```
import tensorflow as tf
#from tensorflow.keras.preprocessing.image import ImageDataGenerator -> 안 씀
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout, BatchNormalization
#from tensorflow.keras.regularizers import l2
from tensorflow.keras.optimizers import Adam
import numpy as np
import matplotlib.pyplot as plt
```

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

x_train.shape, x_test.shape

#x_train, y_train = x_train[:10000]
```

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

```
x_train, y_train = x_train[:12000], y_train[:12000]
```

```
x_train
```

```
[[[224, 243, 245],
  [140, 199, 219],
  [117, 178, 206],
  ...,
  [168, 195, 198],
  [148, 188, 198],
  [133, 187, 203]],

 [[202, 234, 243],
  [135, 194, 215],
  [108, 169, 196],
  ...,
  [150, 183, 187],
  [122, 166, 179],
  [100, 155, 172]],

 [[174, 219, 236],
  [139, 198, 221],
  [109, 176, 209],
  ...,
  [132, 166, 175],
  [104, 157, 168],
  [104, 176, 175]],

 ...,

 [[  5,  20,  14],
  [  6,  22,  18],
  [  9,  29,  26],
  ...,
  [ 13,  27,  24],
  [  1,  21,  20],
  [  3,  18,  18]],

 [[ 37,  66,  55],
  [ 43,  73,  61],
  [ 50,  80,  67],
  ...,
  [ 24,  42,  27],
  [  8,  25,  18],
  [  4,  14,  14]],

 [[ 51,  80,  61],
  [ 53,  81,  60],
  [ 55,  87,  65],
  ...,
  [ 45,  61,  35],
  [ 34,  45,  27],
  [ 12,  16,  14]]],


[[[ 58,  47,  62],
  [ 62,  53,  70],
  [ 73,  51,  68],
  ...,
  [ 81,  38,  36],
  [ 88,  45,  42],
  [ 65,  30,  33]],
x_train.shape, y_train.shape
```

```
((12000, 32, 32, 3), (12000, 1))
```

```
model = Sequential()

model.add(Conv2D(32, 3, activation='relu', padding = 'same', input_shape = x_train.shape[1:]))  # 필터 32개, 필터 크기 3
model.add(Conv2D(32, 3, padding='same', activation='relu'))  # 이미지 크기 똑같이. padding = 'same'
model.add(MaxPooling2D())  # default 2 x 2, 이미지 크기 절반.
model.add(Dropout(0.25))



model.add(Conv2D(128, 3, padding = 'same', activation = 'relu'))
model.add(Conv2D(128, 3, padding = 'same', activation = 'relu'))
# 5줄 추가
model.add(MaxPooling2D())
model.add(Conv2D(256, 3, padding = 'same', activation = 'relu'))
model.add(MaxPooling2D())
model.add(Conv2D(512, 3, padding = 'same', activation = 'relu'))
model.add(Conv2D(1024, 3, padding = 'same', activation = 'relu'))
model.add(MaxPooling2D())

#model.add(GlobalAveragePooling2D())  # FC 없애려고 도입.
model.add(Dropout(0.25))



model.add(BatchNormalization())
model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.5))
#model.add(Dense(256, activation = 'relu'))
#model.add(Dropout(0.5))
model.add(Dense(10, activation='softmax'))

model.summary()
```

```
Model: "sequential_1"
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                         ┃ Output Shape                ┃         Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━┩
│ conv2d_7 (Conv2D)                    │ (None, 32, 32, 32)          │             896 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ conv2d_8 (Conv2D)                    │ (None, 32, 32, 32)          │           9,248 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ max_pooling2d_3 (MaxPooling2D)       │ (None, 16, 16, 32)          │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dropout_3 (Dropout)                  │ (None, 16, 16, 32)          │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ conv2d_9 (Conv2D)                    │ (None, 16, 16, 128)         │          36,992 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ conv2d_10 (Conv2D)                   │ (None, 16, 16, 128)         │         147,584 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ max_pooling2d_4 (MaxPooling2D)       │ (None, 8, 8, 128)           │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ conv2d_11 (Conv2D)                   │ (None, 8, 8, 256)           │         295,168 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ max_pooling2d_5 (MaxPooling2D)       │ (None, 4, 4, 256)           │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ conv2d_12 (Conv2D)                   │ (None, 4, 4, 512)           │       1,180,160 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ conv2d_13 (Conv2D)                   │ (None, 4, 4, 1024)          │       4,719,616 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ max_pooling2d_6 (MaxPooling2D)       │ (None, 2, 2, 1024)          │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dropout_4 (Dropout)                  │ (None, 2, 2, 1024)          │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ batch_normalization_1                │ (None, 2, 2, 1024)          │           4,096 │
│ (BatchNormalization)                 │                             │                 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ flatten_1 (Flatten)                  │ (None, 4096)                │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dense_2 (Dense)                      │ (None, 512)                 │       2,097,664 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dropout_5 (Dropout)                  │ (None, 512)                 │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dense_3 (Dense)                      │ (None, 10)                  │           5,130 │
└──────────────────────────────────────┴─────────────────────────────┴─────────────────┘
 Total params: 8,496,554 (32.41 MB)
 Trainable params: 8,494,506 (32.40 MB)
```

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

```
history = model.fit(x_train, y_train, epochs = 30, batch_size = 32)
```

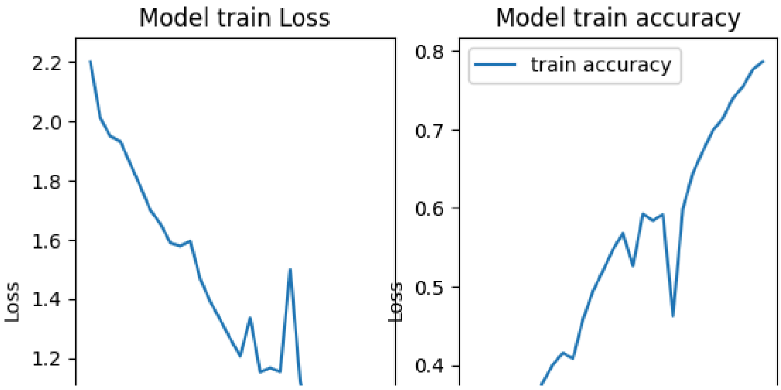
```
Epoch 2/30
375/375 ━━━━━━━━━━━━━━━━━━━━ 15s 14ms/step - accuracy: 0.1977 - loss: 2.0554
Epoch 3/30
375/375 ━━━━━━━━━━━━━━━━━━━━ 10s 13ms/step - accuracy: 0.2225 - loss: 1.9526
Epoch 4/30
375/375 ━━━━━━━━━━━━━━━━━━━━ 5s 14ms/step - accuracy: 0.2558 - loss: 1.9297
Epoch 5/30
375/375 ━━━━━━━━━━━━━━━━━━━━ 10s 14ms/step - accuracy: 0.2699 - loss: 1.8597
Epoch 6/30
375/375 ━━━━━━━━━━━━━━━━━━━━ 10s 14ms/step - accuracy: 0.3132 - loss: 1.7837
Epoch 7/30
375/375 ━━━━━━━━━━━━━━━━━━━━ 5s 14ms/step - accuracy: 0.3476 - loss: 1.7140
Epoch 8/30
375/375 ━━━━━━━━━━━━━━━━━━━━ 5s 13ms/step - accuracy: 0.3747 - loss: 1.6558
Epoch 9/30
375/375 ━━━━━━━━━━━━━━━━━━━━ 5s 14ms/step - accuracy: 0.3962 - loss: 1.6004
Epoch 10/30
375/375 ━━━━━━━━━━━━━━━━━━━━ 10s 13ms/step - accuracy: 0.4190 - loss: 1.5695
Epoch 11/30
375/375 ━━━━━━━━━━━━━━━━━━━━ 5s 13ms/step - accuracy: 0.3882 - loss: 1.6334
Epoch 12/30
375/375 ━━━━━━━━━━━━━━━━━━━━ 5s 13ms/step - accuracy: 0.4564 - loss: 1.4677
Epoch 13/30
375/375 ━━━━━━━━━━━━━━━━━━━━ 5s 14ms/step - accuracy: 0.4891 - loss: 1.3992
Epoch 14/30
375/375 ━━━━━━━━━━━━━━━━━━━━ 5s 13ms/step - accuracy: 0.5167 - loss: 1.3399
Epoch 15/30
375/375 ━━━━━━━━━━━━━━━━━━━━ 5s 13ms/step - accuracy: 0.5468 - loss: 1.2503
Epoch 16/30
375/375 ━━━━━━━━━━━━━━━━━━━━ 5s 14ms/step - accuracy: 0.5710 - loss: 1.2016
Epoch 17/30
375/375 ━━━━━━━━━━━━━━━━━━━━ 5s 13ms/step - accuracy: 0.5015 - loss: 1.3987
Epoch 18/30
```

```
375/375 ━━━━━━━━━━━━━━━━━━━━ 10s 14ms/step - accuracy: 0.6105 - loss: 1.1015
Epoch 20/30
375/375 ━━━━━━━━━━━━━━━━━━━━ 5s 14ms/step - accuracy: 0.5991 - loss: 1.1208
Epoch 21/30
375/375 ━━━━━━━━━━━━━━━━━━━━ 5s 13ms/step - accuracy: 0.4578 - loss: 1.5218
Epoch 22/30
375/375 ━━━━━━━━━━━━━━━━━━━━ 5s 14ms/step - accuracy: 0.5795 - loss: 1.1683
Epoch 23/30
375/375 ━━━━━━━━━━━━━━━━━━━━ 5s 13ms/step - accuracy: 0.6376 - loss: 0.9940
Epoch 24/30
375/375 ━━━━━━━━━━━━━━━━━━━━ 5s 13ms/step - accuracy: 0.6608 - loss: 0.9186
Epoch 25/30
375/375 ━━━━━━━━━━━━━━━━━━━━ 5s 14ms/step - accuracy: 0.7032 - loss: 0.8351
Epoch 26/30
375/375 ━━━━━━━━━━━━━━━━━━━━ 5s 13ms/step - accuracy: 0.7207 - loss: 0.7624
Epoch 27/30
375/375 ━━━━━━━━━━━━━━━━━━━━ 5s 14ms/step - accuracy: 0.7458 - loss: 0.7326
Epoch 28/30
375/375 ━━━━━━━━━━━━━━━━━━━━ 10s 14ms/step - accuracy: 0.7534 - loss: 0.6890
Epoch 29/30
375/375 ━━━━━━━━━━━━━━━━━━━━ 5s 13ms/step - accuracy: 0.7818 - loss: 0.6219
Epoch 30/30
375/375 ━━━━━━━━━━━━━━━━━━━━ 5s 13ms/step - accuracy: 0.7958 - loss: 0.5849
```

```
plt.subplot(121)
plt.plot(history.history['loss'], label='train loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.title('Model train Loss')

plt.subplot(122)
plt.plot(history.history['accuracy'], label='train accuracy')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.title('Model train accuracy')

plt.legend()
plt.show()
```


Model train Loss
Loss
Model train accuracy
train accuracy
Loss