Homework 3: Image classification using Convolutional Networks (CNN)
Train a convolutional neural network to classify images from the CIFAR10 dataset.

1. โหลดข้อมูลและตรวจสอบข้อมูลเบื้องต้น

```
[1] import numpy as np
    import tensorflow as tf
    import matplotlib.pyplot as plt
```

```
[2] cifar = tf.keras.datasets.cifar10
```

```
[3] (X_train, y_train), (X_test, y_test) = cifar.load_data()  # Load dataset
```

จะได้ข้อมูลสำหรับ train และ test แยกกันเรียบร้อยแล้ว ลองตรวจสอบมิติและจำนวนข้อมูล

```
[4] print(X_train.shape) # ตรวจสอบจำนวนข้อมูลภาพ (feature) สำหรับ train
    print(X_test.shape) # test

    (50000, 32, 32, 3)
    (10000, 32, 32, 3)
```

```
[5] print(y_train.shape) # class
    print(y_test.shape)

    (50000, 1)
    (10000, 1)
```

```
[6] # ตรวจดูค่าต่ำสุดและสูงสุดของ Pixel (ค่า Min/Max ใน matrix X_train)
    # ค่าใน matrix คือค่าระดับความสว่างของแต่ละ pixe -> 0=มืดสุด, 255=สว่างสุด
    np.min(X_train), np.max(X_train)

    (0, 255)
```

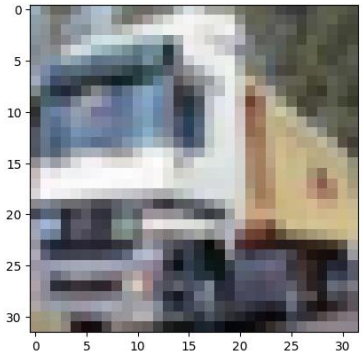
```
[7] np.unique(y_train)

    array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9], dtype=uint8)
```

```
[8] np.isnan(X_train).sum() # ตรวจสอบข้อมูลว่ามี missing data หรือไม่

    0
```

- จากโค้ด [4] จะเห็นได้ว่ามีข้อมูลภาพสี (3 channel) ขนาด 32x32 Pixel สำหรับ Train (X_train) จำนวน 50,000 ภาพ ส่วนข้อมูลสำหรับ test มีจำนวน 10,000 ภาพ
- ข้อมูลภาพเป็นโครงสร้างเมทริกซ์ที่เก็บค่าของ Pixel ระหว่าง 0-255
- y_train, y_test คือ label มีค่า 0-9 (10 class)
- ไม่มี missing data

2. Visualization

```
[9]  plt.imshow(X_train[1])
     plt.show()
```



```
[10] def visual_multi(i):
       nplots = 40
       fig = plt.figure(figsize=(8, 4))
       for j in range(nplots):
         plt.subplot(4, 10, j+1)
         plt.imshow(X_train[i+j])
         plt.title(y_train[i+j])
         plt.axis('off')
         plt.xticks([]); plt.yticks([])
       plt.show()
```

```
[11] visual_multi(0) # แสดงตั้งแต่ index 0
```

3. เตรียมข้อมูลเพื่อ train และ test

3.1) จัดรูปแบบโครงสร้าง:

```
[12] print(X_train.shape) # ตรวจสอบจำนวนข้อมูลภาพ (feature) สำหรับ train
     print(X_test.shape) # test

     (50000, 32, 32, 3)
     (10000, 32, 32, 3)
```

เนื่องจากที่เราตรวจสอบข้อมูล พบว่าข้อมูลที่ load มามีการระบุว่ามี 3 channel และรูปแบบโครงสร้างข้อมูลตรงกับข้อกำหนดของ convolution tensorflow (samples x W x H x channel) จึงไม่ต้องทำการจัดรูปแบบโครงสร้างใหม่

3.2) Normalization:

```
[13] X_train = X_train.astype('float32')/255.0
     X_test = X_test.astype('float32')/255.0
```

```
[14] np.min(X_train), np.max(X_train)

     (0.0, 1.0)
```

4. สร้าง model

```
[15] # กำหนดค่าจำนวนคลาสที่จะให้ model จำแนก (ใช้ numpy นับว่ามีกี่คลาส)
     num_classes = len(np.unique(y_train))
     num_classes

     10
```

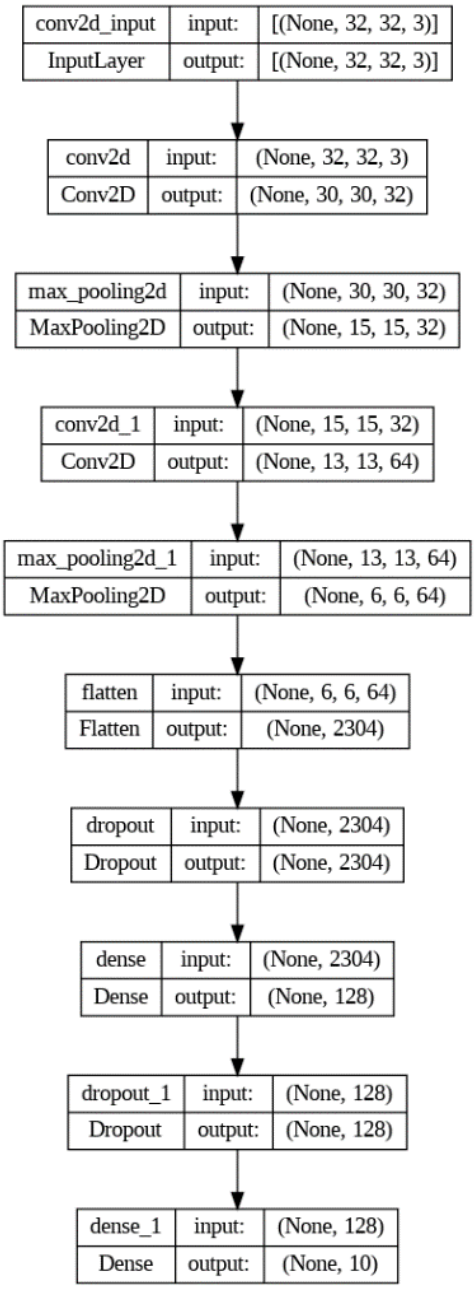
```
[16] # กำหนดมิติ input (ให้ code หาค่าให้)
     in_shape = X_train.shape[1:]
     in_shape

     (32, 32, 3)
```

```
[17] from tensorflow.keras import Sequential
     from tensorflow.keras.layers import Dense, Conv2D, MaxPool2D, Flatten, Dropout
```

```
[19] model = Sequential()
     model.add(Conv2D(32, (3, 3), activation='relu', input_shape=in_shape)) # ชั้นแรก
     model.add(MaxPool2D((2, 2)))

     model.add(Conv2D(64, (3, 3), activation='relu')) # ชั้นที่ 2
     model.add(MaxPool2D((2, 2)))

     model.add(Flatten())
     model.add(Dropout(0.5))

     model.add(Dense(128, activation='relu')) # ชั้นที่ 3
     model.add(Dropout(0.5))

     model.add(Dense(num_classes, activation='softmax'))
```

โครงสร้างของ Model: มีโครงสร้าง Convolution + MaxPooling 3 ชั้น ดังนี้

- ชั้นแรก Convolution (Conv2D) + MaxPooling (MaxPool) มี Kernel=32 รับ input ขนาด 32x32x3 (ภาพสี 3 channel) โดยกำหนดให้ชั้น Conv2D ใช้ activation ReLU (ประมวลผลตัดสินก่อนส่งให้ชั้นถัดไป)
- ชั้นที่ 2 Conv2D และ MaxPool มี kernel=64
- ชั้นที่ 3 Conv2D และ MaxPool มี kernel=128
- มีการใช้ dropout สำหรับลด overfitting



5. Compile และ train Model

```
[22] model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
     tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir='log')
```

```
[23] import time
     start = time.time()

     history = model.fit(X_train, y_train,
                         epochs=300,
                         batch_size=256,
                         verbose=1,
                         validation_split=0.1,
                         callbacks=[tensorboard_callback])

     end = time.time()
     print("Time Taken: {:.2f} minutes".format((end-start)/60))
```

```
Epoch 1/300
176/176 [==============================] - 14s 14ms/step - loss: 1.8123 - accuracy: 0.3298 - va
Epoch 2/300
176/176 [==============================] - 1s 8ms/step - loss: 1.4901 - accuracy: 0.4594 - val_
Epoch 3/300
176/176 [==============================] - 2s 9ms/step - loss: 1.3858 - accuracy: 0.5024 - val_
Epoch 4/300
176/176 [==============================] - 1s 8ms/step - loss: 1.3073 - accuracy: 0.5331 - val
```

```
Epoch 299/300
176/176 [==============================] - 2s 10ms/step - loss: 0.5342 - accuracy: 0.8069 - val_
Epoch 300/300
176/176 [==============================] - 2s 9ms/step - loss: 0.5263 - accuracy: 0.8104 - val_
Time Taken: 8.43 minutes
```

6. Evaluate Model

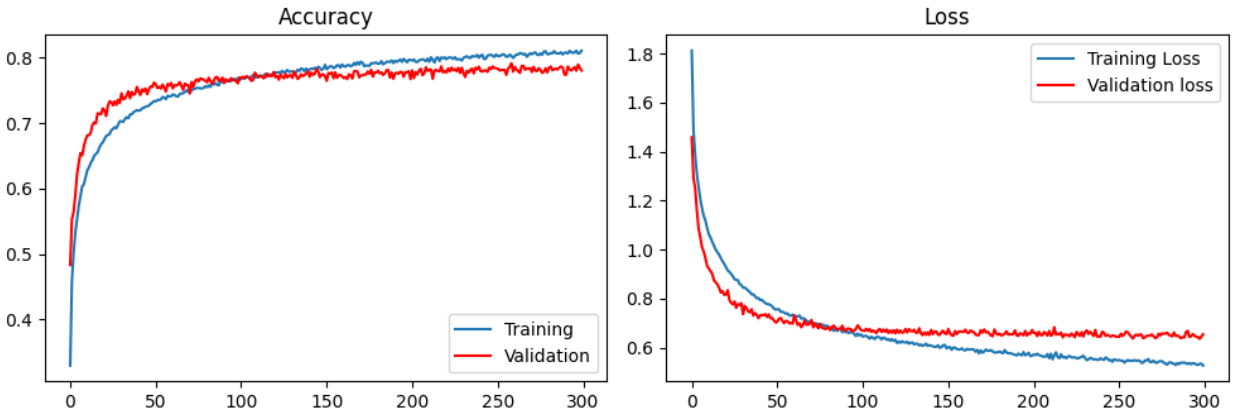
```
[25] score = model.evaluate(X_test, y_test, verbose=0)
     print('Accuracy (test set): {:.3f}'.format(score[1]))
```

```
Accuracy (test set): 0.774
```

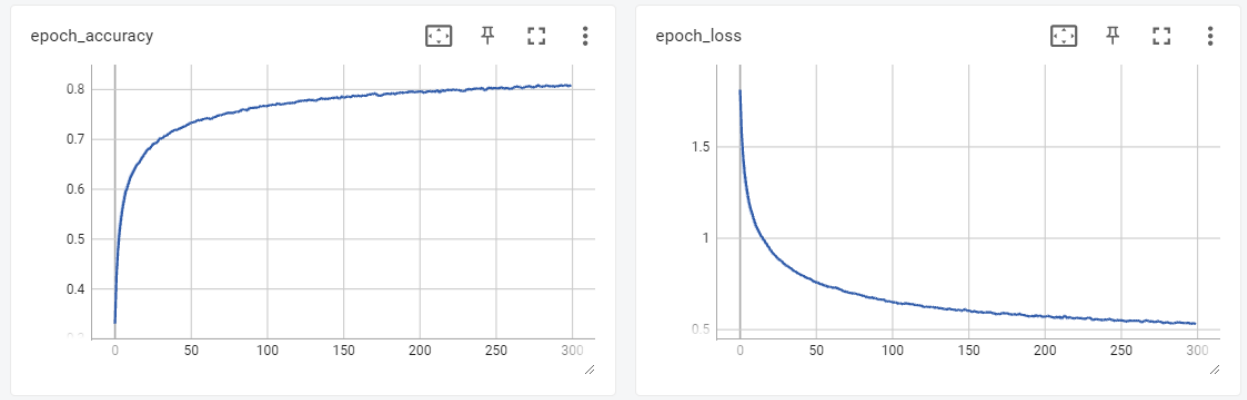
```
[26] score = model.evaluate(X_train, y_train, verbose=0)
     print('Accuracy (train set): {:.3f}'.format(score[1]))
```

```
Accuracy (train set): 0.942
```

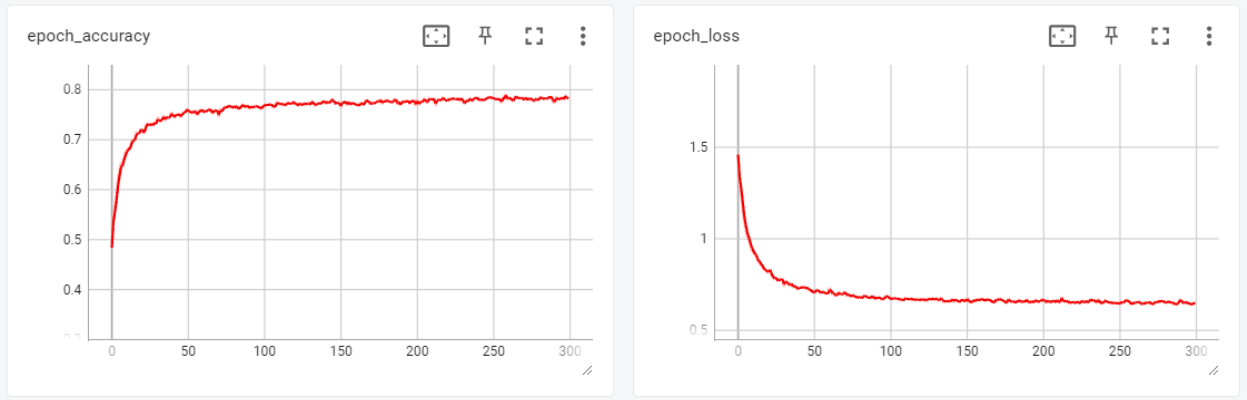
Graph Results: ซึ่งจากกราฟก็จะเห็นได้ว่าค่า Accuracy และ Loss ของ Validation Set ค่อนข้างน่าพอใจ เทรนด้วยจำนวน epochs ที่เหมาะสมจนค่าค่อนข้างคงที่ และจากกราฟก็จะเห็นได้ว่าโมเดลไม่เกิด Overfitting



เปรียบเทียบ Accuracy และ Loss ระหว่าง Training set และ Validation set



Accuracy และ Loss ของ Train set



Accuracy และ Loss ของ Validation set