Image Classification Using Densenet-121

Yunkyeong Heo

AM20216804

Index

01 Densenet

02 Yoga Pose Image Classification

03 Library

04 Data

- DataSet
- Data Analysis
- Data classification

05 Model

06 Result and Evaluation

07 Conclusion

Densenet

- As the layer deepens in CNN, the error is greatly reduced, resulting in a phenomenon in which learning is not possible.

" Apply resnet skip-connection to solve this problem "

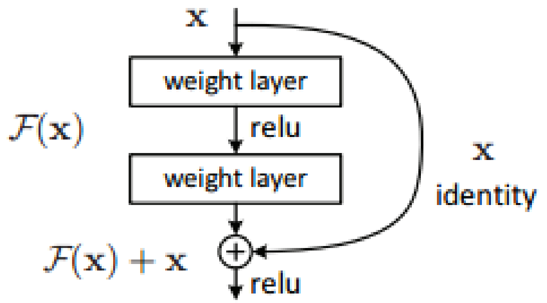

x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 2. Residual learning: a building block.

- However, in the case of resnet, initial information may not be clearly transmitted because it proceeds through an operation that adds input x to output f(x).

Densenet

$$\mathbf{x}_\ell = H_\ell(\mathbf{x}_{\ell-1}) + \mathbf{x}_{\ell-1}.$$

- The densenet utilizes a connection different from resnet though the concept of dense connectivity.

- Unlike skip-connection, a direct connection is performed for all consecutive layers.

- A method of concating at the channel level rather than the + operation when merging feature map information

$$\mathbf{x}_\ell = H_\ell([\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{\ell-1}]),$$

- In this connection, the previous information and the current information are not mixed because the feature map remains intact.

Densenet

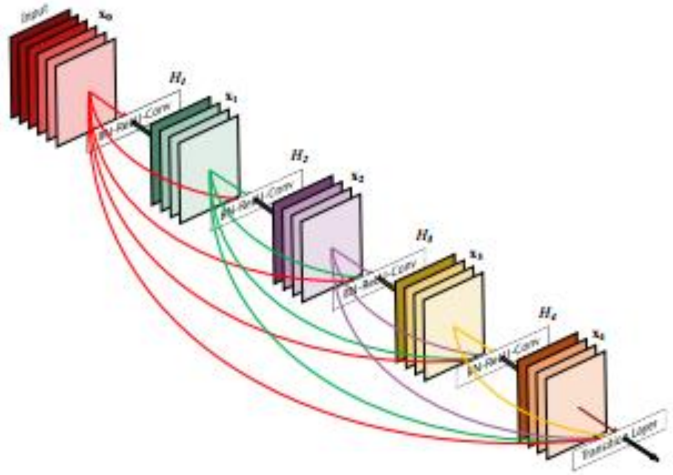
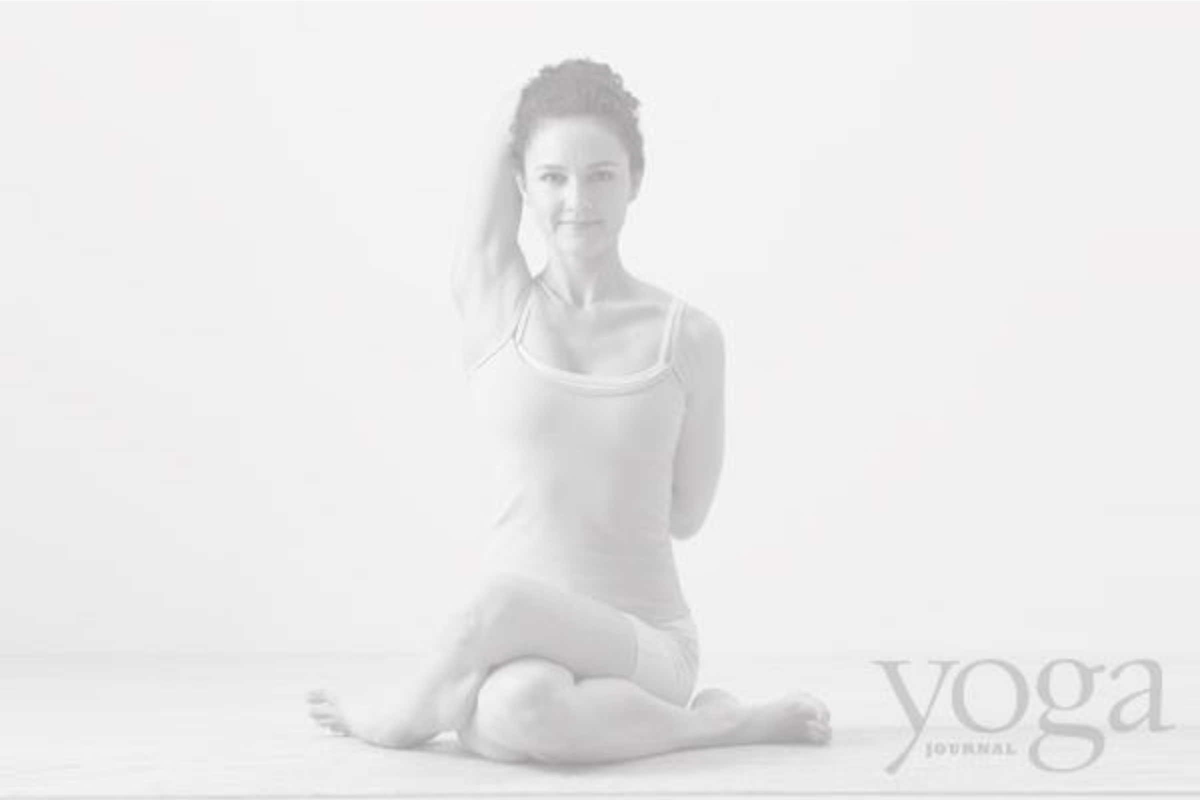


Figure 1: A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.

- It can be seen that dense connectivity is concatenated into feature maps of sequentially connected layers in which feature maps of each stage are concatenated as shown in the figure above.

Yoga Pose Image Classification



Classify the yoga pose being performed in the image

Library

- matplotlib

  A library for drawing data, chart, or plots in Python
- cv2

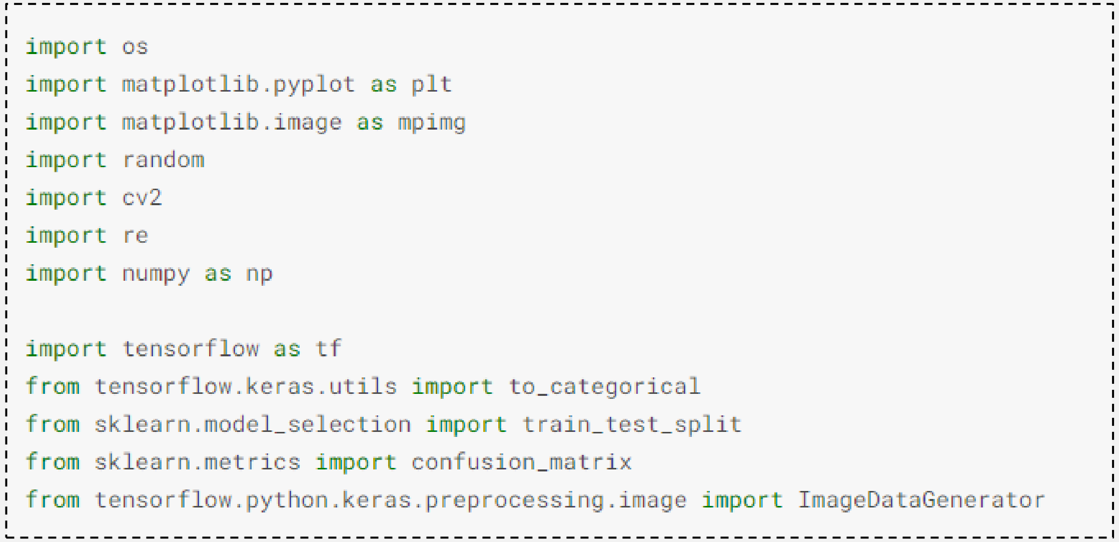
  Open source library used for real-time image/video processing
- Re

  Regular Expression
- numpy

  A linear algebra library that performs numerical operations such as vectors and matrices
- tensorflow

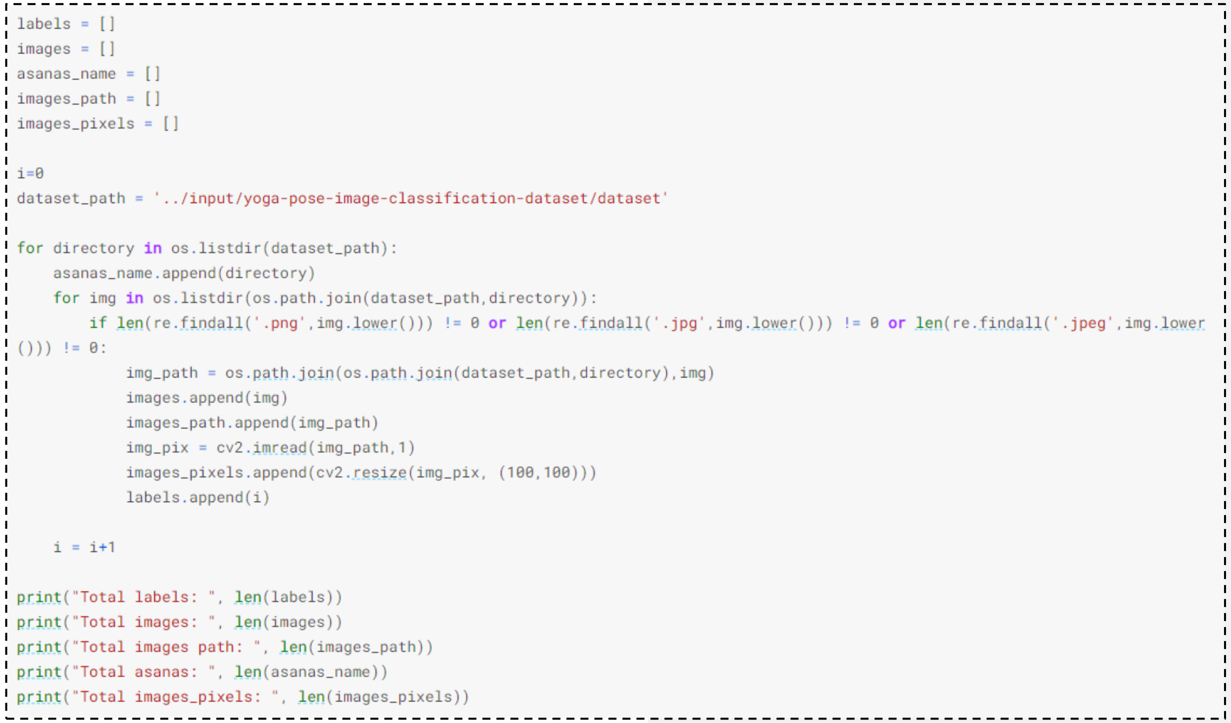
  A library that provides various functions for easy implementation of deep learning programs
- sklearn

  Machine learning library

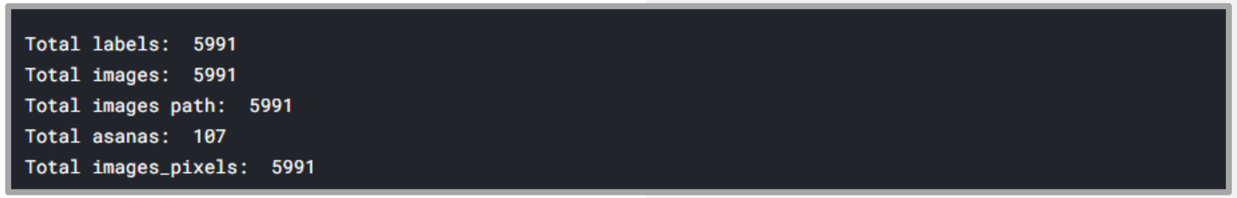


```
import os
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import random
import cv2
import re
import numpy as np

import tensorflow as tf
from tensorflow.keras.utils import to_categorical
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from tensorflow.python.keras.preprocessing.image import ImageDataGenerator
```

DataSet



```
labels = []
images = []
asanas_name = []
images_path = []
images_pixels = []

i=0
dataset_path = '../input/yoga-pose-image-classification-dataset/dataset'

for directory in os.listdir(dataset_path):
    asanas_name.append(directory)
    for img in os.listdir(os.path.join(dataset_path,directory)):
        if len(re.findall('.png',img.lower())) != 0 or len(re.findall('.jpg',img.lower())) != 0 or len(re.findall('.jpeg',img.lower
())) != 0:
            img_path = os.path.join(os.path.join(dataset_path,directory),img)
            images.append(img)
            images_path.append(img_path)
            img_pix = cv2.imread(img_path,1)
            images_pixels.append(cv2.resize(img_pix, (100,100)))
            labels.append(i)

    i = i+1

print("Total labels: ", len(labels))
print("Total images: ", len(images))
print("Total images path: ", len(images_path))
print("Total asanas: ", len(asanas_name))
print("Total images_pixels: ", len(images_pixels))
```



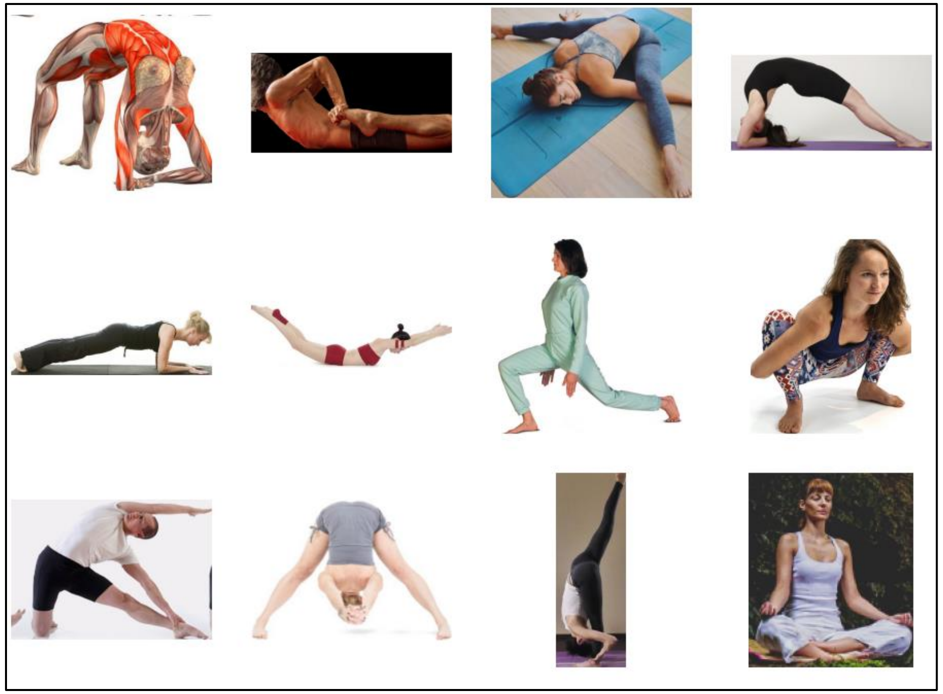
```
Total labels:  5991
Total images:  5991
Total images path:  5991
Total asanas:  107
Total images_pixels:  5991
```

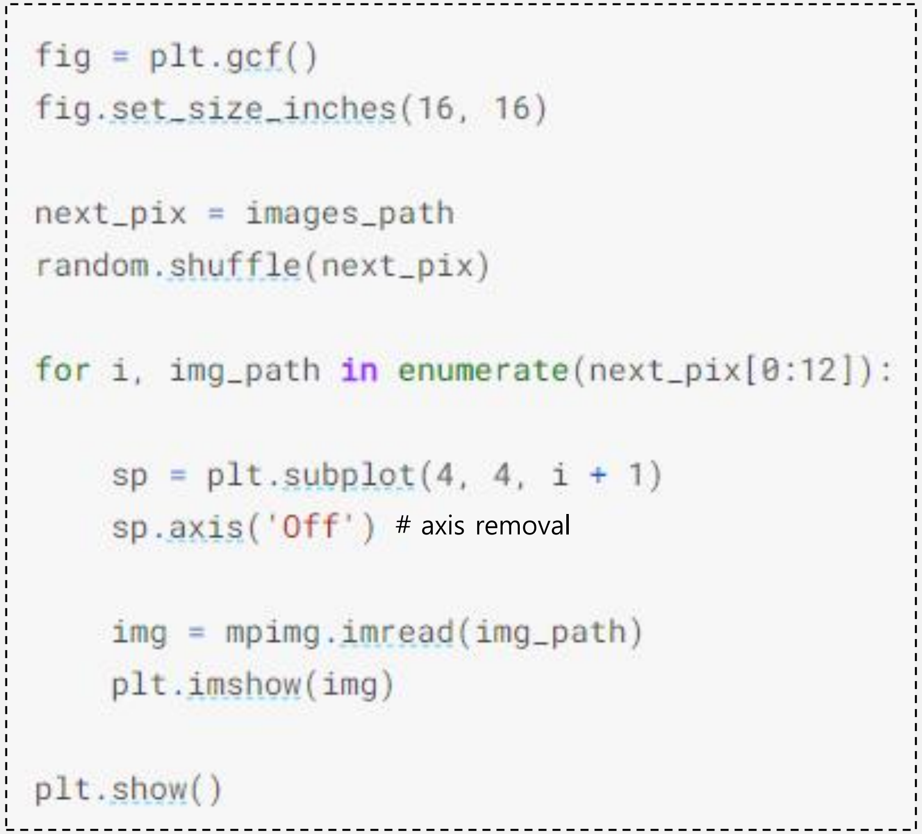
Data Analysis

- Subplot(rows, columns, index)

  Indicates the total number of rows and columns in the picture, and indicates the index of the image to display.

- 4 rows and columns are created and 12 images loaded





```
fig = plt.gcf()
fig.set_size_inches(16, 16)

next_pix = images_path
random.shuffle(next_pix)

for i, img_path in enumerate(next_pix[0:12]):

    sp = plt.subplot(4, 4, i + 1)
    sp.axis('Off') # axis removal

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

plt.show()
```

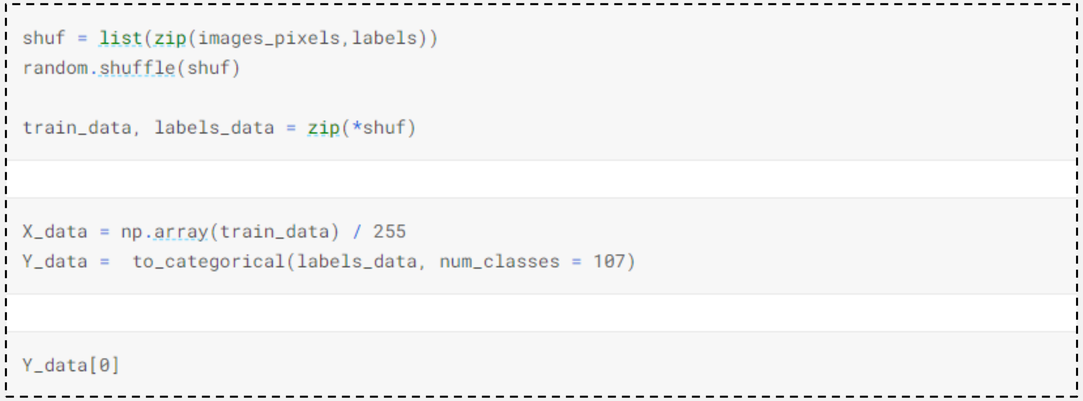
Data classification

- to_categorical

  One-hot encoding function (convert decimal integer format to binary format)

- Create an array of size 107 and put 1(hot) in the label_data value position

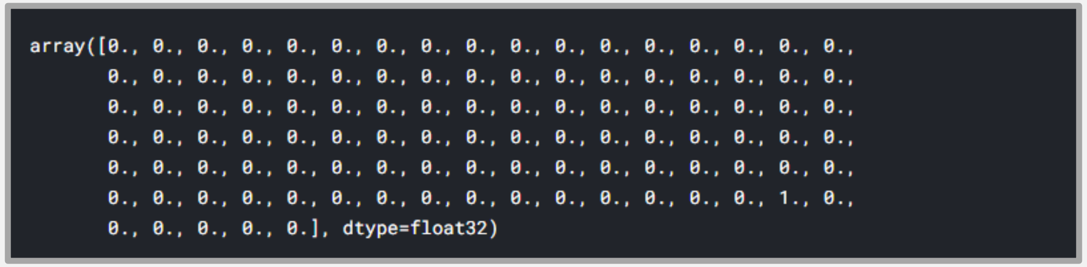
- Create X_data, Y_data shape



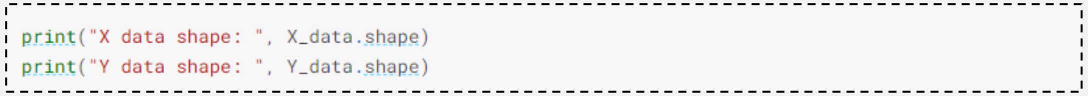
```
shuf = list(zip(images_pixels,labels))
random.shuffle(shuf)

train_data, labels_data = zip(*shuf)
```

```
X_data = np.array(train_data) / 255
Y_data =  to_categorical(labels_data, num_classes = 107)
```

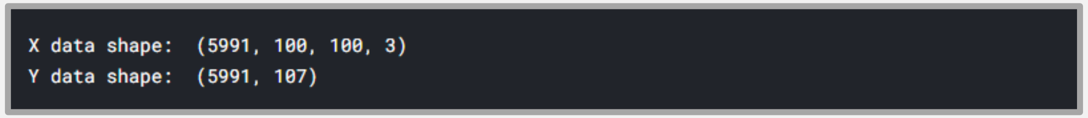
```
Y_data[0]
```



```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0.,
       0., 0., 0., 0., 0.], dtype=float32)
```



```
print("X data shape: ", X_data.shape)
print("Y data shape: ", Y_data.shape)
```



```
X data shape:  (5991, 100, 100, 3)
Y data shape:  (5991, 107)
```

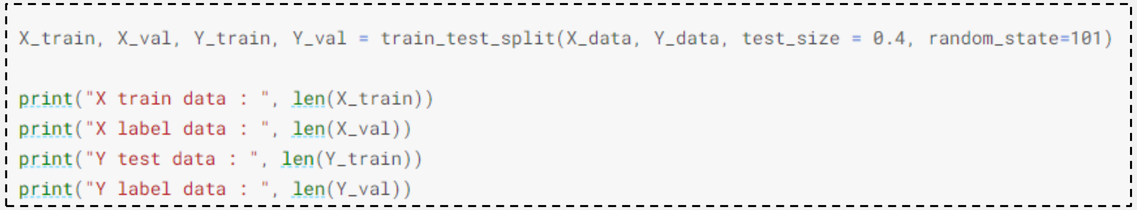
Data classification

- Separate the training data set and the data set

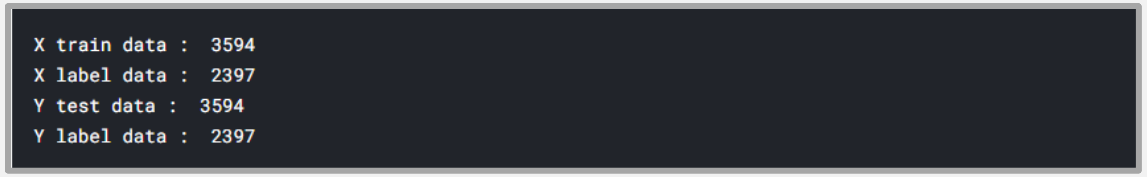
  Designate 40% of the entire data set as the test(validation)set

- ImageDataGenerator(Image augmentation)

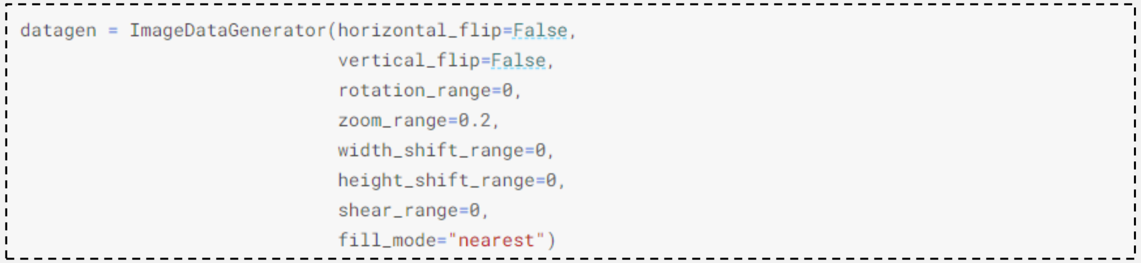
  A method of increasing the amount of learning data by gradually transforming the learning data such as rotation, enlargement, movement and inversion



```
X_train, X_val, Y_train, Y_val = train_test_split(X_data, Y_data, test_size = 0.4, random_state=101)

print("X train data : ", len(X_train))
print("X label data : ", len(X_val))
print("Y test data : ", len(Y_train))
print("Y label data : ", len(Y_val))
```



```
X train data :  3594
X label data :  2397
Y test data :  3594
Y label data :  2397
```



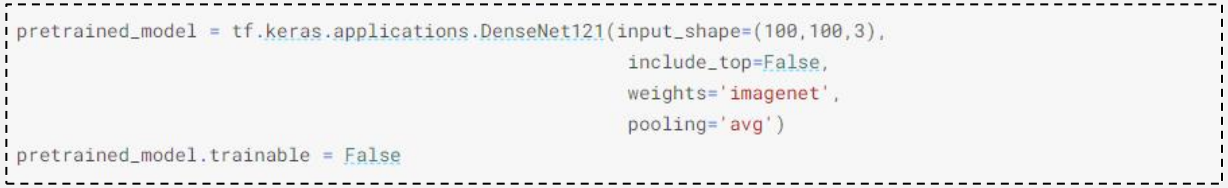
```
datagen = ImageDataGenerator(horizontal_flip=False,
                             vertical_flip=False,
                             rotation_range=0,
                             zoom_range=0.2,
                             width_shift_range=0,
                             height_shift_range=0,
                             shear_range=0,
                             fill_mode="nearest")
```

Model

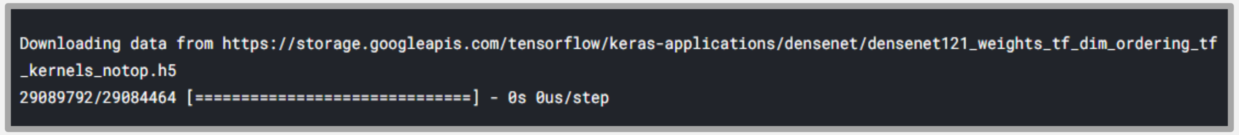
- By finely adjusting the weights of the pre-learning model with four parameters, after analyzing the type and total number of new data to be classified in advance, based on that, only part of the pre-learning model weights can be retrained or the weights can be re-learned from the beginning. can

- If we look at each parameter here,
  - weights is a dataset use for pre-training, and imagenet is usually used
  - If include_top = False, it means that only feature extractors of the pre-trained model are imported and classifiers are not imported.
  - input_shape represents the size of the image tensor to be newly trained.
  - pooling by average

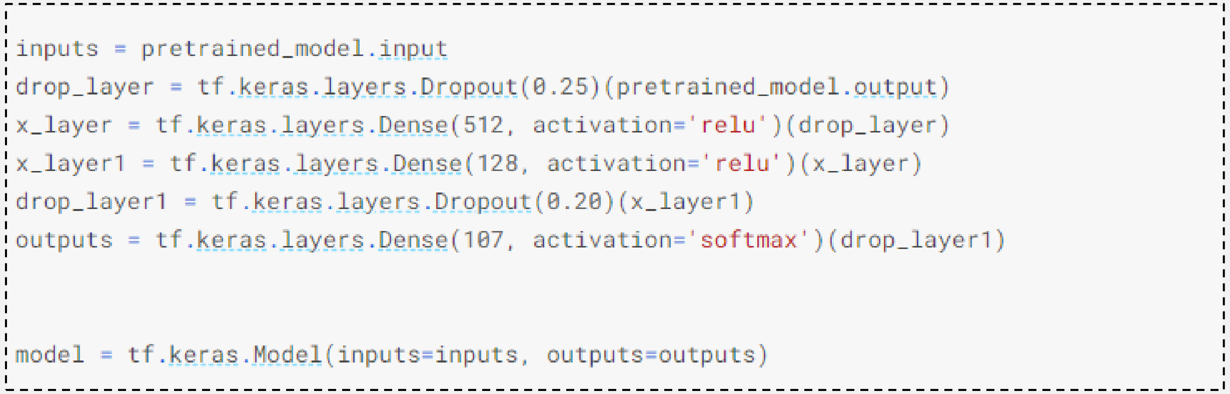
- Define the loss, optimizer, and metric through the compilation function and repeat 20 times through the fit



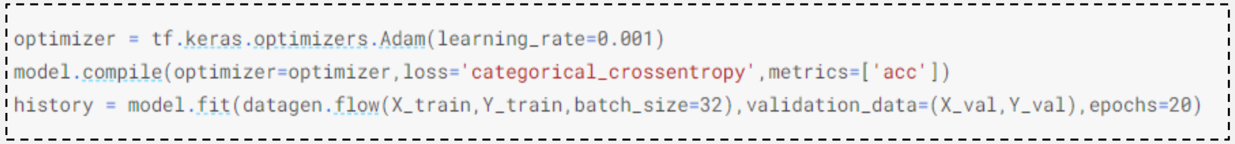
```
pretrained_model = tf.keras.applications.DenseNet121(input_shape=(100,100,3),
                                                     include_top=False,
                                                     weights='imagenet',
                                                     pooling='avg')
pretrained_model.trainable = False
```



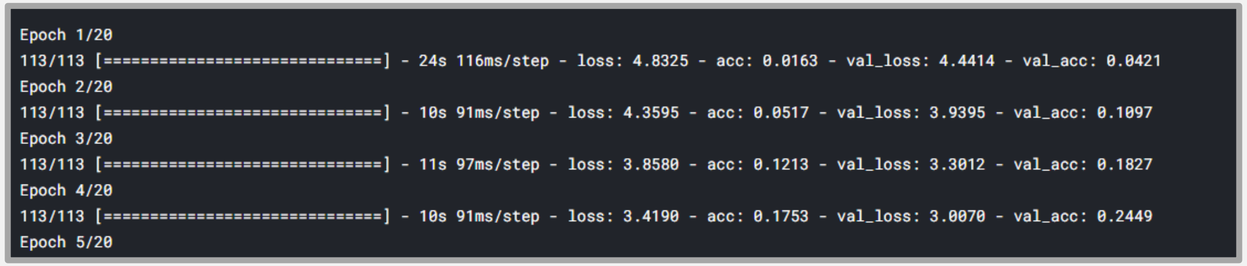
```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/densenet/densenet121_weights_tf_dim_ordering_tf
_kernels_notop.h5
29089792/29084464 [==============================] - 0s 0us/step
```



```
inputs = pretrained_model.input
drop_layer = tf.keras.layers.Dropout(0.25)(pretrained_model.output)
x_layer = tf.keras.layers.Dense(512, activation='relu')(drop_layer)
x_layer1 = tf.keras.layers.Dense(128, activation='relu')(x_layer)
drop_layer1 = tf.keras.layers.Dropout(0.20)(x_layer1)
outputs = tf.keras.layers.Dense(107, activation='softmax')(drop_layer1)


model = tf.keras.Model(inputs=inputs, outputs=outputs)
```



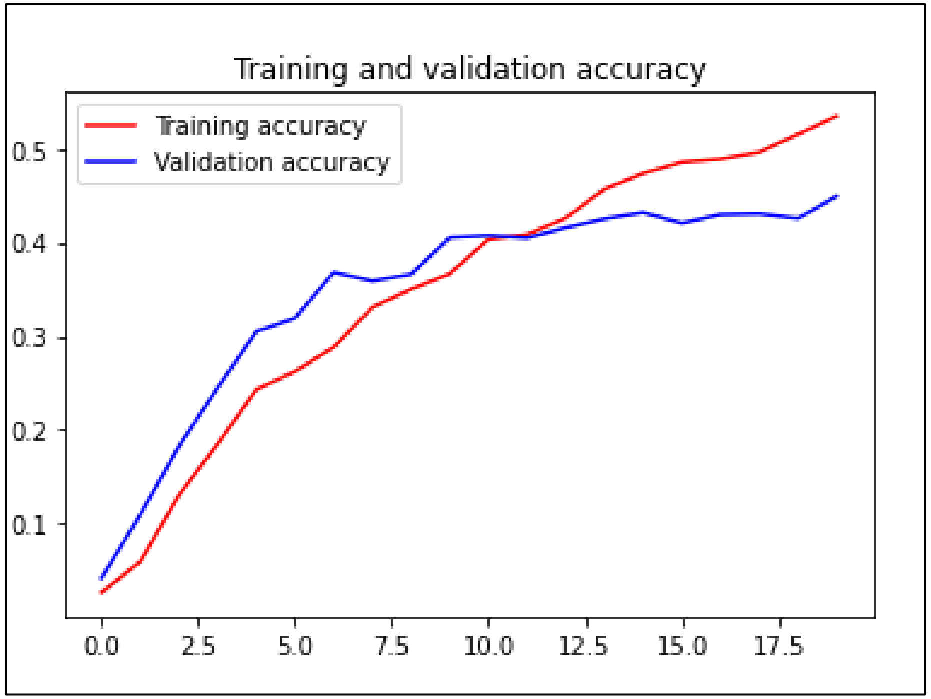
```
optimizer = tf.keras.optimizers.Adam(learning_rate=0.001)
model.compile(optimizer=optimizer,loss='categorical_crossentropy',metrics=['acc'])
history = model.fit(datagen.flow(X_train,Y_train,batch_size=32),validation_data=(X_val,Y_val),epochs=20)
```

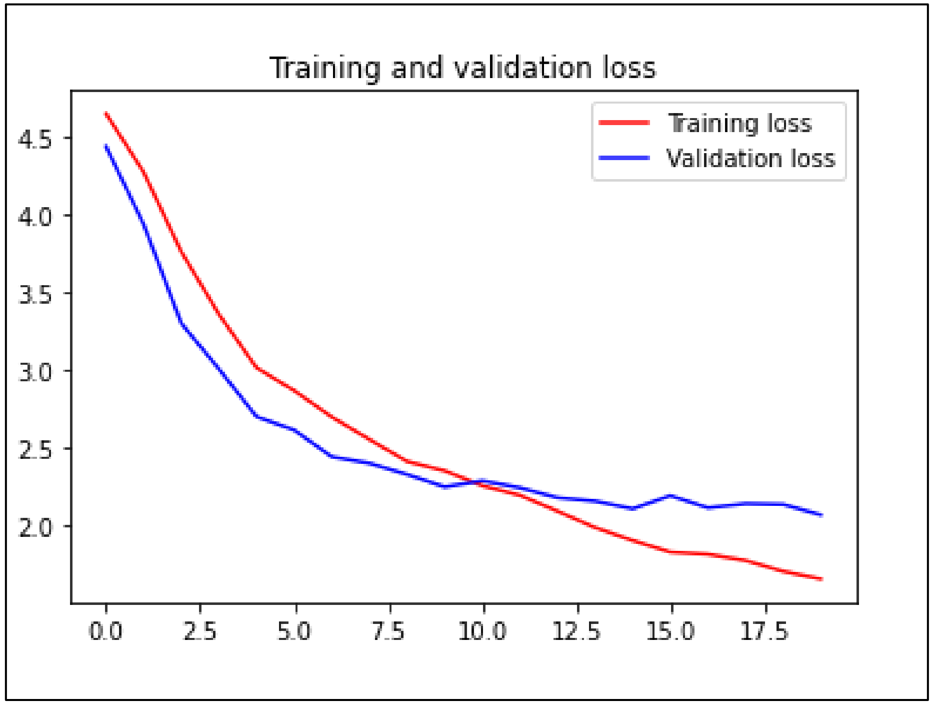


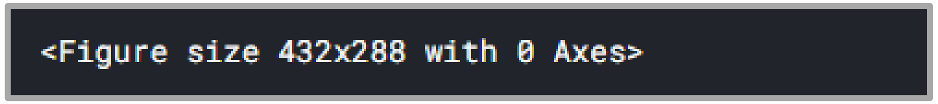
```
Epoch 1/20
113/113 [==============================] - 24s 116ms/step - loss: 4.8325 - acc: 0.0163 - val_loss: 4.4414 - val_acc: 0.0421
Epoch 2/20
113/113 [==============================] - 10s 91ms/step - loss: 4.3595 - acc: 0.0517 - val_loss: 3.9395 - val_acc: 0.1097
Epoch 3/20
113/113 [==============================] - 11s 97ms/step - loss: 3.8580 - acc: 0.1213 - val_loss: 3.3012 - val_acc: 0.1827
Epoch 4/20
113/113 [==============================] - 10s 91ms/step - loss: 3.4190 - acc: 0.1753 - val_loss: 3.0070 - val_acc: 0.2449
Epoch 5/20
```

Result and Evaluation

Training and validation accuracy & Training and validation lost







```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

Conclusion

- As a result of repeated learning 20times, the loss value was confirmed as 1.6036 accuracy 0.5495 value

- The compact internal representation of densenet and reduced feature redundancy make it good feature extractor in computer vision.

References

1. https://www.kaggle.com/ysthehurricane/107-yoga-asanas-classification-using-densenet-121
2. https://blog.naver.com/PostView.naver?blogId=winddori2002&logNo=222050432149&parentCategoryNo=&categoryNo=32&viewDate=&isShowPopularPosts=false&from=postView
3. https://velog.io/@skhim520/DenseNet-논문-리뷰