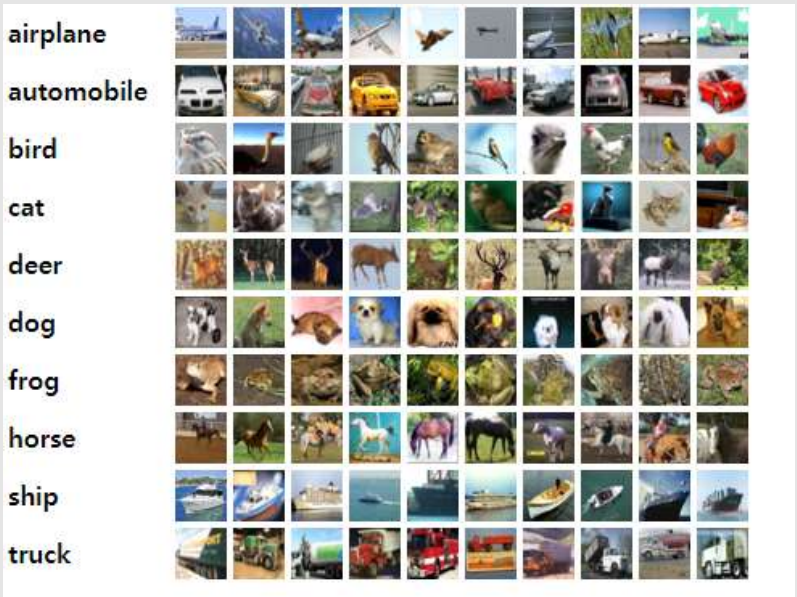
13주. Keras CNN			
학번	32152339	이름	송준영

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 노드수 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 :

```
// source code 의 폰트는 Courier10 BT Bold으로 하시오
#데이터 로드 및 가공
nb_classes=10
(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)

X_train.shape,X_test.shape

X_train,X_test = X_train/255.0 , X_test/255.0

seed=32152339
np.random.seed(seed)

#모델 선언
def cnn_model():
    model = Sequential()
    model.add(Convolution2D(64, 3, 3,
                        border_mode='same',
                        activation='relu',
                        input_shape=(32, 32,3)))
    model.add(Dropout(0.5))
    model.add(Convolution2D(64,                 3,                  3,
activation='relu',border_mode='same'))
    model.add(Dropout(0.5))
    model.add(MaxPooling2D(pool_size=(2, 2)))

    model.add(Convolution2D(128,                3,                  3,
activation='relu',border_mode='same'))
    model.add(Dropout(0.5))
    model.add(MaxPooling2D(pool_size=(2, 2)))

    model.add(Convolution2D(256,                3,                  3,
activation='relu',border_mode='same'))
    model.add(Dropout(0.5))
    model.add(MaxPooling2D(pool_size=(2, 2)))

    #DNN
    model.add(Flatten())
    model.add(Dense(512, activation='relu'))

    model.add(Dense(512, activation='relu'))

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

```
    model.compile(loss='categorical_crossentropy',
                  optimizer='adadelta',
                  metrics=['accuracy'])

    return model

model = cnn_model()
es = EarlyStopping(monitor='val_loss', min_delta=0.001, patience=3,
                   verbose=1,         mode='min',         baseline=None,
restore_best_weights=True)
#모델 적합
disp = model.fit(X_train,y_train,
                 validation_data=(X_test,y_test),
                 nb_epoch=20,
                 batch_size=200,
                 verbose=1,
                 callbacks=[es])

#모델 평가
train_scores = model.evaluate(X_train,y_train,verbose=0)
test_scores = model.evaluate(X_test,y_test,verbose=0)
print(f'train accuracy : {train_scores[1]}')
print(f'test accuracy : {test_scores[1]}')
```

실행화면 캡쳐:

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

```
Train on 50000 samples, validate on 10000 samples
Epoch 1/20
50000/50000 [==============================] - 117s 2ms/step - loss: 2.0852 - acc: 0.2370 - val_loss: 2.0691 - val_acc: 0.3270
Epoch 2/20
50000/50000 [==============================] - 286s 6ms/step - loss: 1.5978 - acc: 0.4268 - val_loss: 1.7677 - val_acc: 0.4206
Epoch 3/20
50000/50000 [==============================] - 286s 6ms/step - loss: 1.3634 - acc: 0.5122 - val_loss: 1.6488 - val_acc: 0.4469
Epoch 4/20
50000/50000 [==============================] - 286s 6ms/step - loss: 1.1766 - acc: 0.5821 - val_loss: 1.3841 - val_acc: 0.5985
Epoch 5/20
50000/50000 [==============================] - 286s 6ms/step - loss: 1.0292 - acc: 0.6385 - val_loss: 1.4029 - val_acc: 0.5205
Epoch 6/20
50000/50000 [==============================] - 286s 6ms/step - loss: 0.8955 - acc: 0.6847 - val_loss: 1.1836 - val_acc: 0.6026
Epoch 7/20
50000/50000 [==============================] - 286s 6ms/step - loss: 0.8080 - acc: 0.7171 - val_loss: 1.0667 - val_acc: 0.6714
Epoch 8/20
50000/50000 [==============================] - 286s 6ms/step - loss: 0.7234 - acc: 0.7474 - val_loss: 1.0653 - val_acc: 0.6667
Epoch 9/20
50000/50000 [==============================] - 286s 6ms/step - loss: 0.6476 - acc: 0.7738 - val_loss: 0.9590 - val_acc: 0.6846
Epoch 10/20
50000/50000 [==============================] - 286s 6ms/step - loss: 0.5910 - acc: 0.7920 - val_loss: 0.8809 - val_acc: 0.7366
Epoch 11/20
50000/50000 [==============================] - 286s 6ms/step - loss: 0.5322 - acc: 0.8138 - val_loss: 0.8992 - val_acc: 0.7186
Epoch 12/20
50000/50000 [==============================] - 286s 6ms/step - loss: 0.4795 - acc: 0.8296 - val_loss: 0.8086 - val_acc: 0.7561
Epoch 13/20
50000/50000 [==============================] - 286s 6ms/step - loss: 0.4334 - acc: 0.8473 - val_loss: 0.8439 - val_acc: 0.7270
Epoch 14/20
50000/50000 [==============================] - 286s 6ms/step - loss: 0.3885 - acc: 0.8628 - val_loss: 0.7869 - val_acc: 0.7413
Epoch 15/20
50000/50000 [==============================] - 286s 6ms/step - loss: 0.3446 - acc: 0.8772 - val_loss: 0.7565 - val_acc: 0.7510
Epoch 16/20
50000/50000 [==============================] - 286s 6ms/step - loss: 0.3119 - acc: 0.8876 - val_loss: 0.7455 - val_acc: 0.7566
Epoch 17/20
50000/50000 [==============================] - 286s 6ms/step - loss: 0.2856 - acc: 0.8986 - val_loss: 0.7706 - val_acc: 0.7511
Epoch 18/20
50000/50000 [==============================] - 286s 6ms/step - loss: 0.2598 - acc: 0.9080 - val_loss: 0.7340 - val_acc: 0.7544
Epoch 19/20
50000/50000 [==============================] - 286s 6ms/step - loss: 0.2335 - acc: 0.9182 - val_loss: 0.7315 - val_acc: 0.7593
Epoch 20/20
50000/50000 [==============================] - 286s 6ms/step - loss: 0.2146 - acc: 0.9246 - val_loss: 0.7422 - val_acc: 0.7517
```

② train accuracy 와 test accuracy 출력 부분

```
In [22]: train_scores = model.evaluate(X_train,y_train,verbose=0)
    ...: test_scores = model.evaluate(X_test,y_test,verbose=0)
    ...: print(f'train accuracy : {train_scores[1]}')
    ...: print(f'test accuracy : {test_scores[1]}')
train accuracy : 0.93212
test accuracy : 0.7517
```

평가기준 :

프로그램의 정상적으로 실행여부 : 5점

test accuracy : 5점