이미지 증강(Image Augmentation)을 사용하여 CNN 학습

> ## Overfitting 대응책

```
import warnings
warnings.filterwarnings('ignore')
```

Import Keras

```
import keras

keras.__version__
```

```
    '2.4.3'
```

I. Google Drive Mount

- 'dogs_and_cats_small.zip' 디렉토리를 구글드라이브에 업로드

```
from google.colab import drive

drive.mount('/content/drive')
```

```
    Mounted at /content/drive
```

> ## 1) 구글 드라이브 마운트 결과 확인

```
!ls -l '/content/drive/My Drive/Colab Notebooks/datasets/dogs_and_cats_small.zip'
```

```
    -rw------- 1 root root 90618980 Mar  4 04:51 '/content/drive/My Drive/Colab Notebooks/dataset
```

> ## 2) unzip 'dogs_and_cats_small.zip'

```
!unzip /content/drive/My₩ Drive/Colab₩ Notebooks/datasets/dogs_and_cats_small.zip
```

```
    Archive:  /content/drive/My Drive/Colab Notebooks/datasets/dogs_and_cats_small.zip
      inflating: test/cats/cat.1501.jpg
      inflating: test/cats/cat.1502.jpg
      inflating: test/cats/cat.1503.jpg
```

```
  inflating: test/cats/cat.1504.jpg
  inflating: test/cats/cat.1505.jpg
  inflating: test/cats/cat.1506.jpg
  inflating: test/cats/cat.1507.jpg
  inflating: test/cats/cat.1508.jpg
  inflating: test/cats/cat.1509.jpg
  inflating: test/cats/cat.1510.jpg
  inflating: test/cats/cat.1511.jpg
  inflating: test/cats/cat.1512.jpg
  inflating: test/cats/cat.1513.jpg
  inflating: test/cats/cat.1514.jpg
  inflating: test/cats/cat.1515.jpg
  inflating: test/cats/cat.1516.jpg
  inflating: test/cats/cat.1517.jpg
  inflating: test/cats/cat.1518.jpg
  inflating: test/cats/cat.1519.jpg
  inflating: test/cats/cat.1520.jpg
  inflating: test/cats/cat.1521.jpg
  inflating: test/cats/cat.1522.jpg
  inflating: test/cats/cat.1523.jpg
  inflating: test/cats/cat.1524.jpg
  inflating: test/cats/cat.1525.jpg
  inflating: test/cats/cat.1526.jpg
  inflating: test/cats/cat.1527.jpg
  inflating: test/cats/cat.1528.jpg
  inflating: test/cats/cat.1529.jpg
  inflating: test/cats/cat.1530.jpg
  inflating: test/cats/cat.1531.jpg
  inflating: test/cats/cat.1532.jpg
  inflating: test/cats/cat.1533.jpg
  inflating: test/cats/cat.1534.jpg
  inflating: test/cats/cat.1535.jpg
  inflating: test/cats/cat.1536.jpg
  inflating: test/cats/cat.1537.jpg
  inflating: test/cats/cat.1538.jpg
  inflating: test/cats/cat.1539.jpg
  inflating: test/cats/cat.1540.jpg
  inflating: test/cats/cat.1541.jpg
  inflating: test/cats/cat.1542.jpg
  inflating: test/cats/cat.1543.jpg
  inflating: test/cats/cat.1544.jpg
  inflating: test/cats/cat.1545.jpg
  inflating: test/cats/cat.1546.jpg
  inflating: test/cats/cat.1547.jpg
  inflating: test/cats/cat.1548.jpg
  inflating: test/cats/cat.1549.jpg
  inflating: test/cats/cat.1550.jpg
  inflating: test/cats/cat.1551.jpg
  inflating: test/cats/cat.1552.jpg
  inflating: test/cats/cat.1553.jpg
  inflating: test/cats/cat.1554.jpg
  inflating: test/cats/cat.1555.jpg
  inflating: test/cats/cat.1556.jpg
  inflating: test/cats/cat.1557.jpg
  inflating: test/cats/cat.1558.jpg
  inflating: test/cats/cat.1559.jpg
```

```
!ls -l
```

```
total 20
drwx------ 5 root root 4096 Mar  8 07:35 drive
```

```
drwxr-xr-x 1 root root 4096 Mar  1 14:35 sample_data
drwxr-xr-x 4 root root 4096 Mar  8 07:35 test
drwxr-xr-x 4 root root 4096 Mar  8 07:35 train
drwxr-xr-x 4 root root 4096 Mar  8 07:35 validation
```

3) [Optional] Image Augmentation Test

- rotation_range = 40 : 0도에서 40도 사이에서 임의의 각도록 회전
- width_shift_range = 0.2 : 20% 픽셀 내외로 좌우 이동
- height_shift_range = 0.2 : 20% 픽셀 내외로 상하 이동
- shear_range = 0.2 : 0.2 라디안 내외로 시계 반대방향으로 변형
- zoom_range = 0.2 : 80%에서 120% 범위에서 확대/축소
- horizontal_flip = True : 수평방향 뒤집기
- vertical_flip = True : 수직방향 뒤집기
- fill_mode = 'nearest' : 주변 픽셀로 이미지 채우기

```
from keras.preprocessing.image import ImageDataGenerator

datagen = ImageDataGenerator(rotation_range = 40,
                             width_shift_range = 0.2,
                             height_shift_range = 0.2,
                             shear_range = 0.2,
                             zoom_range = 0.2,
                             horizontal_flip = True,
                             vertical_flip = True,
                             fill_mode = 'nearest')
```

```
from keras.preprocessing import image
import matplotlib.pyplot as plt
import os

train_cats_dir = train_dir = os.path.join('train', 'cats')
fnames = sorted([os.path.join(train_cats_dir, fname) for fname in os.listdir(train_cats_dir)])

# 테스트 이미지 선택
img_path = fnames[77]

# 이미지 읽고 크기 변경
img = image.load_img(img_path, target_size=(150, 150))

# (150, 150, 3) 배열 변환
x = image.img_to_array(img)

# (1, 150, 150, 3) 변환
x = x.reshape((1,) + x.shape)

# 랜덤하게 변환된 이미지 배치 생성
i = 0
```

```
for batch in datagen.flow(x, batch_size=1):
    plt.figure(i)
    imgplot = plt.imshow(image.array_to_img(batch[0]))
    i += 1
    if i % 4 == 0:
        break

plt.show()
```

II. Data Preprocessing

1) Image_File Directory Setting

- train_dir
- valid_dir
- test_dir

```
train_dir = 'train'
valid_dir = 'validation'
test_dir = 'test'
```

2) ImageDataGenerator( ) & flow_from_directory( )

- Normalization & Augmentation
    - ImageDataGenerator( )
- Resizing & Generator
    - flow_from_directory( )

```
from keras.preprocessing.image import ImageDataGenerator

# With Augmentation
train_datagen = ImageDataGenerator(rescale = 1./255,
                                   rotation_range = 40,
                                   width_shift_range = 0.2,
                                   height_shift_range = 0.2,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip=True,
                                   vertical_flip = True,
                                   fill_mode = 'nearest')

# Without Augmentation
```

```
valid_datagen = ImageDataGenerator(rescale = 1./255)



# With Augmentation
train_generator = train_datagen.flow_from_directory(
                  train_dir,
                  target_size = (150, 150),
                  batch_size = 20,
                  class_mode = 'binary')

# Without Augmentation
valid_generator = valid_datagen.flow_from_directory(
                  valid_dir,
                  target_size = (150, 150),
                  batch_size = 20,
                  class_mode = 'binary')
```

```
Found 2000 images belonging to 2 classes.
Found 1000 images belonging to 2 classes.
```

III. CNN Keras Modeling

1) Model Define

- Feature Extraction & Classification
    - Dropout Layer

```
from keras import layers
from keras import models

model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation = 'relu', input_shape = (150, 150, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation = 'relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation = 'relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation = 'relu'))
model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Flatten())
model.add(layers.Dropout(0.5))
model.add(layers.Dense(512, activation = 'relu'))
model.add(layers.Dense(1, activation = 'sigmoid'))
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d (Conv2D)              (None, 148, 148, 32)      896       
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 74, 74, 32)        0         
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 72, 72, 64)        18496     
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 36, 36, 64)        0         
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 34, 34, 128)       73856     
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 17, 17, 128)       0         
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 15, 15, 128)       147584    
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 7, 7, 128)         0         
_________________________________________________________________
flatten (Flatten)            (None, 6272)              0         
_________________________________________________________________
dropout (Dropout)            (None, 6272)              0         
_________________________________________________________________
dense (Dense)                (None, 512)               3211776   
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 513       
=================================================================
Total params: 3,453,121
Trainable params: 3,453,121
Non-trainable params: 0
_________________________________________________________________
```

2) Model Compile

- 모델 학습방법 설정

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

3) Model Fit

- 약 30분
    - epochs : 60 -> 100

```
%%time

Hist_dandc = model.fit(train_generator,
                       steps_per_epoch = 100,
                       epochs = 100,
                       validation_data = valid_generator,
                       validation_steps = 50)
```

```
Epoch 1/100
100/100 [==============================] - 51s 176ms/step - loss: 0.7200 - accuracy: 0.511
Epoch 2/100
100/100 [==============================] - 17s 171ms/step - loss: 0.6921 - accuracy: 0.505
Epoch 3/100
100/100 [==============================] - 17s 172ms/step - loss: 0.6935 - accuracy: 0.472
Epoch 4/100
100/100 [==============================] - 17s 174ms/step - loss: 0.6939 - accuracy: 0.493
Epoch 5/100
100/100 [==============================] - 17s 174ms/step - loss: 0.6933 - accuracy: 0.488
Epoch 6/100
100/100 [==============================] - 17s 170ms/step - loss: 0.6936 - accuracy: 0.490
Epoch 7/100
100/100 [==============================] - 17s 169ms/step - loss: 0.6933 - accuracy: 0.494
Epoch 8/100
100/100 [==============================] - 17s 172ms/step - loss: 0.6932 - accuracy: 0.499
Epoch 9/100
100/100 [==============================] - 17s 170ms/step - loss: 0.6934 - accuracy: 0.490
Epoch 10/100
100/100 [==============================] - 17s 170ms/step - loss: 0.6918 - accuracy: 0.509
Epoch 11/100
100/100 [==============================] - 17s 169ms/step - loss: 0.6905 - accuracy: 0.513
Epoch 12/100
100/100 [==============================] - 17s 171ms/step - loss: 0.6919 - accuracy: 0.484
Epoch 13/100
100/100 [==============================] - 17s 171ms/step - loss: 0.6942 - accuracy: 0.489
Epoch 14/100
100/100 [==============================] - 17s 172ms/step - loss: 0.6932 - accuracy: 0.491
Epoch 15/100
100/100 [==============================] - 17s 174ms/step - loss: 0.6925 - accuracy: 0.508
Epoch 16/100
100/100 [==============================] - 17s 171ms/step - loss: 0.6932 - accuracy: 0.514
Epoch 17/100
100/100 [==============================] - 17s 169ms/step - loss: 0.6934 - accuracy: 0.502
Epoch 18/100
100/100 [==============================] - 17s 170ms/step - loss: 0.6922 - accuracy: 0.509
Epoch 19/100
100/100 [==============================] - 17s 169ms/step - loss: 0.6910 - accuracy: 0.524
Epoch 20/100
100/100 [==============================] - 17s 168ms/step - loss: 0.6832 - accuracy: 0.537
Epoch 21/100
100/100 [==============================] - 17s 168ms/step - loss: 0.6910 - accuracy: 0.551
Epoch 22/100
100/100 [==============================] - 17s 169ms/step - loss: 0.6833 - accuracy: 0.547
Epoch 23/100
100/100 [==============================] - 17s 169ms/step - loss: 0.6756 - accuracy: 0.571
Epoch 24/100
100/100 [==============================] - 17s 170ms/step - loss: 0.6787 - accuracy: 0.571
Epoch 25/100
100/100 [==============================] - 17s 174ms/step - loss: 0.6731 - accuracy: 0.603
Epoch 26/100
100/100 [==============================] - 17s 174ms/step - loss: 0.6644 - accuracy: 0.622
Epoch 27/100
100/100 [==============================] - 17s 172ms/step - loss: 0.6708 - accuracy: 0.593
Epoch 28/100
100/100 [==============================] - 17s 171ms/step - loss: 0.6607 - accuracy: 0.607
```

```
Epoch 29/100
100/100 [==============================] - 17s 171ms/step - loss: 0.6482 - accuracy: 0.632
```

4) 학습 결과 시각화

- Loss Visualization

```
import matplotlib.pyplot as plt

epochs = range(1, len(Hist_dandc.history['loss']) + 1)

plt.figure(figsize = (9, 6))
plt.plot(epochs, Hist_dandc.history['loss'])
plt.plot(epochs, Hist_dandc.history['val_loss'])

plt.title('Training & Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend(['Training Loss', 'Validation Loss'])
plt.grid()
plt.show()
```

- Accuracy Visualization

```
import matplotlib.pyplot as plt

epochs = range(1, len(Hist_dandc.history['loss']) + 1)

plt.figure(figsize = (9, 6))
plt.plot(epochs, Hist_dandc.history['accuracy'])
plt.plot(epochs, Hist_dandc.history['val_accuracy'])

plt.title('Training & Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend(['Training Accuracy', 'Validation Accuracy'])
plt.grid()
plt.show()
```

5) Model Evaluate

- test_generator

```
test_datagen = ImageDataGenerator(rescale = 1./255)

test_generator = test_datagen.flow_from_directory(
test_generator = test_datagen.flow_from_directory(
                 test_dir,
                 target_size = (150, 150),
                 batch_size = 20,
                 class_mode = 'binary')
```

```
Found 1000 images belonging to 2 classes.
```

- Loss & Accuracy

```
loss, accuracy = model.evaluate(test_generator,
                                steps = 50)

print('Loss = {:.5f}'.format(loss))
print('Accuracy = {:.5f}'.format(accuracy))
```

```
50/50 [==============================] - 3s 51ms/step - loss: 0.5316 - accuracy: 0.7640
Loss = 0.53165
Accuracy = 0.76400
```

IV. Model Save & Load to Google Drive

1) Google Drive Mount

```
from google.colab import drive

drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/c
```

2) Model Save

```
model.save('/content/drive/My Drive/Colab Notebooks/models/003_dogs_and_cats_augmentation.h5')
```

```
!ls -l /content/drive/My₩ Drive/Colab₩ Notebooks/models
```

```
total 105697
-rw------- 1 root root    34592 Mar  8 01:56 001_Model_iris.h5
-rw------- 1 root root 41498896 Mar  8 07:23 002_dogs_and_cats_small.h5
-rw------- 1 root root 41499744 Mar  8 08:04 003_dogs_and_cats_augmentation.h5
-rw------- 1 root root 25199032 Mar  8 07:43 004_dogs_and_cats_feature_extraction.h5
```

3) Model Load

```
from keras.models import load_model

model_google = load_model('/content/drive/My Drive/Colab Notebooks/models/003_dogs_and_cats_augment
```

```
loss, accuracy = model_google.evaluate(test_generator,
                                       steps = 50)

print('Loss = {:.5f}'.format(loss))
print('Accuracy = {:.5f}'.format(accuracy))
```

```
50/50 [==============================] - 3s 53ms/step - loss: 0.5316 - accuracy: 0.7640
Loss = 0.53165
Accuracy = 0.76400
```

#

#

#

The End

#

#

#