Face Mask

Detection - VGG16

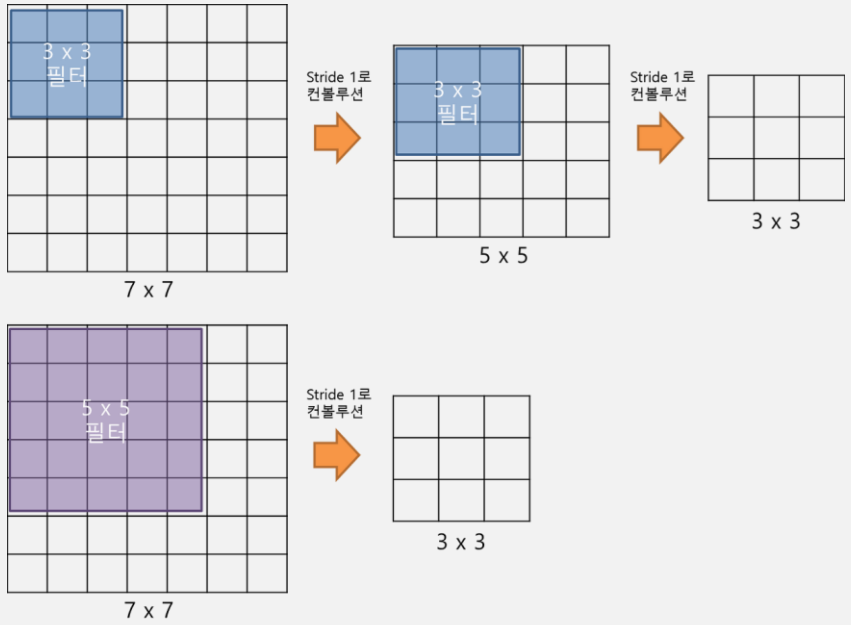
AM20216801

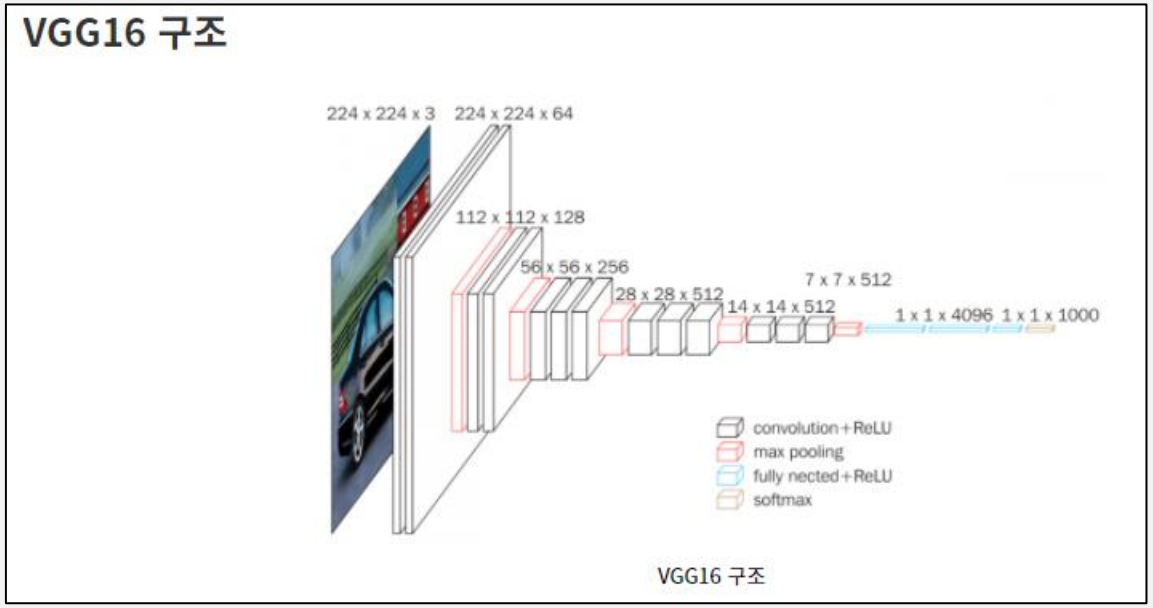
JANG CHEOL
HEE

CONTENTS

VGG16

- ✔ VGGNet is a model developed by VGG, a research team at Oxford University, and is a model that won the 2014 ImageNet image recognition contest.
- ✔ VGGNet represents a model with 16 or 19 layers.
- ✔ VGG16 consists of 13 convolutional layers and 3 fully connected layers.





Library

- numpy
- keras
- shutil
- os
- random
- matplotlib

```
#importing the necessary libraries
import numpy as np
import keras
import shutil
import os
import random
import matplotlib.pyplot as plt
%matplotlib inline

from keras.preprocessing.image import ImageDataGenerator
```

```
[ ]  from google.colab import drive
     drive.mount('/content/drive')
```

Data set

- 209 mask images and 132 no mask images were used.
- Selected photos randomly and classified train data and test data.
- Used ImageDataGenerator to preprocess by dividing by 255 and augmentation the image.
- Data using flow_from_directory
  Batch, Target Size Class Mode Adjustment “Categorical”.

```
[ ]  #assigning image paths to variables
     mask_data = "/content/sample_data/Mask/"
     no_mask_data = "/content/sample_data/No Mask/"

[ ]  total_mask_images = os.listdir(mask_data)
     print("no of mask images:: {}".format(len(total_mask_images)))
     total_nonmask_images = os.listdir(no_mask_data)
     print("no of non-mask images:: {}".format(len(total_nonmask_images)))

     no of mask images:: 209
     no of non-mask images:: 132
```

```
for images in random.sample(total_mask_images,100):
    shutil.copy(mask_data+images, '/content/train/mask')
for images in random.sample(total_mask_images,30):
    shutil.copy(mask_data+images, '/content/test/mask')
for images in random.sample(total_nonmask_images,100):
    shutil.copy(no_mask_data+images, '/content/train/no mask')
for images in random.sample(total_nonmask_images,30):
    shutil.copy(no_mask_data+images, '/content/test/no mask')

[ ] train_batch = ImageDataGenerator(rescale=1./255, zoom_range=0.2, horizontal_flip=True, vertical_flip=True, shear_range=0.2).#
            flow_from_directory('./train', target_size=(224,224), batch_size=32, class_mode = 'categorical')
    test_batch = ImageDataGenerator(rescale=1./255).#
            flow_from_directory('./test', target_size = (224,224), batch_size=32, class_mode='categorical')

    Found 197 images belonging to 3 classes.
    Found 60 images belonging to 3 classes.
```

```
[ ]  train_batch.class_indices

     {'.ipynb_checkpoints': 0, 'mask': 1, 'no mask': 2}

[ ]  class_mask = ['miss','mask', 'no mask']
```

Import model

```
[ ]  #import vgg16
     from keras.applications.vgg16 import VGG16

     #vgg16 accepts image size (224,224) only
     IMAZE_SIZE = [224,224]
     vgg = VGG16(input_shape=IMAZE_SIZE+[3], weights='imagenet', include_top=False)

     Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
     58892288/58889256 [==============================] - 2s 0us/step
     58900480/58889256 [==============================] - 2s 0us/step
```

```
[ ]  for layers in vgg.layers:
         layers.trainable = False

     vgg.summary()
```

```
vgg.summary()

Model: "vgg16"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 224, 224, 3)]     0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
=================================================================
Total params: 14,714,688
Trainable params: 0
Non-trainable params: 14,714,688
_________________________________________________________________
```

import vgg16 and input_shape = [224,224,3],
weights = “imagenet”,
With include_top = False, not to import fully connected layers.

layers.trainable = False to prevent it from being updated.

Total params became non-trainable params.

Model

- Create a flatten layer.
- The output of the dense layer is three activations = “softmax”.
- A flatten layer and a dense layer are added to the existing model

```
[ ] flatten_layer = keras.layers.Flatten()(vgg.output)
    prediction_layer = keras.layers.Dense(3, activation='softmax')(flatten_layer)

[ ] model = keras.models.Model(inputs = vgg.input, outputs = prediction_layer)

    model.summary()

    Model: "model"
    _________________________________________________________________
    Layer (type)                 Output Shape              Param #
    =================================================================
    input_1 (InputLayer)         [(None, 224, 224, 3)]     0
    _________________________________________________________________
    block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
    _________________________________________________________________
    block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
    _________________________________________________________________
    block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
    _________________________________________________________________
    block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
    _________________________________________________________________
    block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
    _________________________________________________________________
    block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
    _________________________________________________________________
    block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
    _________________________________________________________________
    block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
    _________________________________________________________________
    block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
    _________________________________________________________________
    block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
    _________________________________________________________________
    block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
    _________________________________________________________________
    block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
    _________________________________________________________________
    block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
    _________________________________________________________________
    block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
    _________________________________________________________________
    block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
    _________________________________________________________________
    block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
    _________________________________________________________________
    block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
    _________________________________________________________________
    block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
    _________________________________________________________________
    flatten (Flatten)            (None, 25088)             0
    _________________________________________________________________
    dense (Dense)                (None, 3)                 75267
    =================================================================
    Total params: 14,789,955
    Trainable params: 75,267
    Non-trainable params: 14,714,688
    _________________________________________________________________
```

Model

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

     r = model.fit_generator(train_batch, validation_data=test_batch, epochs=5, steps_per_epoch=len(train_batch), validation_steps=len(test_batch))

     /usr/local/lib/python3.7/dist-packages/keras/engine/training.py:1972: 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 '
     /usr/local/lib/python3.7/dist-packages/PIL/TiffImagePlugin.py:788: UserWarning: Corrupt EXIF data.  Expecting to read 4 bytes but only got 0.
       warnings.warn(str(msg))
     Epoch 1/5
     7/7 [==============================] - 56s 3s/step - loss: 0.9788 - accuracy: 0.5127 - val_loss: 0.4102 - val_accuracy: 0.8167
     Epoch 2/5
     7/7 [==============================] - 7s 1s/step - loss: 0.4650 - accuracy: 0.7716 - val_loss: 0.1781 - val_accuracy: 0.9333
     Epoch 3/5
     7/7 [==============================] - 7s 1s/step - loss: 0.3088 - accuracy: 0.9036 - val_loss: 0.1741 - val_accuracy: 0.9500
     Epoch 4/5
     7/7 [==============================] - 7s 1s/step - loss: 0.1740 - accuracy: 0.9442 - val_loss: 0.0788 - val_accuracy: 0.9833
     Epoch 5/5
     7/7 [==============================] - 7s 1s/step - loss: 0.1254 - accuracy: 0.9645 - val_loss: 0.0518 - val_accuracy: 0.9833
```

Use optimizer = “adam”.

Use loss = “categorical_crossentropy”.

metrics = [“accuracy”]

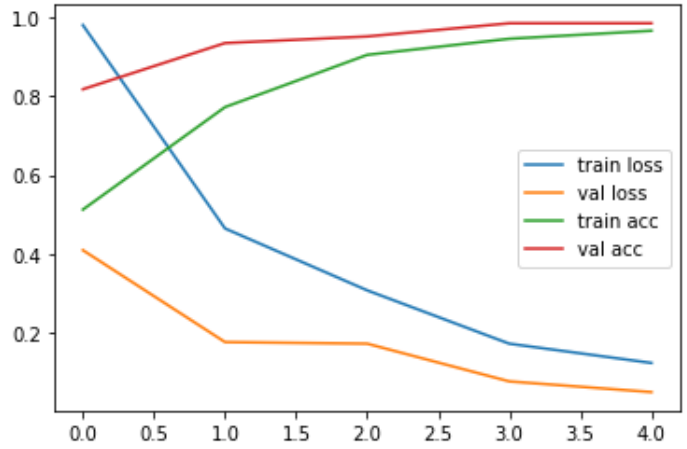
- loss : training loss value
- acc : training accuracy
- val_loss : validation loss value
- val_acc : verification accuracy

Result

```
plt.plot(r.history['loss'], label = 'train loss')
plt.plot(r.history['val_loss'], label='val loss')
plt.legend()


plt.plot(r.history['accuracy'], label = 'train acc')
plt.plot(r.history['val_accuracy'], label='val acc')
plt.legend()
```

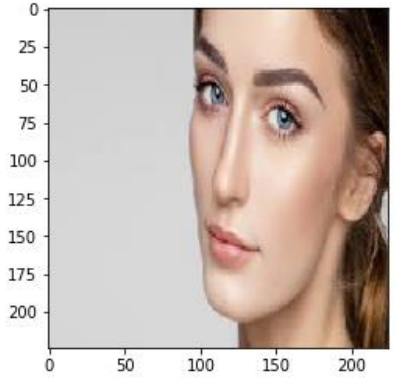
```
<matplotlib.legend.Legend at 0x7fc1c59fca10>
```




```
[ ] from keras.preprocessing import image
    from keras.applications.imagenet_utils import preprocess_input
```

```
img = image.load_img('/content/test/no mask/No Mask112.jpg', target_size=(224,224))
x=image.img_to_array(img)
x = np.expand_dims(x,0)
y = preprocess_input(x)
pred = class_mask[np.argmax(model.predict(y))]
print(pred)
plt.imshow(img)
```

```
no mask
<matplotlib.image.AxesImage at 0x7fc2322c9850>
```



Conclusion

- After classifying whether or not a mask was used with vgg16, approach with 0.1254 loss and 0.9645 accuracy at 5 epoch.

Reference

- https://bskyvision.com/504
- https://www.kaggle.com/arkajitmajumder/face-mask-detection/notebook
- https://eremo2002.tistory.com/57?category=779320