# 1번

In [1]:

```python
import tensorflow as tf
from tensorflow import keras
from keras import models
from keras import layers

import numpy as np
import matplotlib.pyplot as plt

cifar10 = keras.datasets.cifar10

(train_images, train_labels), (test_images, test_labels) = cifar10.load_data()

train_labels = train_labels.flatten()
test_labels = test_labels.flatten()

train_images = train_images / 255.0
test_images = test_images / 255.0

# class 명칭을 지정해 줌
class_names = [
    "airplane",
    "automobile",
    "bird",
    "cat",
    "deer",
    "dog",
    "frog",
    "horse",
    "ship",
    "truck",
]

model = models.Sequential()

model.add(layers.Conv2D(32, (3,3), padding="SAME", activation='relu', input_shape=(32, 32, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3,3), padding="SAME", activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3,3), padding="SAME", activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))

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

# 파라미터 수 읽기
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d (Conv2D)              (None, 32, 32, 32)        896       
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 16, 16, 32)        0         
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 16, 16, 64)        18496     
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 8, 8, 64)          0         
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 8, 8, 64)          36928     
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 4, 4, 64)          0         
_________________________________________________________________
flatten (Flatten)            (None, 1024)              0         
_________________________________________________________________
dense (Dense)                (None, 64)                65600     
_________________________________________________________________
dense_1 (Dense)              (None, 10)                650       
=================================================================
Total params: 122,570
Trainable params: 122,570
Non-trainable params: 0
_________________________________________________________________
```

In [2]:

```python
batch_size = 64
early_stop = keras.callbacks.EarlyStopping(monitor='val_loss', patience=10)
model.fit(train_images, train_labels,
          batch_size=batch_size,
          validation_split = 0.1,
          epochs=20,
          callbacks=[early_stop])

test_loss, test_acc = model.evaluate(test_images,  test_labels, verbose=2)

print('\n테스트 정확도:', test_acc)
```

```
Epoch 1/20
704/704 [==============================] - 22s 31ms/step - loss: 1.5035 - accuracy: 0.4584 - val_l
oss: 1.1930 - val_accuracy: 0.5796
Epoch 2/20
704/704 [==============================] - 22s 32ms/step - loss: 1.1015 - accuracy: 0.6107 - val_l
oss: 1.0344 - val_accuracy: 0.6386
Epoch 3/20
704/704 [==============================] - 21s 30ms/step - loss: 0.9432 - accuracy: 0.6702 - val_l
oss: 0.9079 - val_accuracy: 0.6794
Epoch 4/20
704/704 [==============================] - 22s 31ms/step - loss: 0.8404 - accuracy: 0.7082 - val_l
oss: 0.8242 - val_accuracy: 0.7192
Epoch 5/20
704/704 [==============================] - 24s 35ms/step - loss: 0.7696 - accuracy: 0.7309 - val_l
oss: 0.8241 - val_accuracy: 0.7158
Epoch 6/20
704/704 [==============================] - 27s 39ms/step - loss: 0.7078 - accuracy: 0.7531 - val_l
oss: 0.7837 - val_accuracy: 0.7328
Epoch 7/20
704/704 [==============================] - 24s 34ms/step - loss: 0.6472 - accuracy: 0.7755 - val_l
oss: 0.7909 - val_accuracy: 0.7314
Epoch 8/20
704/704 [==============================] - 22s 31ms/step - loss: 0.6040 - accuracy: 0.7885 - val_l
oss: 0.7932 - val_accuracy: 0.7370
Epoch 9/20
704/704 [==============================] - 23s 33ms/step - loss: 0.5601 - accuracy: 0.8039 - val_l
oss: 0.8059 - val_accuracy: 0.7332
Epoch 10/20
704/704 [==============================] - 22s 31ms/step - loss: 0.5145 - accuracy: 0.8197 - val_l
oss: 0.7938 - val_accuracy: 0.7366
Epoch 11/20
704/704 [==============================] - 22s 32ms/step - loss: 0.4813 - accuracy: 0.8316 - val_l
oss: 0.8061 - val_accuracy: 0.7418
Epoch 12/20
704/704 [==============================] - 22s 32ms/step - loss: 0.4436 - accuracy: 0.8434 - val_l
oss: 0.8239 - val_accuracy: 0.7364
Epoch 13/20
704/704 [==============================] - 22s 31ms/step - loss: 0.4130 - accuracy: 0.8538 - val_l
oss: 0.8594 - val_accuracy: 0.7360
Epoch 14/20
704/704 [==============================] - 22s 31ms/step - loss: 0.3741 - accuracy: 0.8694 - val_l
oss: 0.8247 - val_accuracy: 0.7480
Epoch 15/20
704/704 [==============================] - 22s 31ms/step - loss: 0.3549 - accuracy: 0.8733 - val_l
oss: 0.8820 - val_accuracy: 0.7448
Epoch 16/20
704/704 [==============================] - 22s 31ms/step - loss: 0.3152 - accuracy: 0.8880 - val_l
oss: 0.9220 - val_accuracy: 0.7430
313/313 - 2s - loss: 1.0121 - accuracy: 0.7289

테스트 정확도: 0.7289000153541565
```

# 2번

In [3]:

```python
#Dropout 적용
cifar10 = keras.datasets.cifar10

(train_images, train_labels), (test_images, test_labels) = cifar10.load_data()

train_labels = train_labels.flatten()
test_labels = test_labels.flatten()

train_images = train_images / 255.0
test_images = test_images / 255.0

# class 명칭을 지정해 줌
class_names = [
    "airplane",
    "automobile",
    "bird",
    "cat",
    "deer",
    "dog",
    "frog",
    "horse",
    "ship",
    "truck",
]

model2 = models.Sequential()

model2.add(layers.Conv2D(32, (3,3), padding="SAME", activation='relu', input_shape=(32, 32, 3)))
model2.add(layers.MaxPooling2D((2, 2)))
model2.add(layers.Dropout(0.2))
model2.add(layers.Conv2D(64, (3,3), padding="SAME", activation='relu'))
model2.add(layers.MaxPooling2D((2, 2)))
model2.add(layers.Dropout(0.3))
model2.add(layers.Conv2D(64, (3,3), padding="SAME", activation='relu'))
model2.add(layers.MaxPooling2D((2, 2)))
model2.add(layers.Dropout(0.4))
model2.add(layers.Flatten())
model2.add(layers.Dense(64, activation='relu'))
model2.add(layers.Dense(10, activation='softmax'))

model2.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# 파라미터 수 읽기
model2.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_3 (Conv2D)            (None, 32, 32, 32)        896       
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 16, 16, 32)        0         
_________________________________________________________________
dropout (Dropout)            (None, 16, 16, 32)        0         
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 16, 16, 64)        18496     
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 8, 8, 64)          0         
_________________________________________________________________
dropout_1 (Dropout)          (None, 8, 8, 64)          0         
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 8, 8, 64)          36928     
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 4, 4, 64)          0         
_________________________________________________________________
dropout_2 (Dropout)          (None, 4, 4, 64)          0         
_________________________________________________________________
flatten_1 (Flatten)          (None, 1024)              0         
_________________________________________________________________
dense_2 (Dense)              (None, 64)                65600     
_________________________________________________________________
dense_3 (Dense)              (None, 10)                650       
=================================================================
Total params: 122,570
Trainable params: 122,570
Non-trainable params: 0
_________________________________________________________________
```

In [4]:

```python
batch_size = 64
early_stop = keras.callbacks.EarlyStopping(monitor='val_loss', patience=10)
model2.fit(train_images, train_labels,
          batch_size=batch_size,
          validation_split = 0.1,
          epochs=20,
          callbacks=[early_stop])

test_loss, test_acc = model2.evaluate(test_images,  test_labels, verbose=2)

print('\n테스트 정확도:', test_acc)
```

```
Epoch 1/20
704/704 [==============================] - 25s 34ms/step - loss: 1.7053 - accuracy: 0.3711 - val_l
oss: 1.3590 - val_accuracy: 0.5160
Epoch 2/20
704/704 [==============================] - 24s 34ms/step - loss: 1.3516 - accuracy: 0.5096 - val_l
oss: 1.1794 - val_accuracy: 0.5862
Epoch 3/20
704/704 [==============================] - 24s 35ms/step - loss: 1.2188 - accuracy: 0.5619 - val_l
oss: 1.0464 - val_accuracy: 0.6412
Epoch 4/20
704/704 [==============================] - 25s 36ms/step - loss: 1.1230 - accuracy: 0.5962 - val_l
oss: 0.9824 - val_accuracy: 0.6612
Epoch 5/20
704/704 [==============================] - 25s 35ms/step - loss: 1.0649 - accuracy: 0.6219 - val_l
oss: 1.0602 - val_accuracy: 0.6320
Epoch 6/20
704/704 [==============================] - 25s 35ms/step - loss: 1.0155 - accuracy: 0.6370 - val_l
oss: 0.8740 - val_accuracy: 0.7066
Epoch 7/20
704/704 [==============================] - 25s 35ms/step - loss: 0.9753 - accuracy: 0.6533 - val_l
oss: 0.8259 - val_accuracy: 0.7262
Epoch 8/20
704/704 [==============================] - 25s 35ms/step - loss: 0.9267 - accuracy: 0.6721 - val_l
oss: 0.8067 - val_accuracy: 0.7292
Epoch 9/20
704/704 [==============================] - 25s 36ms/step - loss: 0.9058 - accuracy: 0.6803 - val_l
oss: 0.7920 - val_accuracy: 0.7310
Epoch 10/20
704/704 [==============================] - 25s 36ms/step - loss: 0.8807 - accuracy: 0.6865 - val_l
oss: 0.7722 - val_accuracy: 0.7444
Epoch 11/20
704/704 [==============================] - 25s 36ms/step - loss: 0.8556 - accuracy: 0.6955 - val_l
oss: 0.7524 - val_accuracy: 0.7464
Epoch 12/20
704/704 [==============================] - 25s 36ms/step - loss: 0.8450 - accuracy: 0.6991 - val_l
oss: 0.7301 - val_accuracy: 0.7542
Epoch 13/20
704/704 [==============================] - 25s 36ms/step - loss: 0.8263 - accuracy: 0.7080 - val_l
oss: 0.7350 - val_accuracy: 0.7604
Epoch 14/20
704/704 [==============================] - 25s 36ms/step - loss: 0.8094 - accuracy: 0.7139 - val_l
oss: 0.7055 - val_accuracy: 0.7646
Epoch 15/20
704/704 [==============================] - 25s 36ms/step - loss: 0.7992 - accuracy: 0.7177 - val_l
oss: 0.6963 - val_accuracy: 0.7646
Epoch 16/20
704/704 [==============================] - 25s 36ms/step - loss: 0.7816 - accuracy: 0.7251 - val_l
oss: 0.7009 - val_accuracy: 0.7596
Epoch 17/20
704/704 [==============================] - 25s 35ms/step - loss: 0.7743 - accuracy: 0.7254 - val_l
oss: 0.7001 - val_accuracy: 0.7644
Epoch 18/20
704/704 [==============================] - 25s 35ms/step - loss: 0.7625 - accuracy: 0.7300 - val_l
oss: 0.6567 - val_accuracy: 0.7788
Epoch 19/20
704/704 [==============================] - 25s 36ms/step - loss: 0.7461 - accuracy: 0.7346 - val_l
oss: 0.6710 - val_accuracy: 0.7762
Epoch 20/20
704/704 [==============================] - 24s 34ms/step - loss: 0.7462 - accuracy: 0.7348 - val_l
oss: 0.6681 - val_accuracy: 0.7808
313/313 - 2s - loss: 0.7070 - accuracy: 0.7544

테스트 정확도: 0.7544000148773193
```

# 3번

In [5]:

```python
#3
cifar10 = keras.datasets.cifar10

(train_images, train_labels), (test_images, test_labels) = cifar10.load_data()

train_labels = train_labels.flatten()
test_labels = test_labels.flatten()

train_images = train_images / 255.0
test_images = test_images / 255.0

# class 명칭을 지정해 줌
class_names = [
    "airplane",
    "automobile",
    "bird",
    "cat",
    "deer",
    "dog",
    "frog",
    "horse",
    "ship",
    "truck",
]

model3 = models.Sequential()

model3.add(layers.Conv2D(32, (3,3), padding="SAME", activation='relu', input_shape=(32, 32, 3)))
model3.add(layers.MaxPooling2D((2, 2)))
model3.add(layers.Conv2D(32, (3,3), padding="SAME", activation='relu'))

model3.add(layers.Conv2D(64, (3,3), padding="SAME", activation='relu'))
model3.add(layers.MaxPooling2D((2, 2)))
model3.add(layers.Conv2D(64, (3,3), padding="SAME", activation='relu'))

model3.add(layers.Flatten())
model3.add(layers.Dense(64, activation='relu'))
model3.add(layers.Dense(10, activation='softmax'))

model3.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# 파라미터 수 읽기
model3.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_6 (Conv2D)            (None, 32, 32, 32)        896       
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 16, 16, 32)        0         
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 16, 16, 32)        9248      
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 16, 16, 64)        18496     
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 8, 8, 64)          0         
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 8, 8, 64)          36928     
_________________________________________________________________
flatten_2 (Flatten)          (None, 4096)              0         
_________________________________________________________________
dense_4 (Dense)              (None, 64)                262208    
_________________________________________________________________
dense_5 (Dense)              (None, 10)                650       
=================================================================
Total params: 328,426
Trainable params: 328,426
Non-trainable params: 0
_________________________________________________________________
```

In [6]:

```python
batch_size = 64
early_stop = keras.callbacks.EarlyStopping(monitor='val_loss', patience=10)
model3.fit(train_images, train_labels,
          batch_size=batch_size,
          validation_split = 0.1,
          epochs=20,
          callbacks=[early_stop])

test_loss, test_acc = model3.evaluate(test_images,  test_labels, verbose=2)

print('\n테스트 정확도:', test_acc)
```

```
Epoch 1/20
704/704 [==============================] - 28s 39ms/step - loss: 1.5393 - accuracy: 0.4401 - val_l
oss: 1.3148 - val_accuracy: 0.5260
Epoch 2/20
704/704 [==============================] - 28s 39ms/step - loss: 1.1364 - accuracy: 0.5972 - val_l
oss: 1.0151 - val_accuracy: 0.6362
Epoch 3/20
704/704 [==============================] - 27s 39ms/step - loss: 0.9460 - accuracy: 0.6642 - val_l
oss: 0.9692 - val_accuracy: 0.6610
Epoch 4/20
704/704 [==============================] - 28s 40ms/step - loss: 0.8178 - accuracy: 0.7130 - val_l
oss: 0.8817 - val_accuracy: 0.6962
Epoch 5/20
704/704 [==============================] - 28s 40ms/step - loss: 0.7214 - accuracy: 0.7473 - val_l
oss: 0.8179 - val_accuracy: 0.7164
Epoch 6/20
704/704 [==============================] - 28s 40ms/step - loss: 0.6402 - accuracy: 0.7757 - val_l
oss: 0.8096 - val_accuracy: 0.7282
Epoch 7/20
704/704 [==============================] - 27s 39ms/step - loss: 0.5621 - accuracy: 0.8042 - val_l
oss: 0.8188 - val_accuracy: 0.7252
Epoch 8/20
704/704 [==============================] - 27s 39ms/step - loss: 0.4909 - accuracy: 0.8281 - val_l
oss: 0.7983 - val_accuracy: 0.7440
Epoch 9/20
704/704 [==============================] - 28s 39ms/step - loss: 0.4218 - accuracy: 0.8529 - val_l
oss: 0.8723 - val_accuracy: 0.7290
Epoch 10/20
704/704 [==============================] - 27s 39ms/step - loss: 0.3628 - accuracy: 0.8733 - val_l
oss: 0.9345 - val_accuracy: 0.7298
Epoch 11/20
704/704 [==============================] - 27s 39ms/step - loss: 0.3144 - accuracy: 0.8901 - val_l
oss: 0.9795 - val_accuracy: 0.7326
Epoch 12/20
704/704 [==============================] - 27s 39ms/step - loss: 0.2623 - accuracy: 0.9078 - val_l
oss: 1.0632 - val_accuracy: 0.7164
Epoch 13/20
704/704 [==============================] - 28s 39ms/step - loss: 0.2314 - accuracy: 0.9172 - val_l
oss: 1.1263 - val_accuracy: 0.7250
Epoch 14/20
704/704 [==============================] - 28s 39ms/step - loss: 0.2022 - accuracy: 0.9281 - val_l
oss: 1.1696 - val_accuracy: 0.7244
Epoch 15/20
704/704 [==============================] - 28s 39ms/step - loss: 0.1638 - accuracy: 0.9410 - val_l
oss: 1.2723 - val_accuracy: 0.7224
Epoch 16/20
704/704 [==============================] - 28s 40ms/step - loss: 0.1557 - accuracy: 0.9454 - val_l
oss: 1.3779 - val_accuracy: 0.7242
Epoch 17/20
704/704 [==============================] - 28s 40ms/step - loss: 0.1357 - accuracy: 0.9510 - val_l
oss: 1.4335 - val_accuracy: 0.7048
Epoch 18/20
704/704 [==============================] - 28s 40ms/step - loss: 0.1302 - accuracy: 0.9532 - val_l
oss: 1.6494 - val_accuracy: 0.7220
313/313 - 2s - loss: 1.7212 - accuracy: 0.7058

테스트 정확도: 0.7057999968528748
```