Intro-week4

> 말과 사람 이미지 분류하기

> 경로와 파일명

```
import os

# horses/humans 데이터셋 경로 지정
train_horse_dir = './tmp/horse-or-human/horses'
train_human_dir = './tmp/horse-or-human/humans'

# horses 파일 이름 리스트
train_horse_names = os.listdir(train_horse_dir)
print(train_horse_names[:10])

# humans 파일 이름 리스트
train_human_names = os.listdir(train_human_dir)
print(train_human_names[:10])

# horses/humans 총 이미지 파일 개수
print('total training horse images:', len(os.listdir(train_horse_dir)))
print('total training human images:', len(os.listdir(train_human_dir)))
```

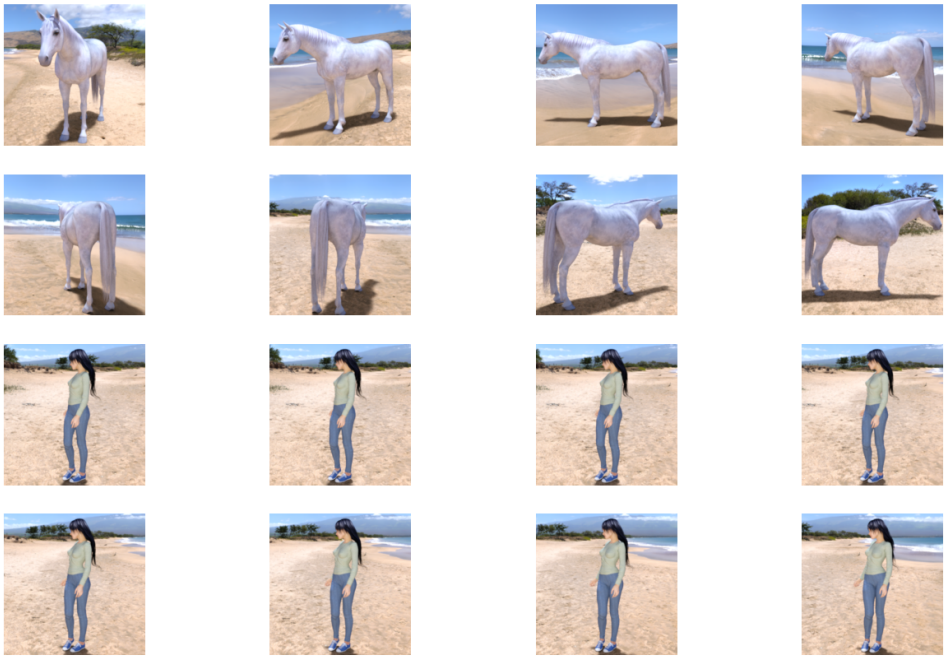
os.listdir()을 이용해 경로세 포함된 파일 이름을 리스트 형태로 불러온다. 위의 리스트를 확인해보면 말 이미지가 500개, 사람 이미지가 527개 있음을 알 수 있다.

> 이미지 확인하기

```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

nrows = 4
ncols = 4

pic_index = 0

fig = plt.gcf()
fig.set_size_inches(ncols * 4, nrows * 4)

pic_index += 8
next_horse_pix = [os.path.join(train_horse_dir, fname) for fname in
train_horse_names[pic_index-8:pic_index]]
next_human_pix = [os.path.join(train_human_dir, fname) for fname in
train_human_names[pic_index-8:pic_index]]
```

```
for i, img_path in enumerate(next_horse_pix+next_human_pix):
sp = plt.subplot(nrows, ncols, i + 1)
sp.axis('Off')

img = mpimg.imread(img_path)
plt.imshow(img)

plt.show()
```

Matplotlib를 이용해 말과 사람의 이미지를 각각 8개씩 띄워보자.



모델 구성하기

- 이미지 분류와 훈련을 위해 합성곱 신경망을 구성한다.

```
import tensorflow as tf

model = tf.keras.models.Sequential([
    # The first convolution
    tf.keras.layers.Conv2D(16, (3, 3), activation='relu', input_shape=(300, 300,
3)),
    tf.keras.layers.MaxPool2D(2, 2),
    # The second convolution
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu'),
    tf.keras.layers.MaxPool2D(2, 2),
```

```
    # The third convolution
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPool2D(2, 2),
    # The fourth convolution
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPool2D(2, 2),
    # The fifth convolution
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPool2D(2, 2),
    # Flatten
    tf.keras.layers.Flatten(),
    # 512 Neuron (Hidden layer)
    tf.keras.layers.Dense(512, activation='relu'),
    # 1 Output neuron
    tf.keras.layers.Dense(1, activation='sigmoid')
])

model.summary()
```

다섯 단계의 합성곱 뉴런층과 두 단계의 Dense층으로 전체 합성곱 신경망을 구성하였다. **model.summary()**를 통해 각 뉴런층과 Output shape을 얻을 수 있다.

모델 컴파일

- 모델 컴파일 과정에서는 앞서 구현한 CNN의 손실함수와 옵티마이저를 설정한다.

```
from tensorflow.keras.optimizers import RMSprop

model.compile(loss='binary_crossentropy',
              optimizer=RMSprop(lr=0.001),
              metrics=['accuracy'])
```

지금까지 다루었던 예제와 다르게 손실함수로 **'binary_crossentropy'**를 사용하였다.

출력층의 활성화함수로 **'sigmoid'**를 사용하였다. 이것이 0,1로 분류되는 binary 분류 문제에 적합하기 때문이다.

옵티마이저로는 **RMSprop**를 사용하였다. RMSprop(Root Mean Square Propagation) 알고리즘은 훈련 과정 중에 학습률을 적절하게 변화시킨다.

이미지 데이터 전처리

- 훈련을 진행하기 전, tf.keras.preprocessing.image 모듈의 ImageDataGenerator 클래스를 이용해서 데이터 전처리를 진행한다.

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(rescale=1/255)

train_generator = train_datagen.flow_from_directory(
  './tmp/horse-or-human',
  target_size=(300, 300),
  batch_size=128,
  class_mode='binary'
)
```

rescale : 이미지 테이터에 곱해질 값을 설정한다.

ImageDataGenerator 클래스의 **flow_from_directory** 메서드는 이미지 데이터셋의 경로를 받아서 데이터의 배치를 만들어낸다.

모델 훈련

- fit() 메서드에 train_generator객체를 입력하고 훈련을 시작한다.

```
history = model.fit(
  train_generator,
  steps_per_epoch=8,
  epochs=15,
  verbose=1
)
```

Quiz

1. Image Generator 를 사용하여 이미지에 레이블을 지정하는 방법은?
    - 이미지가 포함된 디렉토리를 기반으로 한다.

2. Image Generator 에서 이미지를 정규화하는 데 사용되는 방법은?
    - rescale

3. 이미지의 훈련 크기를 어떻게 지정하였는가?
    - training generator의 target_size 매개변수

4. input_shape를 (300, 300, 3)으로 지정할 때 이는 무엇을 의미합니까?
    - 모든 이미지는 300x300 픽셀이며 색상을 정의하는 것은 3바이트이다.

5. Training data의 정확도는 1.000에 가깝지만 validation data는 그렇지 않다. 여기서 생기는 리스크는?
    - 훈련 데이터에 과적합하였다.