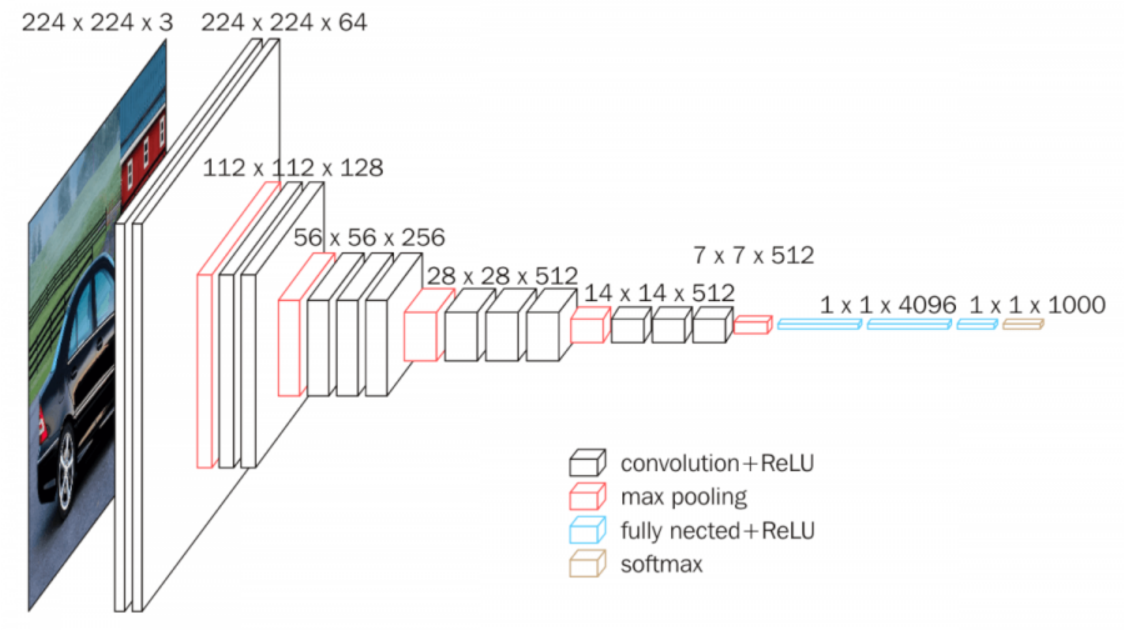
Project5

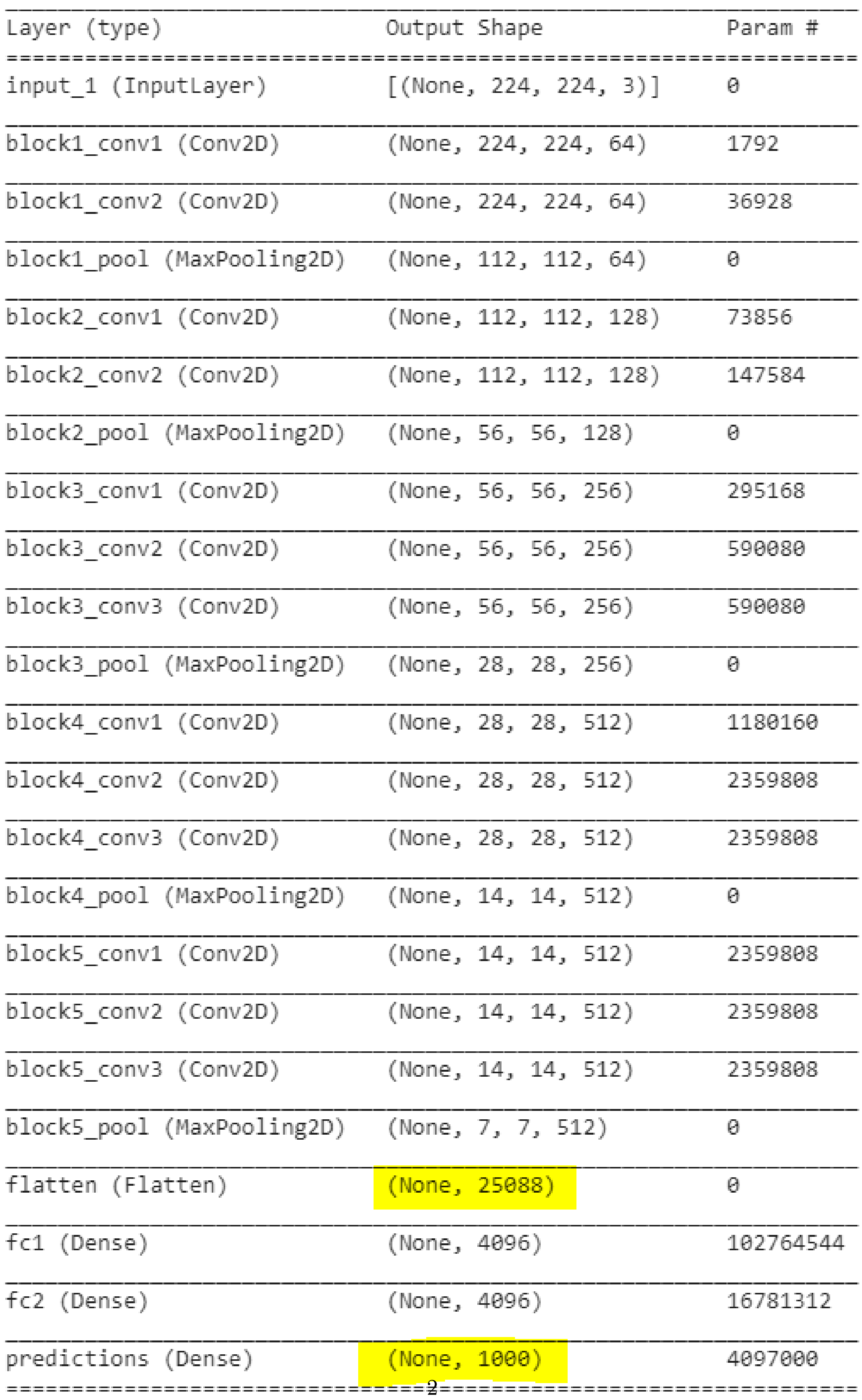
January 23, 2021

1 Brief Review

The structure of the layers of VGG-16 Model which is trained on the ImageNet Dataset is shown below.



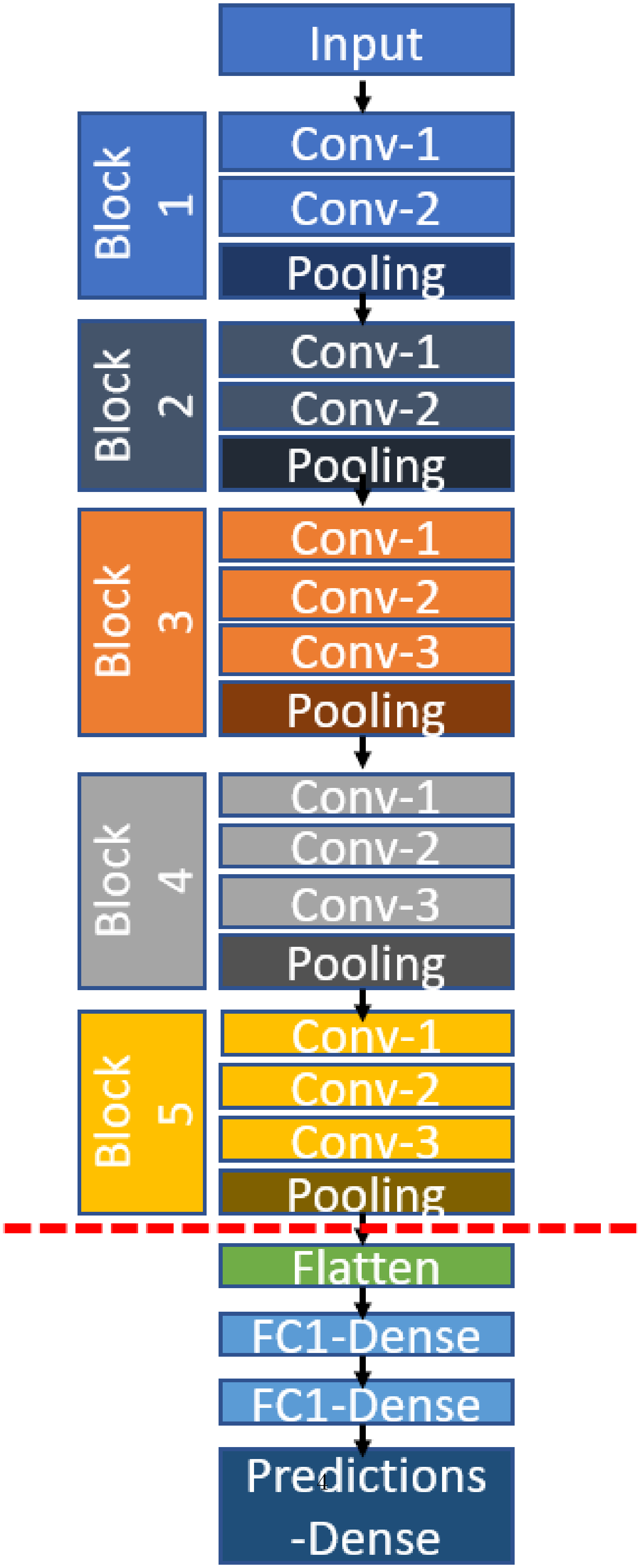
And also you can see another table which describes the summary of VGG-16



```
_________________________________________________________________
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         
_________________________________________________________________
fc1 (Dense)                  (None, 4096)              102764544 
_________________________________________________________________
fc2 (Dense)                  (None, 4096)              16781312  
_________________________________________________________________
predictions (Dense)          (None, 1000)              4097000   
=================================================================
```

The VGG-16 contains 16 layers. As you can see, after flatten of the last convolutional layer, there is 25,088 features (the 1st highlighted) and in the final layer (prediction or final dense layer), there are 1000 nodes (the 2nd highlighted), because VGG-16 is trained for 1000-class classification problems.

For using this Model and applying transfer learning on that for our specific 6-class classification, we can extract 25088 features from the Medical-MNIST dataset using the VGG model. The cut for feature extraction has been shown in the image below.


Input
Block 1
Conv-1
Conv-2
Pooling
Block 2
Conv-1
Conv-2
Pooling
Block 3
Conv-1
Conv-2
Conv-3
Pooling
Block 4
Conv-1
Conv-2
Conv-3
Pooling
Block 5
Conv-1
Conv-2
Conv-3
Pooling
Flatten
FC1-Dense
FC1-Dense
Predictions
-Dense

1.1 VGG-16 model

[2]:

```
from keras.layers import Input, Lambda, Dense, Flatten
from keras.models import Model
from keras.applications.vgg16 import VGG16
from keras.utils import  plot_model

model = VGG16()
model.summary()
```

```
Model: "vgg16"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (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
_________________________________________________________________
fc1 (Dense)                  (None, 4096)              102764544
_________________________________________________________________
fc2 (Dense)                  (None, 4096)              16781312
_________________________________________________________________
predictions (Dense)          (None, 1000)              4097000
=================================================================
Total params: 138,357,544
Trainable params: 138,357,544
Non-trainable params: 0
_________________________________________________________________
```

[4]:

```
from keras import models

base_model = VGG16(weights='imagenet')
model_VGG16 = models.Model(inputs=base_model.input, outputs=base_model.
 ↪get_layer('flatten').output)
model_VGG16.summary()
```

```
Model: "model_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_4 (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
=================================================================
Total params: 14,714,688
Trainable params: 14,714,688
Non-trainable params: 0
-----------------------------------------------------------------
```

2 Importing Medical-MNIST dataset

Now we will import the images from Medical-MNIST dataset and convert them to a normalized value for each pixel and a number between 0 to 5 for each class.

[5]:

```python
import numpy as np
import pandas as pd
import os
import torch
from PIL import Image


torch.manual_seed(1)
classes=['Abdomen CT',"Breast MRI",'Chest XRay','Chest CT','Hand','Head CT']

AbdomenCT_dir='./medical-mnist/AbdomenCT/'
BreastMRI_dir='./medical-mnist/BreastMRI/'
ChestXRay_dir='./medical-mnist/CXR/'
ChestCT_dir='./medical-mnist/ChestCT/'
Hand_dir='./medical-mnist/Hand/'
HeadCT_dir='./medical-mnist/HeadCT/'
```

```python
directories=[AbdomenCT_dir, BreastMRI_dir, ChestXRay_dir, ChestCT_dir, Hand_dir, HeadCT_dir]


def create_dataset(directories, height=64,width=64):
    X=[]
    Y=[]
    data_index = 0
    for class_num in range(len(directories)):
        print(class_num)
        images=os.listdir(directories[class_num])
        print(images[0])
        print(len(images))
        print(directories[class_num]+images[0])
        for i in range(len(images)):
            images[i] = directories[class_num]+images[i]
            image_x = Image.open(images[i])
            image_x = image_x.resize((width,height))
#             image_x = image_x.convert('L')
            image_x = np.asarray(image_x)
            image_x = image_x.astype('float32')
            image_x /= 255
            image_y = class_num
            X.append(image_x)
            Y.append(image_y)
    return X, Y

X, y = create_dataset(directories=directories)
```

```
0
001498.jpeg
10000
./medical-mnist/AbdomenCT/001498.jpeg
1
001498.jpeg
8954
./medical-mnist/BreastMRI/001498.jpeg
2
001498.jpeg
10000
./medical-mnist/CXR/001498.jpeg
3
001498.jpeg
10000
./medical-mnist/ChestCT/001498.jpeg
4
001498.jpeg
```

```
10000
./medical-mnist/Hand/001498.jpeg
5
001498.jpeg
10000
./medical-mnist/HeadCT/001498.jpeg
```

3 Adding our prefered flatten layer to the VGG model

[7]:

```python
from keras import optimizers
for layer in model.layers:
    layer.trainable = False

prediction = Dense(6, activation='softmax')(model_VGG16.output)
model = Model(inputs=model_VGG16.input, outputs=prediction)
model.summary()
```

```
Model: "model_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_4 (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_2 (Dense)              (None, 6)                 150534
=================================================================
Total params: 14,865,222
Trainable params: 150,534
Non-trainable params: 14,714,688
_________________________________________________________________
```

[ ]:

```
from keras import optimizers

model.compile(loss='categorical_crossentropy',
              optimizer=optimizers.Adam(),
              metrics=['accuracy'])

X = np.array((X))
y = np.array((y))

model.fit(X, y, batch_size=32, epochs=20, validation_split=0.3)
```