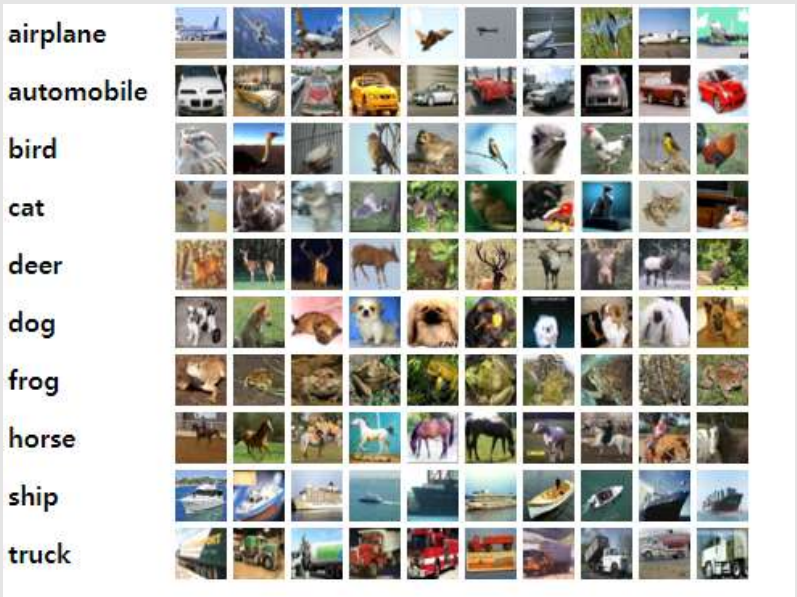
13주. Keras CNN			
학번	32153180	이름	이상민

Q1 (10점) CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class



- CIFAR-10 dataset 에 대해 CNN structure를 설계하고 모델을 개발한후 테스트 결과를 제시하시오
  (train accuracy 와 test accuracy를 제시)
  * hidden layer 의 수는 3개 이상, layer별 노드 수 및 기타 매개변수는 각자 정한다.
  * CIFAR-10 데이터셋을 읽어서 train, test set 을 준비하는 코드

```
from keras.datasets import cifar10

# load dataset
(X_train, y_train), (X_test, y_test) = cifar10.load_data()
y_train = np_utils.to_categorical(y_train, nb_classes)
y_test = np_utils.to_categorical(y_test, nb_classes)
```

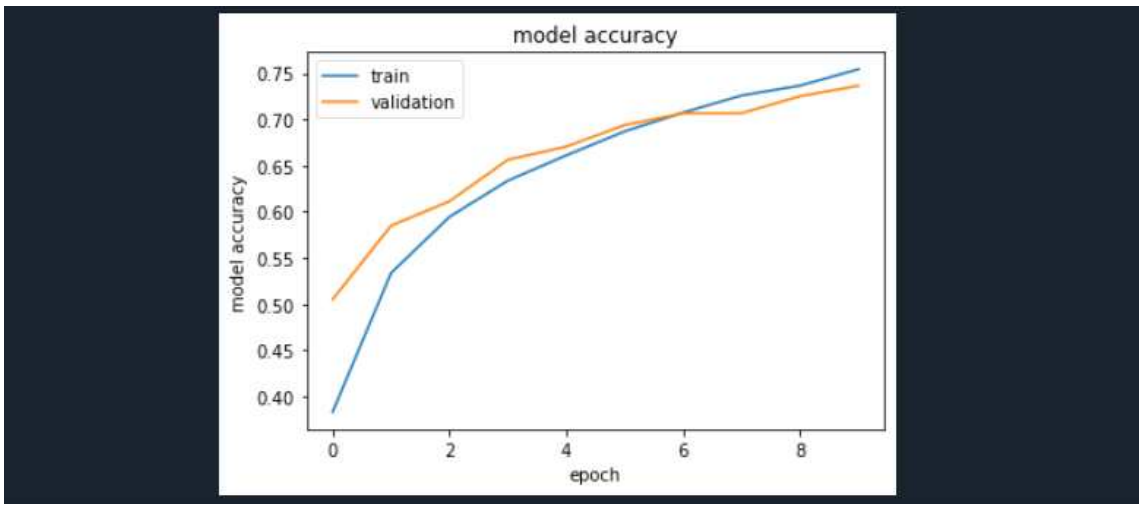
Source code :

```
# library
from keras.datasets import cifar10
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from keras.layers import Flatten
from keras.layers.convolutional import Convolution2D
from keras.layers.convolutional import MaxPooling2D
from keras.utils import np_utils
import numpy as np
import matplotlib.pyplot as plt

# define image size (32 X 32)
rows, cols = 32, 32

# load dataset and one hot encoded
(X_train, y_train), (X_test, y_test) = cifar10.load_data()
X_train, X_test = X_train / 255.0, X_test / 255.0
X_train = X_train.reshape(X_train.shape[0], rows, cols, 3)
X_test = X_test.reshape(X_test.shape[0], rows, cols, 3)

y_train = np_utils.to_categorical(y_train, num_classes)
y_test = np_utils.to_categorical(y_test, num_classes)
```

```
# create CNN model function
def cnn_model():
    model = Sequential()
    model.add(Convolution2D(32, kernel_size=(3, 3),
                        strides=(1, 1),
                        input_shape=(rows, cols, 3),
                        activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Convolution2D(64, kernel_size=(3, 3),
                        activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Convolution2D(128, kernel_size=(3, 3),
                        activation='relu'))
    model.add(Dropout(0.4))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Flatten())
    model.add(Dense(512, activation='relu'))
    model.add(Dense(128, activation='relu'))
    model.add(Dense(num_classes, activation='softmax'))

    model.compile(loss='categorical_crossentropy',
                optimizer='adam',
                metrics=['accuracy'])
    return model

model = cnn_model()

disp = model.fit(X_train, y_train,
                validation_data=(X_test, y_test),
                epochs=10, batch_size=200, verbose=1)

train_score = model.evaluate(X_train, y_train, verbose=0)
test_score = model.evaluate(X_test, y_test, verbose=0)
print('Train loss : {}'.format(train_score[0]))
print('Train accuracy : {}'.format(train_score[1]))
print('Test loss : {}'.format(test_score[0]))
print('Test accuracy : {}'.format(test_score[1]))
```

```
plt.plot(disp.history['accuracy'])
plt.plot(disp.history['val_accuracy'])
plt.title('model accuracy')
plt.xlabel('epoch')
plt.ylabel('model accuracy')
plt.legend(['train', 'validation'], loc='upper left')
plt.show()
```

실행화면 캡쳐:



① 다음과 같은 모델 training 실행화면 (맨 뒷부분 10줄 정도만)

```
Epoch 6/10
50000/50000 [==============================] - 75s 2ms/step - loss: 0.8923 - accuracy: 0.6871 - val_loss:
0.9055 - val_accuracy: 0.6939
Epoch 7/10
50000/50000 [==============================] - 74s 1ms/step - loss: 0.8285 - accuracy: 0.7074 - val_loss:
0.8739 - val_accuracy: 0.7066
Epoch 8/10
50000/50000 [==============================] - 74s 1ms/step - loss: 0.7841 - accuracy: 0.7259 - val_loss:
0.8507 - val_accuracy: 0.7067
Epoch 9/10
50000/50000 [==============================] - 74s 1ms/step - loss: 0.7515 - accuracy: 0.7366 - val_loss:
0.8073 - val_accuracy: 0.7251
Epoch 10/10
50000/50000 [==============================] - 74s 1ms/step - loss: 0.6966 - accuracy: 0.7543 - val_loss:
0.7953 - val_accuracy: 0.7365
```

② train accuracy 와 test accuracy 출력 부분

```
In [41]: print('Train loss : {}'.format(train_score[0]))
    ...: print('Train accuracy : {}'.format(train_score[1]))
    ...: print('Test loss : {}'.format(test_score[0]))
    ...: print('Test accuracy : {}'.format(test_score[1]))
Train loss : 0.6509400594711303
Train accuracy : 0.8050000071525574
Test loss : 0.7952527019500732
Test accuracy : 0.7365000247955322
```