TF2.0 신경망 만들기

- CNN 신경망 이해
- 고양이와 개의 분류를 CNN을 이용하여 구현해 보기(2)

In [3]:

```
!pip install -q tensorflow-gpu==2.0.0-rc1
```

```
|████████████████████████████████| 380.5MB 45kB/s
|████████████████████████████████| 4.3MB 53.8MB/s
|████████████████████████████████| 501kB 56.7MB/s
```

In [0]:

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv2D, Flatten, Dropout, MaxPooling2D
from tensorflow.keras.preprocessing.image import ImageDataGenerator

import os
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:

```
print(tf.__version__)
```

```
2.0.0-rc1
```

학습 내용

- 앞에서 배운 내용
  - 데이터 로드
  - ImageGenerator를 생성
  - 모델 학습
  - 모델 평가

In [0]:

```
_URL = 'https://storage.googleapis.com/mledu-datasets/cats_and_dogs_filtered.zip'

path_to_zip = tf.keras.utils.get_file('cats_and_dogs.zip', origin=_URL, extract=True)

PATH = os.path.join(os.path.dirname(path_to_zip), 'cats_and_dogs_filtered')
```

In [0]:

```
## 경로 지정
train_dir = os.path.join(PATH, 'train')
validation_dir = os.path.join(PATH, 'validation')

train_cats_dir = os.path.join(train_dir, 'cats')  # directory with our training cat pictures
train_dogs_dir = os.path.join(train_dir, 'dogs')  # directory with our training dog pictures
validation_cats_dir = os.path.join(validation_dir, 'cats')  # directory with our validation cat
validation_dogs_dir = os.path.join(validation_dir, 'dogs')  # directory with our validation dog

## 경로에 이미지 데이터의 개수
num_cats_tr = len(os.listdir(train_cats_dir))
num_dogs_tr = len(os.listdir(train_dogs_dir))

num_cats_val = len(os.listdir(validation_cats_dir))
num_dogs_val = len(os.listdir(validation_dogs_dir))

total_train = num_cats_tr + num_dogs_tr
total_val = num_cats_val + num_dogs_val
```

In [5]:

```
print('total training cat images:', num_cats_tr)
print('total training dog images:', num_dogs_tr)

print('total validation cat images:', num_cats_val)
print('total validation dog images:', num_dogs_val)
print("--")
print("Total training images:", total_train)
print("Total validation images:", total_val)
```

```
total training cat images: 1000
total training dog images: 1000
total validation cat images: 500
total validation dog images: 500
--
Total training images: 2000
Total validation images: 1000
```

In [0]:

```
batch_size = 512
epochs = 15
IMG_HEIGHT = 150
IMG_WIDTH = 150
```

In [0]:

```
train_image_generator = ImageDataGenerator(rescale=1./255) # Generator for our training data
validation_image_generator = ImageDataGenerator(rescale=1./255) # Generator for our validation
```

In [8]:

```
train_data_gen = train_image_generator.flow_from_directory(batch_size=batch_size,
                                                           directory=train_dir,
                                                           shuffle=True,
                                                           target_size=(IMG_HEIGHT, IMG_WIDTH),
                                                           class_mode='binary')

val_data_gen = validation_image_generator.flow_from_directory(batch_size=batch_size,
                                                              directory=validation_dir,
                                                              target_size=(IMG_HEIGHT, IMG_WIDT
                                                              class_mode='binary')
```

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

In [9]:

```
model = Sequential([
    Conv2D(16, 3, padding='same', activation='relu', input_shape=(IMG_HEIGHT, IMG_WIDTH ,3)),
    MaxPooling2D(),
    Conv2D(32, 3, padding='same', activation='relu'),
    MaxPooling2D(),
    Conv2D(64, 3, padding='same', activation='relu'),
    MaxPooling2D(),
    Flatten(),
    Dense(512, activation='relu'),
    Dense(1, activation='sigmoid')
])

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

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d (Conv2D)              (None, 150, 150, 16)      448       
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 75, 75, 16)        0         
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 75, 75, 32)        4640      
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 37, 37, 32)        0         
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 37, 37, 64)        18496     
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 18, 18, 64)        0         
_________________________________________________________________
flatten (Flatten)            (None, 20736)             0         
_________________________________________________________________
dense (Dense)                (None, 512)               10617344  
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 513       
=================================================================
Total params: 10,641,441
Trainable params: 10,641,441
Non-trainable params: 0
_________________________________________________________________
```

In [10]:

```
### 모델 학습
%%time
history = model.fit_generator(
    train_data_gen,
    steps_per_epoch=total_train // batch_size,
    epochs=epochs,
    validation_data=val_data_gen,
    validation_steps=total_val // batch_size
)
```

```
Epoch 1/15
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow_core/pyt
hon/ops/math_grad.py:1394: where (from tensorflow.python.ops.array_ops) is depreca
ted and will be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
3/3 [==============================] - 13s 4s/step - loss: 1.3172 - accuracy: 0.52
49 - val_loss: 1.0547 - val_accuracy: 0.5000
Epoch 2/15
3/3 [==============================] - 6s 2s/step - loss: 0.8586 - accuracy: 0.514
8 - val_loss: 0.8129 - val_accuracy: 0.5000
Epoch 3/15
3/3 [==============================] - 6s 2s/step - loss: 0.7718 - accuracy: 0.499
3 - val_loss: 0.7148 - val_accuracy: 0.5000
Epoch 4/15
3/3 [==============================] - 6s 2s/step - loss: 0.7056 - accuracy: 0.490
6 - val_loss: 0.6959 - val_accuracy: 0.5000
Epoch 5/15
3/3 [==============================] - 6s 2s/step - loss: 0.6935 - accuracy: 0.504
0 - val_loss: 0.6925 - val_accuracy: 0.5508
Epoch 6/15
3/3 [==============================] - 6s 2s/step - loss: 0.6922 - accuracy: 0.520
8 - val_loss: 0.6919 - val_accuracy: 0.5371
Epoch 7/15
3/3 [==============================] - 6s 2s/step - loss: 0.6906 - accuracy: 0.566
4 - val_loss: 0.6896 - val_accuracy: 0.4980
Epoch 8/15
3/3 [==============================] - 6s 2s/step - loss: 0.6887 - accuracy: 0.514
1 - val_loss: 0.6848 - val_accuracy: 0.6562
Epoch 9/15
3/3 [==============================] - 6s 2s/step - loss: 0.6770 - accuracy: 0.679
0 - val_loss: 0.6703 - val_accuracy: 0.6133
Epoch 10/15
3/3 [==============================] - 6s 2s/step - loss: 0.6688 - accuracy: 0.606
2 - val_loss: 0.6613 - val_accuracy: 0.5527
Epoch 11/15
3/3 [==============================] - 6s 2s/step - loss: 0.6588 - accuracy: 0.584
0 - val_loss: 0.6579 - val_accuracy: 0.6348
Epoch 12/15
3/3 [==============================] - 6s 2s/step - loss: 0.6432 - accuracy: 0.639
8 - val_loss: 0.6422 - val_accuracy: 0.6523
Epoch 13/15
3/3 [==============================] - 6s 2s/step - loss: 0.6188 - accuracy: 0.681
6 - val_loss: 0.6316 - val_accuracy: 0.6523
Epoch 14/15
3/3 [==============================] - 6s 2s/step - loss: 0.5982 - accuracy: 0.704
2 - val_loss: 0.6169 - val_accuracy: 0.6523
Epoch 15/15
3/3 [==============================] - 6s 2s/step - loss: 0.5932 - accuracy: 0.680
```

```
3 - val_loss: 0.5973 - val_accuracy: 0.6992
CPU times: user 2min 32s, sys: 11.9 s, total: 2min 44s
Wall time: 1min 37s
```

In [11]:

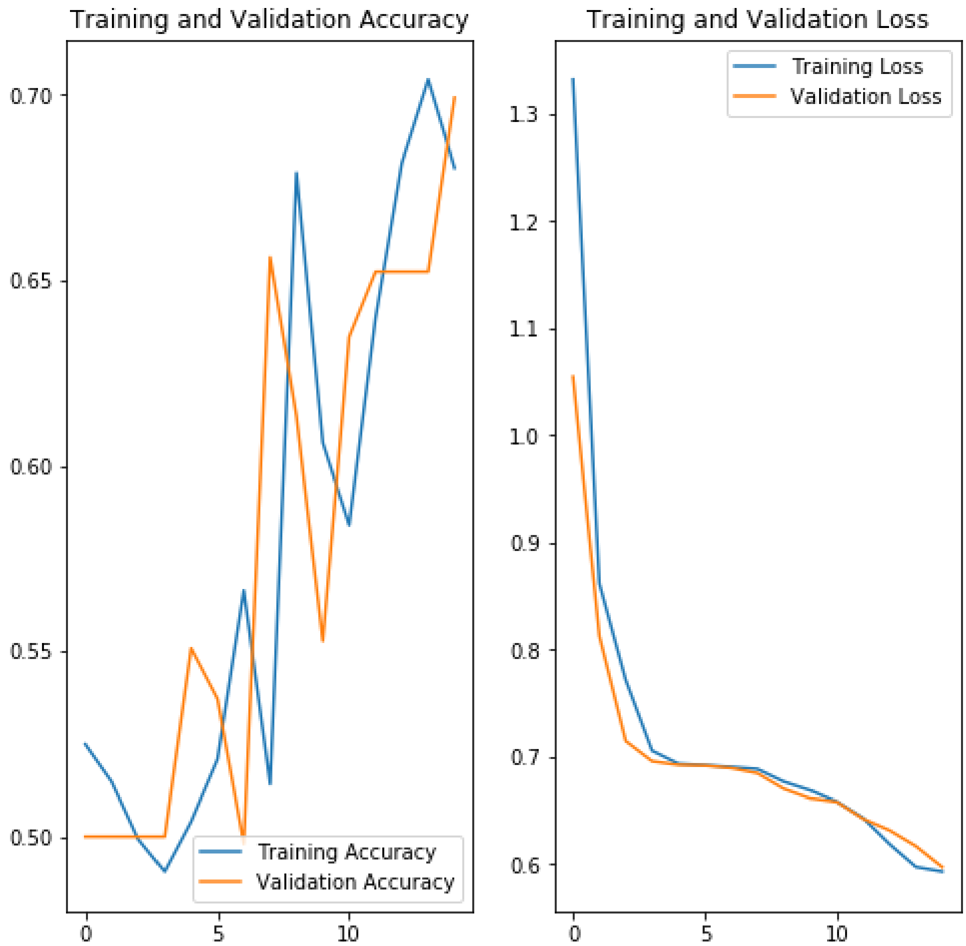
```
### 결과 시각화
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

epochs_range = range(epochs)

plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```



In [0]:

```

```