TF2.0 신경망 만들기

- CNN 신경망 이해
- 고양이와 개의 분류를 CNN을 이용하여 구현해 보기

환경

- Google Colab

In [5]:

```
# 기타 설치시
# !pip install -q tensorflow-gpu==2.0.0-rc1
```

In [6]:

```
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 [7]:

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

2.4.0

데이터 불러오기

- Kaggle의 필터링 된 버전의 Dogs vs Cats 데이터 세트를 사용

In [10]:

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

```
/root/.keras/datasets/cats_and_dogs.zip
/root/.keras/datasets/cats_and_dogs_filtered
```

```
cats_and_dogs_filtered
|__ train
    |_____ cats: [cat.0.jpg, cat.1.jpg, cat.2.jpg ....]
```

```
        |_____ dogs: [dog.0.jpg, dog.1.jpg, dog.2.jpg ...]
    |__ validation
        |_____ cats: [cat.2000.jpg, cat.2001.jpg, cat.2002.jpg ....]
        |_____ dogs: [dog.2000.jpg, dog.2001.jpg, dog.2002.jpg ...]
```

In [11]:

```
train_dir = os.path.join(PATH, 'train')           # 학습용
validation_dir = os.path.join(PATH, 'validation')  # 평가용
print(train_dir)
print(validation_dir)
```

```
/root/.keras/datasets/cats_and_dogs_filtered/train
/root/.keras/datasets/cats_and_dogs_filtered/validation
```

In [12]:

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

print("개 : ", train_dogs_dir)
print("고양이 : ", train_cats_dir)

validation_dogs_dir = os.path.join(validation_dir, 'dogs')  # directory with our validation dog pict
validation_cats_dir = os.path.join(validation_dir, 'cats')  # directory with our validation cat pict

print("개 : ", validation_dogs_dir)
print("고양이 : ", validation_cats_dir)
```

```
개 :  /root/.keras/datasets/cats_and_dogs_filtered/train/dogs
고양이 :  /root/.keras/datasets/cats_and_dogs_filtered/train/cats
개 :  /root/.keras/datasets/cats_and_dogs_filtered/validation/dogs
고양이 :  /root/.keras/datasets/cats_and_dogs_filtered/validation/cats
```

데이터 탐색

In [13]:

```
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 [14]:

```
## 이미지 개수
print('Total training cat images:', num_cats_tr)  # 고양이
print('Total training dog images:', num_dogs_tr)  # 개
print("--")

print('Total validation cat images:', num_cats_val)  # 고양이
print('Total validation dog images:', num_dogs_val)  # 개
print("--")

print("Total training images:", total_train)  #  학습용 = 개 + 고양이
print("Total validation images:", total_val)  #  평가용 = 개 + 고양이
```

```
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 [15]:

```
batch_size = 128
epochs = 15
IMG_HEIGHT = 150
IMG_WIDTH = 150
```

데이터 준비

- tf.keras에서 제공하는 ImageDataGenerator class
- 디스크에서 이미지를 읽고, 적절한 텐서로 사전 처리가 가능하다.

In [16]:

```
train_image_generator = ImageDataGenerator(rescale=1./255)      # 학습용 데이터 생성기
validation_image_generator = ImageDataGenerator(rescale=1./255) # 평가용 데이터 생성기
```

- 이미지 생성기를 정의한 후, flow_from_directory 메서드를 이용
  - 이미지를 로드
  - 이미지의 크기 조정 적용

In [17]:

```
print(train_dir) # 학습용
!ls /root/.keras/datasets/cats_and_dogs_filtered/train
```

```
/root/.keras/datasets/cats_and_dogs_filtered/train
cats  dogs
```

- directory : 디렉터리는 class를 포함하고 있어야 한다.
- target_size : 입력 이미지의 사이즈 지정

- class_mode : 2개의 이미지라면 binary, 그 이상의 이미지 categorical

In [18]:

```
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',
                                                           seed=42)
```

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

In [19]:

```
print(validation_dir) # 학습용
!ls /root/.keras/datasets/cats_and_dogs_filtered/validation
```

```
/root/.keras/datasets/cats_and_dogs_filtered/validation
cats  dogs
```

In [20]:

```
val_data_gen = validation_image_generator.flow_from_directory(batch_size=batch_size,
                                                              directory=validation_dir,
                                                              target_size=(IMG_HEIGHT, IMG_WIDTH),
                                                              class_mode='binary',
                                                              seed=42)
```

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

이미지 추출 후, 이에 대한 시각화

In [21]:

```
sample_training_images, _ = next(train_data_gen)
sample_training_images.shape  # 이미지 추출
```

Out[21]:

```
(128, 150, 150, 3)
```

In [22]:

```
# 이 함수는 이미지를 plot를 하는 함수.
def plotImages(images_arr):
    fig, axes = plt.subplots(1, 5, figsize=(20,20))
    axes = axes.flatten()
    for img, ax in zip( images_arr, axes):
        ax.imshow(img)
        ax.axis('off')
    plt.tight_layout()
    plt.show()
```

In [23]:

```
plotImages(sample_training_images[:5])
```

In [24]:

```
sample_training_images, _ = next(train_data_gen)
plotImages(sample_training_images[:5])
```

모델 만들기(Create the model)

- 개 고양이 분류(이진 분류) : 마지막 뉴런 1개(sigmoid)
- MNIST 분류 : 뉴런 10개(softmax)

In [25]:

```
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')
])
```

모델 컴파일(Compile the model)

- binary_crossentropy : label이 두개
- categorical_crossentropy : label이 여러개

In [26]:

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

In [27]:

```
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
_________________________________________________________________
```

모델 훈련시키기

- model.fit() : 적은 데이터 셋
- model.fit_generator() : 중복 데이터를 피하려고 할때 사용. 클 데이터 셋을 이용 할 때 사용.
  - 여기에서는 ImageDataGenerator의 fit_generator를 사용한다.

In [28]:

```
# batch_size = 128  # 배치 사이즈
# epochs = 15       # 전체 데이터 학습 횟수
print("Total training images:", total_train)  #  학습용 = 개 + 고양이
print("Total validation images:", total_val)  #  평가용 = 개 + 고양이
```

```
Total training images: 2000
Total validation images: 1000
```

In [29]:

```
%%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
)
```

```
/usr/local/lib/python3.6/dist-packages/tensorflow/python/keras/engine/training.py:18
44: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future
version. Please use `Model.fit`, which supports generators.
  warnings.warn('`Model.fit_generator` is deprecated and '
```

```
Epoch 1/15
15/15 [==============================] - 15s 559ms/step - loss: 1.4655 - accuracy:
0.5010 - val_loss: 0.6897 - val_accuracy: 0.5067
Epoch 2/15
15/15 [==============================] - 8s 529ms/step - loss: 0.6889 - accuracy: 0.
5197 - val_loss: 0.6763 - val_accuracy: 0.5670
Epoch 3/15
15/15 [==============================] - 8s 518ms/step - loss: 0.6728 - accuracy: 0.
5871 - val_loss: 0.6542 - val_accuracy: 0.6105
Epoch 4/15
15/15 [==============================] - 8s 511ms/step - loss: 0.6323 - accuracy: 0.
6447 - val_loss: 0.6112 - val_accuracy: 0.6596
Epoch 5/15
15/15 [==============================] - 8s 513ms/step - loss: 0.5729 - accuracy: 0.
7020 - val_loss: 0.6072 - val_accuracy: 0.6942
Epoch 6/15
15/15 [==============================] - 8s 524ms/step - loss: 0.5419 - accuracy: 0.
7356 - val_loss: 0.6162 - val_accuracy: 0.6596
Epoch 7/15
15/15 [==============================] - 8s 524ms/step - loss: 0.5203 - accuracy: 0.
7393 - val_loss: 0.5699 - val_accuracy: 0.6942
Epoch 8/15
15/15 [==============================] - 8s 518ms/step - loss: 0.4473 - accuracy: 0.
7949 - val_loss: 0.5696 - val_accuracy: 0.6987
Epoch 9/15
15/15 [==============================] - 8s 528ms/step - loss: 0.4260 - accuracy: 0.
8089 - val_loss: 0.6210 - val_accuracy: 0.7065
Epoch 10/15
15/15 [==============================] - 8s 519ms/step - loss: 0.3604 - accuracy: 0.
8423 - val_loss: 0.6148 - val_accuracy: 0.7009
Epoch 11/15
15/15 [==============================] - 8s 521ms/step - loss: 0.3323 - accuracy: 0.
8463 - val_loss: 0.6684 - val_accuracy: 0.6819
Epoch 12/15
15/15 [==============================] - 8s 515ms/step - loss: 0.3527 - accuracy: 0.
8389 - val_loss: 0.6708 - val_accuracy: 0.7009
Epoch 13/15
15/15 [==============================] - 8s 511ms/step - loss: 0.2605 - accuracy: 0.
8931 - val_loss: 0.6534 - val_accuracy: 0.7054
Epoch 14/15
15/15 [==============================] - 8s 516ms/step - loss: 0.2546 - accuracy: 0.
```

```
9021 - val_loss: 0.6945 - val_accuracy: 0.7020
Epoch 15/15
15/15 [==============================] - 8s 520ms/step - loss: 0.2007 - accuracy: 0.
9294 - val_loss: 0.7468 - val_accuracy: 0.7121
CPU times: user 2min 9s, sys: 8.01 s, total: 2min 17s
Wall time: 2min 2s
```

학습 모델 결과 시각화

In [30]:

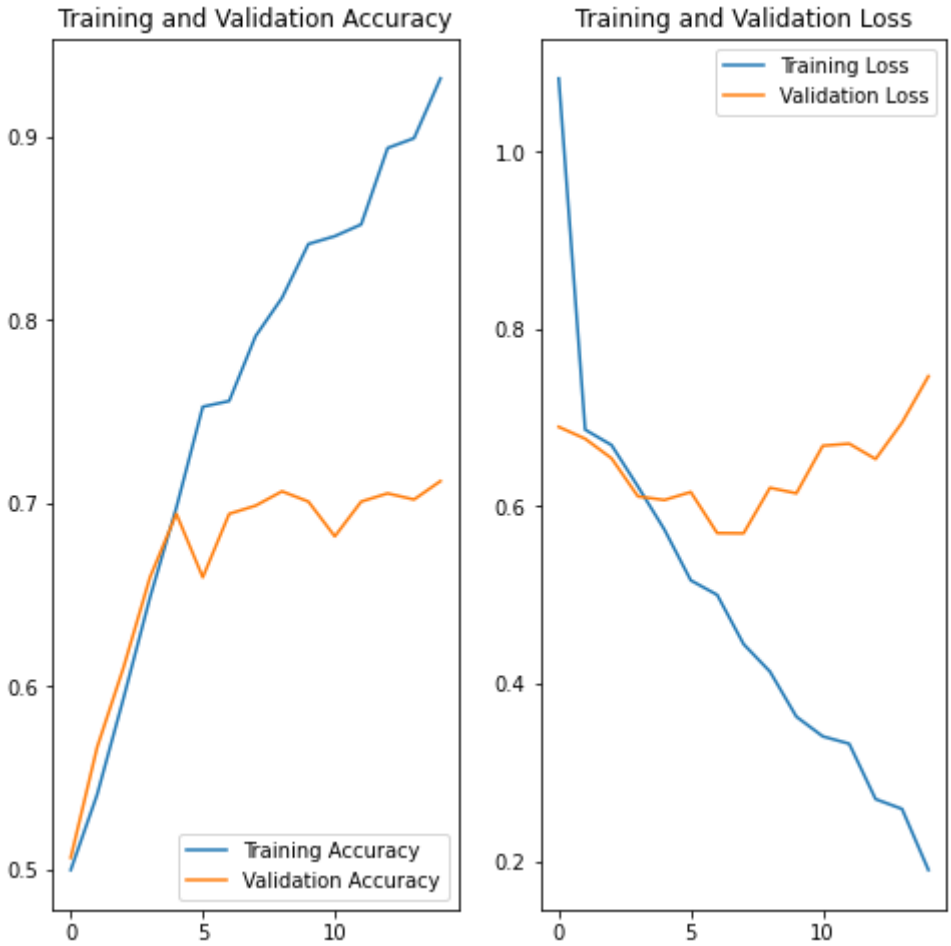
```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

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()
```



REF

- 이미지 분류 : https://www.tensorflow.org/tutorials/images/classification (https://www.tensorflow.org/tutorials/images/classification)

- ImageGenerator Class : https://medium.com/@vijayabhaskar96/tutorial-image-classification-with-keras-flow-from-directory-and-generators-95f75ebe5720 (https://medium.com/@vijayabhaskar96/tutorial-image-classification-with-keras-flow-from-directory-and-generators-95f75ebe5720)
- Conv2D : https://www.tensorflow.org/api_docs/python/tf/keras/layers/Conv2D (https://www.tensorflow.org/api_docs/python/tf/keras/layers/Conv2D)