[1]
```
import tensorflow as tf

import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

[2]
```
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.cifar10.load_data()
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170498071/170498071 ━━━━━━━━━━━━━━━━━━━━ 3s 0us/step
```

[3]
```
y_train = y_train.reshape(-1, 1)
```

[4]
```
print(f"x_train shape: {x_train.shape}")
print(f"y_train shape: {y_train.shape}")
```

```
x_train shape: (50000, 32, 32, 3)
y_train shape: (50000, 1)
```

[5]
```
print(x_train.min(), x_train.max())
print(x_test.min(), x_test.max())
```

```
0 255
0 255
```

[6]
```
x_train = x_train.astype("float32") / 255.0
x_test = x_test.astype("float32") / 255.0
```

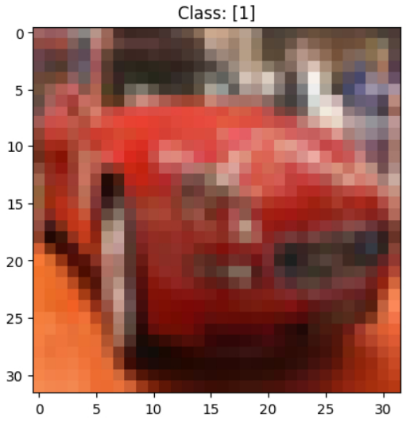
[7]
```
print(x_train.min(), x_train.max())
print(x_test.min(), x_test.max())
```

```
0.0 1.0
0.0 1.0
```

[8]
```
print(f"x_train shape: {x_train.shape}")
```

```
x_train shape: (50000, 32, 32, 3)
```

[9]
```
plt.imshow(x_train[5])
plt.title(f'Class: {y_train[5]}')  # แสดงคลาสของรูป
plt.show()
```



[18]
```
import tensorflow as tf
from keras import layers, models
from tensorflow.keras.callbacks import EarlyStopping


model = models.Sequential()
model.add(layers.InputLayer(shape=(32, 32,3)))

model.add(layers.Conv2D(256, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(256, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))


model.add(layers.BatchNormalization())
model.add(layers.Flatten())

model.add(layers.Dropout(0.3))

model.add(layers.Dense(512, activation="relu"))
model.add(layers.Dense(256, activation="relu"))
model.add(layers.Dense(128, activation="relu"))
model.add(layers.Dense(64, activation="relu"))
model.add(layers.Dense(64, activation="relu"))
model.add(layers.Dense(10, activation="softmax"))

model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.0005),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)
```

```
[19] model.summary()
```

Model: "sequential_2"

Layer (type)	Output Shape	Param #
conv2d_5 (Conv2D)	(None, 30, 30, 256)	7,168
max_pooling2d_5 (MaxPooling2D)	(None, 15, 15, 256)	0
conv2d_6 (Conv2D)	(None, 13, 13, 256)	590,080
max_pooling2d_6 (MaxPooling2D)	(None, 6, 6, 256)	0
conv2d_7 (Conv2D)	(None, 4, 4, 128)	295,040
max_pooling2d_7 (MaxPooling2D)	(None, 2, 2, 128)	0
batch_normalization_2 (BatchNormalization)	(None, 2, 2, 128)	512
flatten_2 (Flatten)	(None, 512)	0
dropout_2 (Dropout)	(None, 512)	0
dense_11 (Dense)	(None, 512)	262,656
dense_12 (Dense)	(None, 256)	131,328
dense_13 (Dense)	(None, 128)	32,896
dense_14 (Dense)	(None, 64)	8,256
dense_15 (Dense)	(None, 64)	4,160
dense_16 (Dense)	(None, 10)	650

```
Total params: 1,332,746 (5.08 MB)
Trainable params: 1,332,490 (5.08 MB)
Non-trainable params: 256 (1.00 KB)
```

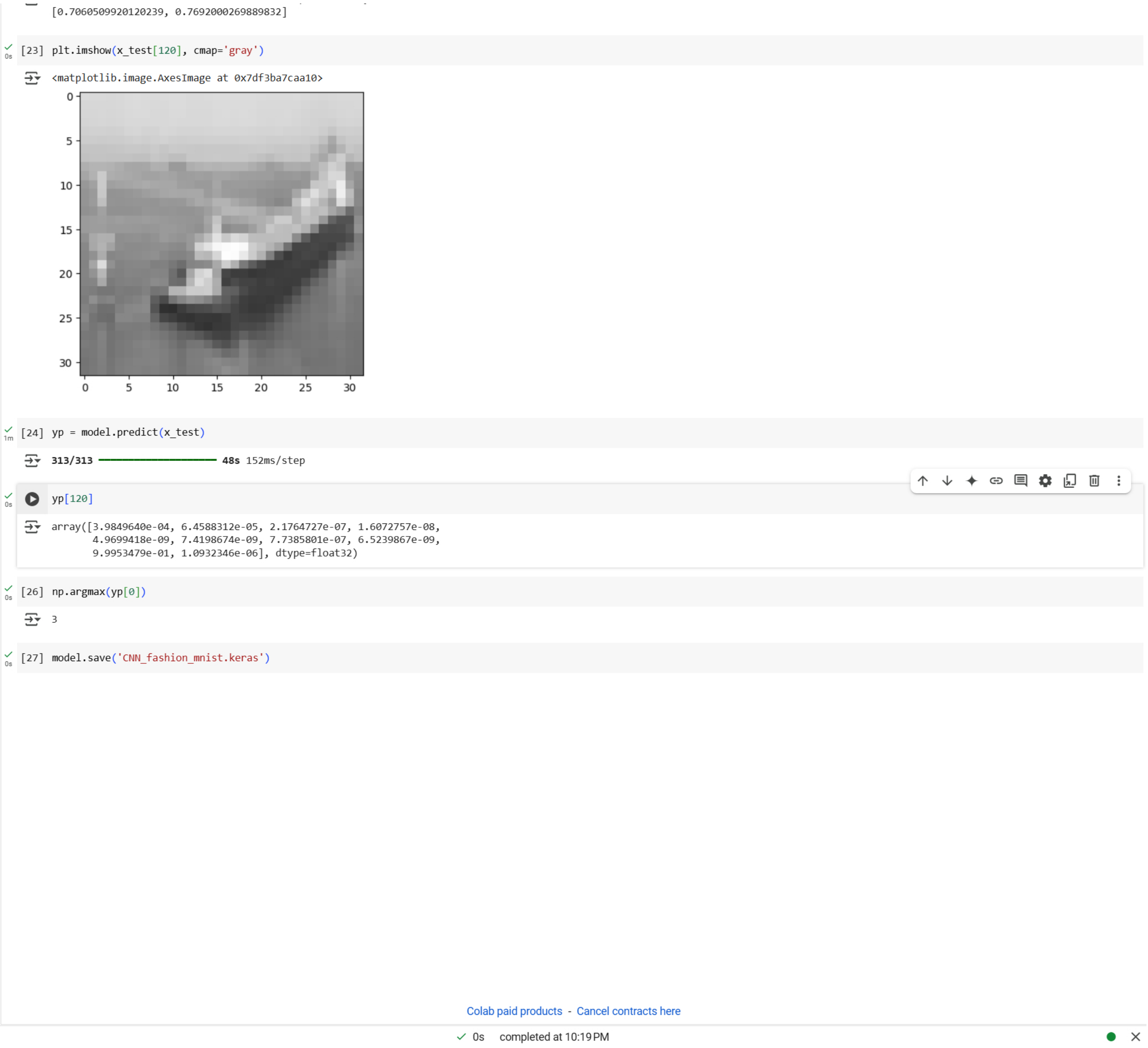
```
[20] model.compile(loss="sparse_categorical_crossentropy", # sparse_categorical_crossentropy
              optimizer="adam",
              metrics=["accuracy"])
```

```
[21] history = model.fit(x_train, y_train, epochs=150, validation_data=(x_train, y_train), callbacks=[early_stopping])
```

```
Epoch 1/150
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 1223s 780ms/step - accuracy: 0.3403 - loss: 1.7816 - val_accuracy: 0.4791 - val_loss: 1.5566
Epoch 2/150
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 1190s 759ms/step - accuracy: 0.5456 - loss: 1.2841 - val_accuracy: 0.5746 - val_loss: 1.1954
Epoch 3/150
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 1264s 809ms/step - accuracy: 0.6093 - loss: 1.1118 - val_accuracy: 0.1662 - val_loss: 4.7727
Epoch 4/150
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 1253s 802ms/step - accuracy: 0.6500 - loss: 1.0041 - val_accuracy: 0.6446 - val_loss: 1.0179
Epoch 5/150
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 1249s 799ms/step - accuracy: 0.6829 - loss: 0.9153 - val_accuracy: 0.7028 - val_loss: 0.8507
Epoch 6/150
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 1258s 784ms/step - accuracy: 0.6995 - loss: 0.8601 - val_accuracy: 0.7319 - val_loss: 0.7540
Epoch 7/150
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 1273s 778ms/step - accuracy: 0.7203 - loss: 0.7995 - val_accuracy: 0.7405 - val_loss: 0.7468
Epoch 8/150
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 1221s 777ms/step - accuracy: 0.7373 - loss: 0.7534 - val_accuracy: 0.7829 - val_loss: 0.6199
Epoch 9/150
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 1289s 824ms/step - accuracy: 0.7507 - loss: 0.7173 - val_accuracy: 0.7329 - val_loss: 0.7640
Epoch 10/150
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 1311s 804ms/step - accuracy: 0.7624 - loss: 0.6862 - val_accuracy: 0.7858 - val_loss: 0.6301
Epoch 11/150
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 1314s 825ms/step - accuracy: 0.7771 - loss: 0.6465 - val_accuracy: 0.8245 - val_loss: 0.5156
Epoch 12/150
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 1239s 759ms/step - accuracy: 0.7771 - loss: 0.6404 - val_accuracy: 0.7781 - val_loss: 0.6708
Epoch 13/150
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 1205s 748ms/step - accuracy: 0.7891 - loss: 0.6075 - val_accuracy: 0.8401 - val_loss: 0.4578
Epoch 14/150
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 1260s 773ms/step - accuracy: 0.7944 - loss: 0.5858 - val_accuracy: 0.8331 - val_loss: 0.4783
Epoch 15/150
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 1227s 776ms/step - accuracy: 0.8042 - loss: 0.5618 - val_accuracy: 0.8679 - val_loss: 0.3835
Epoch 16/150
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 1220s 775ms/step - accuracy: 0.8096 - loss: 0.5498 - val_accuracy: 0.8139 - val_loss: 0.5359
Epoch 17/150
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 1211s 768ms/step - accuracy: 0.8146 - loss: 0.5300 - val_accuracy: 0.8791 - val_loss: 0.3609
Epoch 18/150
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 1173s 751ms/step - accuracy: 0.8207 - loss: 0.5142 - val_accuracy: 0.8464 - val_loss: 0.4283
Epoch 19/150
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 1208s 773ms/step - accuracy: 0.8289 - loss: 0.4941 - val_accuracy: 0.8742 - val_loss: 0.3623
Epoch 20/150
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 1209s 765ms/step - accuracy: 0.8321 - loss: 0.4813 - val_accuracy: 0.8524 - val_loss: 0.4202
Epoch 21/150
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 1222s 764ms/step - accuracy: 0.8370 - loss: 0.4706 - val_accuracy: 0.7604 - val_loss: 0.7090
Epoch 22/150
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 1188s 743ms/step - accuracy: 0.8404 - loss: 0.4591 - val_accuracy: 0.9043 - val_loss: 0.2837
Epoch 23/150
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 1214s 776ms/step - accuracy: 0.8499 - loss: 0.4401 - val_accuracy: 0.8949 - val_loss: 0.3140
Epoch 24/150
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 1226s 784ms/step - accuracy: 0.8477 - loss: 0.4339 - val_accuracy: 0.8814 - val_loss: 0.3405
Epoch 25/150
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 1271s 777ms/step - accuracy: 0.8535 - loss: 0.4181 - val_accuracy: 0.8712 - val_loss: 0.3715
Epoch 26/150
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 1211s 770ms/step - accuracy: 0.8568 - loss: 0.4112 - val_accuracy: 0.8968 - val_loss: 0.3021
Epoch 27/150
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 1233s 777ms/step - accuracy: 0.8557 - loss: 0.4067 - val_accuracy: 0.8767 - val_loss: 0.3507
```

```
[22] model.evaluate(x_test, y_test)
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 67s 213ms/step - accuracy: 0.7730 - loss: 0.6956
```



```
[0.7060509920120239, 0.7692000269889832]
```

[23]
```
plt.imshow(x_test[120], cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x7df3ba7caa10>
```

0
5
10
15
20
25
30
0
5
10
15
20
25
30

[24]
```
yp = model.predict(x_test)
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 48s 152ms/step
```

```
yp[120]
```

```
array([3.9849640e-04, 6.4588312e-05, 2.1764727e-07, 1.6072757e-08,
       4.9699418e-09, 7.4198674e-09, 7.7385801e-07, 6.5239867e-09,
       9.9953479e-01, 1.0932346e-06], dtype=float32)
```

[26]
```
np.argmax(yp[0])
```

```
3
```

[27]
```
model.save('CNN_fashion_mnist.keras')
```

Colab paid products - Cancel contracts here
✓ 0s completed at 10:19 PM