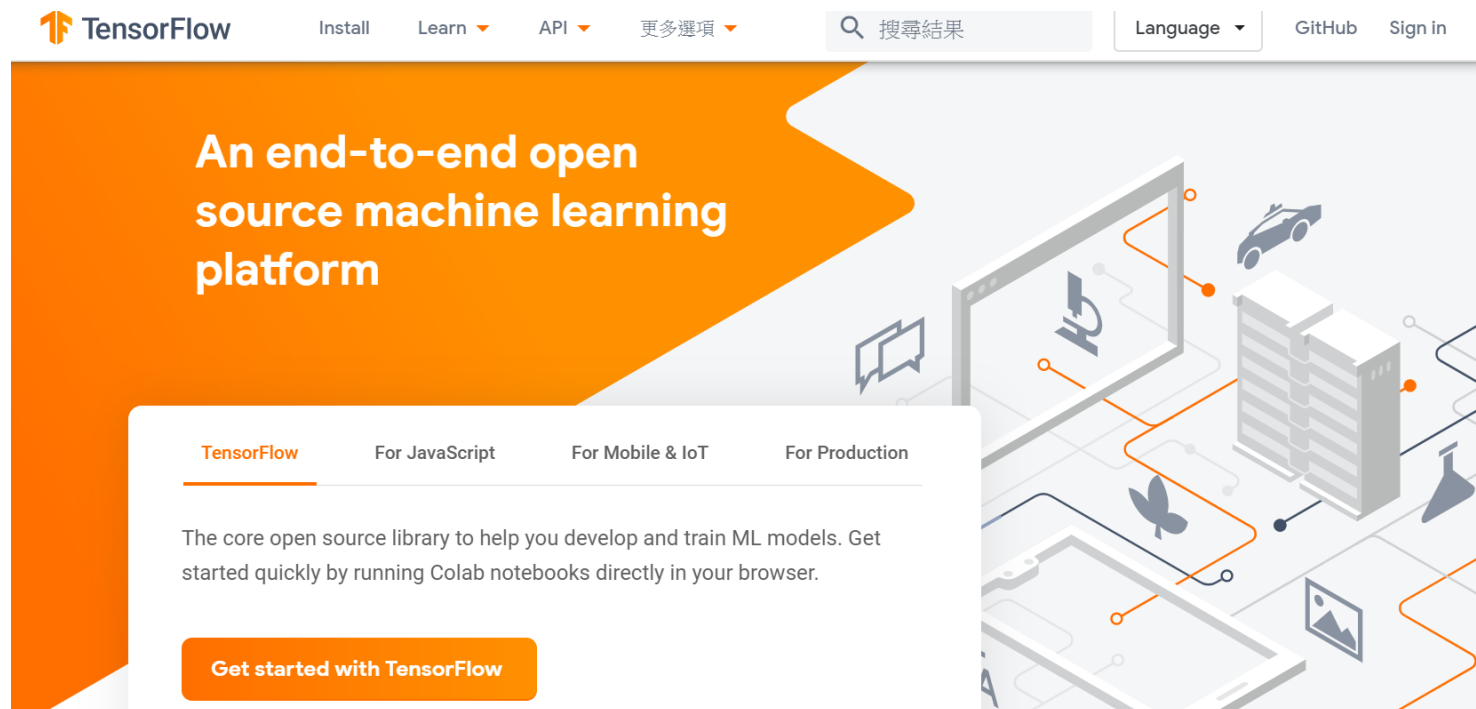


智慧化企業整合

Intelligent Integration of Enterprise

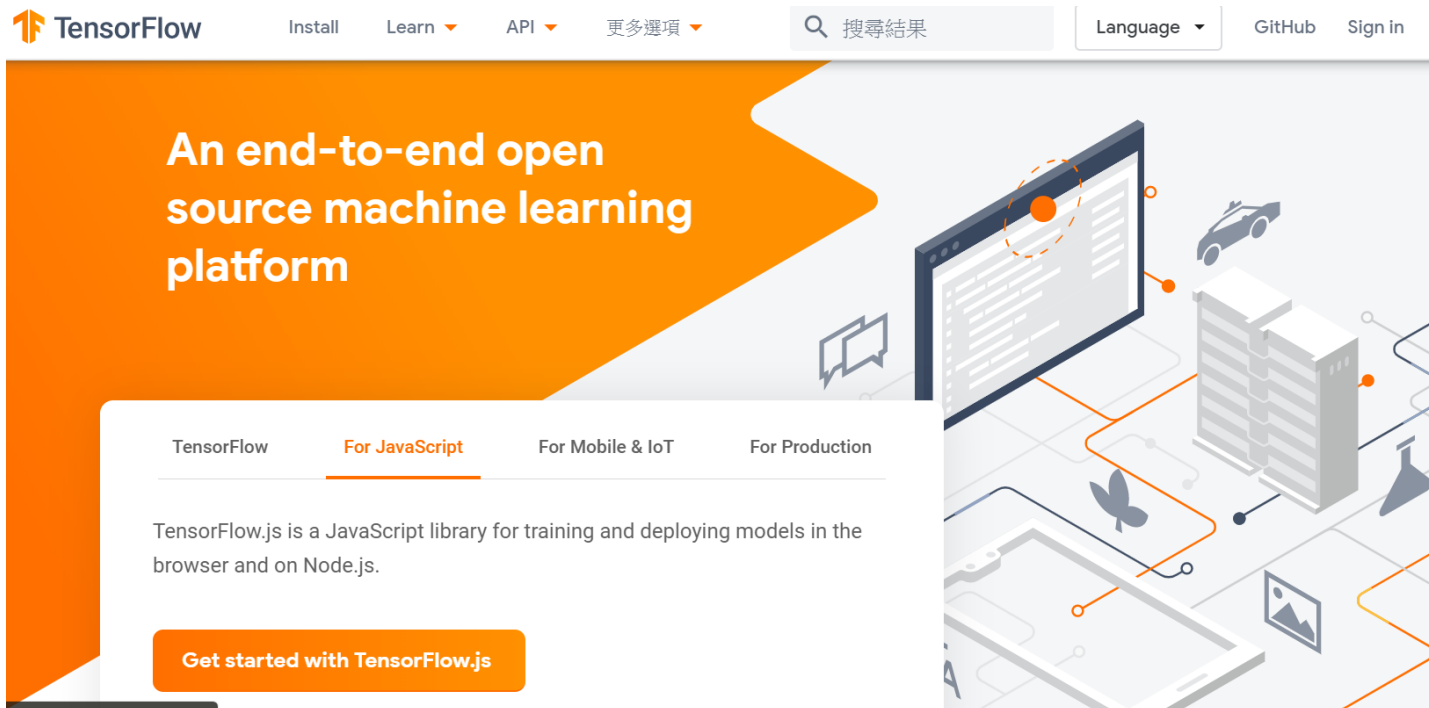
Introduction of TensorFlow

TensorFlow



The screenshot shows the TensorFlow website homepage. At the top left is the TensorFlow logo. The navigation bar includes links for 'Install', 'Learn', 'API', and '更多選項'. A search bar contains the text '搜尋結果'. On the right, there are links for 'Language', 'GitHub', and 'Sign in'. The main content area features a large orange arrow pointing right with the text 'An end-to-end open source machine learning platform'. Below this, there are four tabs: 'TensorFlow' (selected), 'For JavaScript', 'For Mobile & IoT', and 'For Production'. The 'TensorFlow' tab contains the text: 'The core open source library to help you develop and train ML models. Get started quickly by running Colab notebooks directly in your browser.' and a button labeled 'Get started with TensorFlow'. The background of the page is a light gray with a 3D-style illustration of a server rack, a car, a smartphone, and various data flow icons.

TensorFlow



The screenshot shows the TensorFlow website interface. At the top left is the TensorFlow logo. To its right are navigation links: "Install", "Learn" (with a dropdown arrow), "API" (with a dropdown arrow), and "更多選項" (with a dropdown arrow). A search bar contains the text "搜尋結果". To the right of the search bar are "Language" (with a dropdown arrow), "GitHub", and "Sign in".

The main content area features a large orange arrow pointing right, containing the text: "An end-to-end open source machine learning platform".

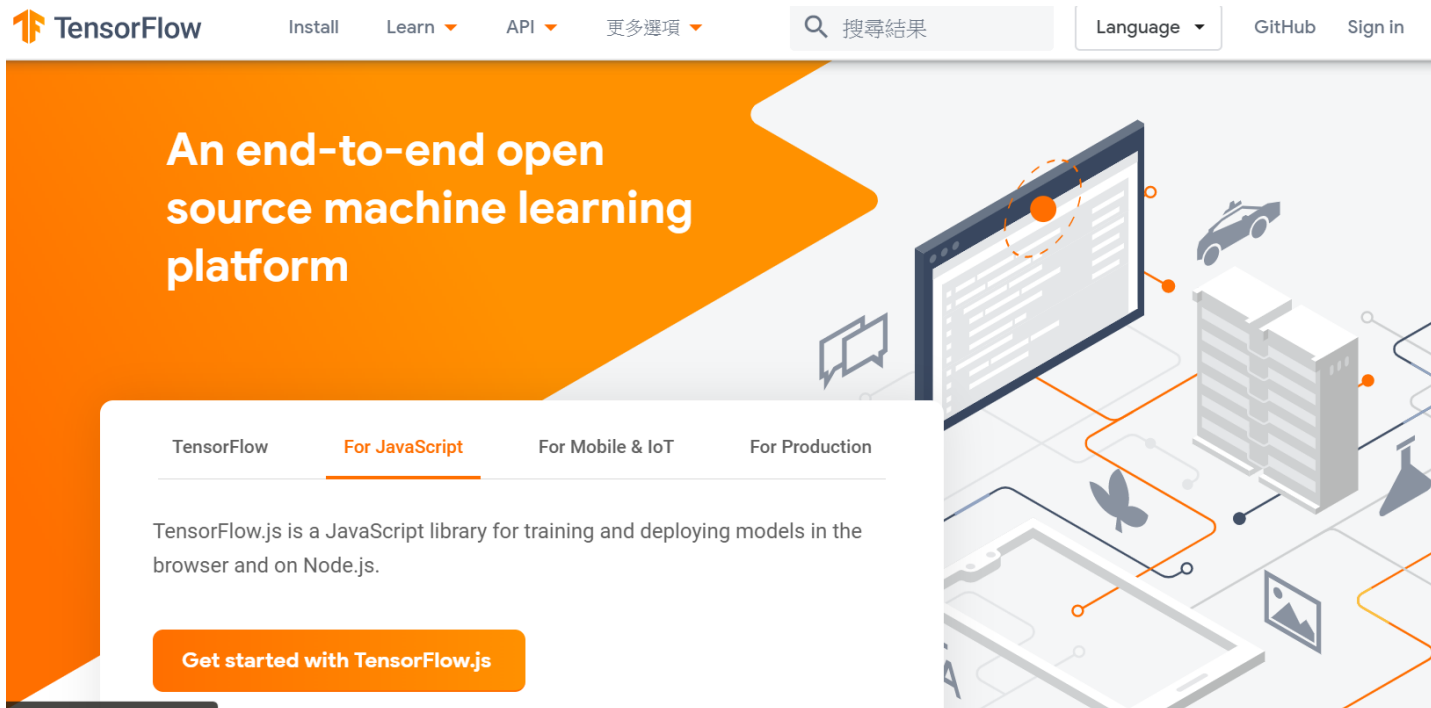
Below this, there is a white box with a navigation menu: "TensorFlow", "For JavaScript" (highlighted with an orange underline), "For Mobile & IoT", and "For Production".

Under the "For JavaScript" tab, the text reads: "TensorFlow.js is a JavaScript library for training and deploying models in the browser and on Node.js."

At the bottom of this white box is an orange button with the text: "Get started with TensorFlow.js".

The background of the website features a 3D-style illustration of a computer monitor, a server rack, a car, and a smartphone, all connected by orange lines representing data flow.

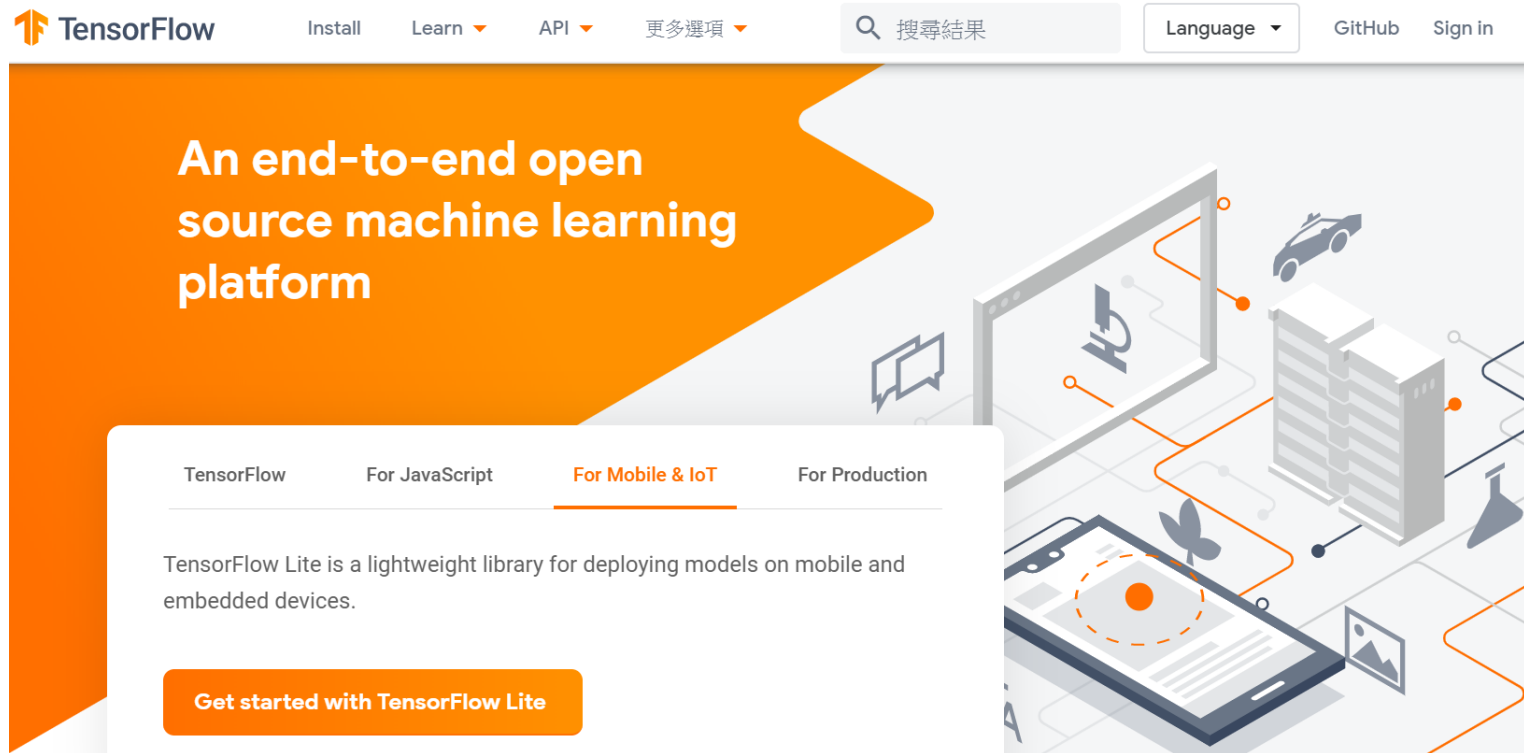
TensorFlow



The screenshot shows the TensorFlow website with the following elements:

- Navigation:** TensorFlow logo, Install, Learn (dropdown), API (dropdown), 更多選項 (dropdown), 搜尋結果 (Search Results), Language (dropdown), GitHub, Sign In.
- Hero Section:** An orange arrow-shaped graphic containing the text: "An end-to-end open source machine learning platform".
- Product Selection:** A white box with tabs for TensorFlow, **For JavaScript** (selected), For Mobile & IoT, and For Production.
- Description:** "TensorFlow.js is a JavaScript library for training and deploying models in the browser and on Node.js."
- Call to Action:** A prominent orange button labeled "Get started with TensorFlow.js".
- Background Illustration:** A stylized 3D illustration of a computer monitor, server racks, a car, and a smartphone, connected by orange lines representing data flow.

TensorFlow



The screenshot shows the TensorFlow website homepage. At the top left is the TensorFlow logo. The navigation bar includes links for "Install", "Learn", "API", and "更多選項". A search bar contains the text "搜尋結果". On the right, there are links for "Language", "GitHub", and "Sign in". The main content area features a large orange arrow pointing right with the text "An end-to-end open source machine learning platform". Below this, a white box contains a navigation menu with "TensorFlow", "For JavaScript", "For Mobile & IoT" (highlighted), and "For Production". The text below the menu reads: "TensorFlow Lite is a lightweight library for deploying models on mobile and embedded devices." At the bottom of this box is an orange button that says "Get started with TensorFlow Lite". The background of the page is a light gray with an illustration of a smartphone, a server rack, and a car, connected by orange lines representing data flow.

TensorFlow

Install Learn API 更多選項

搜尋結果

Language GitHub Sign in

An end-to-end open source machine learning platform

TensorFlow For JavaScript **For Mobile & IoT** For Production

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

Get started with TensorFlow Lite

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

Classification



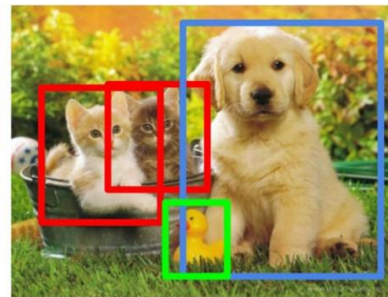
CAT

Classification
+ Localization



CAT

Object Detection



CAT, DOG, DUCK

Instance
Segmentation



CAT, DOG, DUCK

Single object

Multiple objects

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

label = 5



label = 0



label = 4



label = 1



label = 9



label = 2



label = 1



label = 3



label = 1



label = 4



label = 3



label = 5



label = 3



label = 6



label = 1



label = 7



label = 2



label = 8



label = 6



label = 9



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()

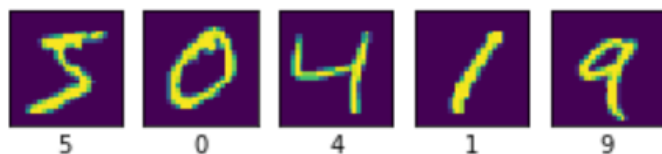
        print("training data: {}".format(train_images.shape))
        print("training labels {}".format(train_labels.shape))

        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

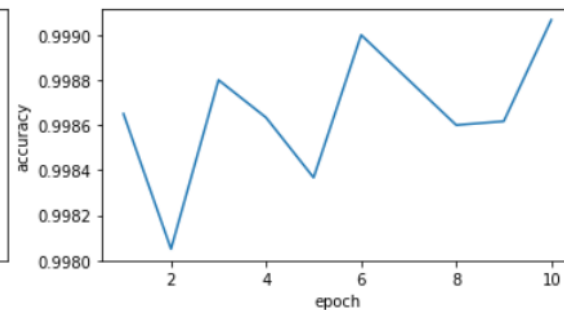
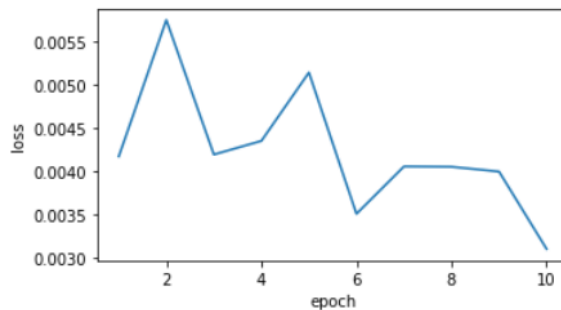
```
In [17]: test_loss, test_acc = model.evaluate(test_images, test_labels)
print('\nTest accuracy:', test_acc)
```

```
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.