Open in Colab

Rice Leaf Disease Image Classification Using Deep Learning

Author : Nurgul Kurbanali kyzy

Business Case

Rice is one of the most cultivated crops in the world, in over a hundred countries. Rice production is the 3rd largest among cereals in the U.S. Arkansas where our stakeholders from ranks 1st in rice production in the U.S., accounting for over 40% of rice production.With rice being such a valuable commodity, a common problem that widespread cultivators face is infestation of rice leaf diseases. Infection by diseases results in great loss to economic to the farmers in every year.

In traditional practices, identification is performed either by visual observation or by testing in laboratory. The visual observation requires expertise and it may vary subject to an individual which may lead to an error while the laboratory test is time consuming and may not be able to provide the results in time. Taking into consideration the large range of possible diseases, it becomes a difficult task for cultivators to individually identify these cases, especially for large fields.

In an effort to solve this problem and assist farmers and everyday gardeners to ensure a healthy crop, *Leaf Green* uses an image classification model to help predict and identify three common leaf diseases *(Brown Spot, Leaf Blast)* along with *Healthy* plants based on 3355 rice images.

💻 Project Notebook was run in **Google Colab**

Import Libraries

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from keras.preprocessing import image
%matplotlib inline
import seaborn as sns
sns.set_theme(style="whitegrid")
from matplotlib.transforms import offset_copy
from keras.preprocessing.image import load_img
```

```
from keras.preprocessing.image import load_img
from keras.preprocessing.image import img_to_array

from tensorflow import keras
import tensorflow as tf
from keras import layers
from keras.models import Sequential
from keras.layers import Dense, Flatten, Dropout, Flatten,Conv2D,MaxPooling2D
from keras.layers import   Activation, BatchNormalization
from sklearn.metrics import classification_report, confusion_matrix
from keras.preprocessing.image import ImageDataGenerator
import itertools
import zipfile
from PIL import Image
import warnings
warnings.filterwarnings('ignore') #ignores warnings
import os
from pathlib import Path
# Libraris for uploding dataset from the Kaggle
! pip install kaggle
! mkdir ~/.kaggle
! cp kaggle.json ~/.kaggle/
! chmod 600 ~/.kaggle/kaggle.json

# Libraries  for splitting the img folder into(train,val,test)
!pip install split-folders
import splitfolders

# Remove unnecessary folder in colab
import shutil
shutil.rmtree('/content/sample_data')
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: kaggle in /usr/local/lib/python3.7/dist-packages (1.5.12)
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages (from kaggle) (4.64.1)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.7/dist-packages (from kaggle) (6.1.2)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.7/dist-packages (from kaggle) (1.24.3)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.7/dist-packages (from kaggle) (2.8.2)
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (from kaggle) (2.23.0)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.7/dist-packages (from kaggle) (1.15.0)
Requirement already satisfied: certifi in /usr/local/lib/python3.7/dist-packages (from kaggle) (2022.6.15)
Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.7/dist-packages (from python-slugify
->kaggle) (1.3)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests->kaggl
e) (3.0.4)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests->kaggle) (2.
```

```
10)
cp: cannot stat 'kaggle.json': No such file or directory
chmod: cannot access '/root/.kaggle/kaggle.json': No such file or directory
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting split-folders
  Downloading split_folders-0.5.1-py3-none-any.whl (8.4 kB)
Installing collected packages: split-folders
Successfully installed split-folders-0.5.1
```

Load Dataset and Split into Subsets

In [ ]:

```
# Upload dataset from Kaggle
!kaggle datasets download -d nizorogbezuode/rice-leaf-images
```

```
Downloading rice-leaf-images.zip to /content
 96% 17.0M/17.7M [00:01<00:00, 12.3MB/s]
100% 17.7M/17.7M [00:01<00:00, 13.4MB/s]
```

In [ ]:

```
# Unzip file
zip_ref = zipfile.ZipFile("/content/rice-leaf-images.zip", "r")
zip_ref.extractall()
zip_ref.close()
```

In [ ]:

```
# Walk through pizza_steak directory and list number of files
for dir_path, dir_names, file_names in os.walk("rice_images"):
    print(f"There are {len(dir_names)} directories and {len(file_names)} images in '{dir_path}'.")
```

```
There are 4 directories and 0 images in 'rice_images'.
There are 0 directories and 779 images in 'rice_images/_LeafBlast'.
There are 0 directories and 1488 images in 'rice_images/_Healthy'.
There are 0 directories and 523 images in 'rice_images/_BrownSpot'.
There are 0 directories and 565 images in 'rice_images/_Hispa'.
```

In [ ]:

```
# Removing underscore of the img folder names
os.rename('/content/rice_images/_BrownSpot', '/content/rice_images/BrownSpot')
os.rename('/content/rice_images/_Healthy', '//content/rice_images/Healthy')
os.rename('/content/rice_images/_Hispa', '/content/rice_images/Hispa')
os.rename('/content/rice_images/_LeafBlast', '/content/rice_images/LeafBlast')
```

In [ ]:

```
# Split dataset into train/validation/test sets
```

```
splitfolders.ratio("/content/rice_images", # The location of full dataset
                   output="dataset", # The output location
                   seed=42, # The number of seed
                   ratio=(.8, .1, .1), # The ratio of splited dataset
                   group_prefix=None,
                   move=False )
```

```
Copying files: 3355 files [00:00, 9283.14 files/s]
```

Data Exploration

The data was sourced from https://www.kaggle.com/datasets/nizorogbezuode/rice-leaf-images dataset . This is a dataset for classification of rice leaf diseases. The images are in 4 folders, classified as follows:

In [ ]:

```
#plot number of classes to identify possible imbalances
num_class = {'Brown Spot':523,
                 'Hispa': 565,
                 'Leaf Blast': 779,
                 'Healthy': 1488}

plt.bar(num_class.keys(), num_class.values(), width = .5, color=['olive', 'olive', 'olive', 'green'])
plt.xlabel('Class Names')
plt.ylabel('Total Img')
plt.title("Number of Total Images per Class");
```

In [ ]:

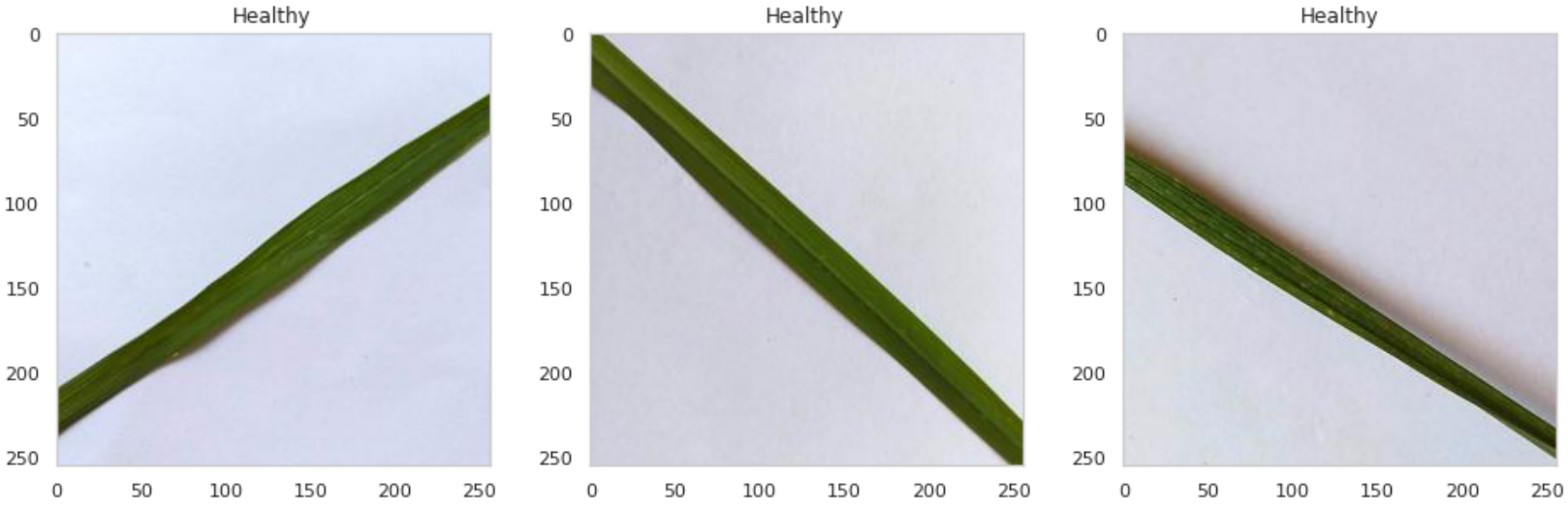
```
# Set path to were dataset is saved
train_dir = '/content/dataset/train/'
val_dir = '/content/dataset/val/'
test_dir = "/content/dataset/test/"

# Set a path within train dataset to all classes
brownspot = [train_dir + '/BrownSpot/' + img for img in os.listdir(train_dir+ '/BrownSpot')[:6]]
healthy = [train_dir  + '/Healthy/' + img for img in os.listdir(train_dir + '/Healthy')[:6]]
hispa = [train_dir  + '/Hispa/' + img for img in os.listdir(train_dir + '/Hispa')[:6]]
leafblast = [train_dir  + '/LeafBlast/' + img for img in os.listdir(train_dir + '/LeafBlast')[:6]]
```

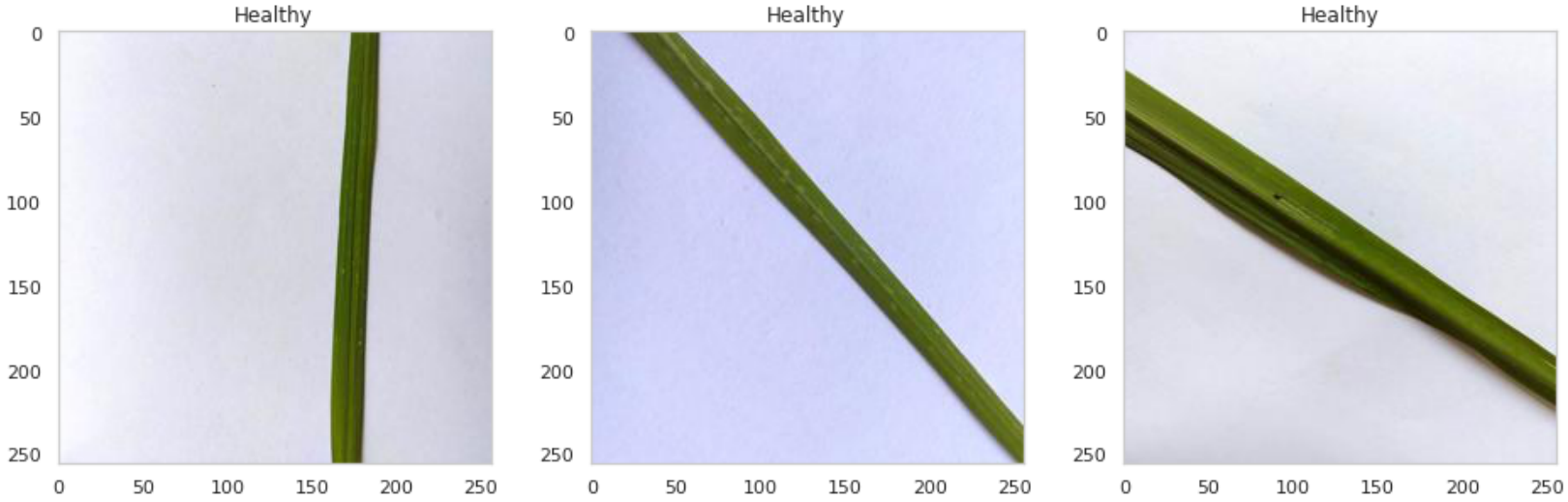
The following disease explanations are from the Rice Knowledge Bank, and more information can be found at http://www.knowledgebank.irri.org/

Natural green leaves indicate **healthy** rice plants.

In [ ]:

```
# Plot samples from healthy class
plt.figure(figsize=(16,16))
for i,k  in enumerate(healthy):
    image = Image.open(k)
    plt.subplot(3,3,i+1)
    plt.imshow(image)
    plt.title("Healthy")
    plt.grid(False);
```

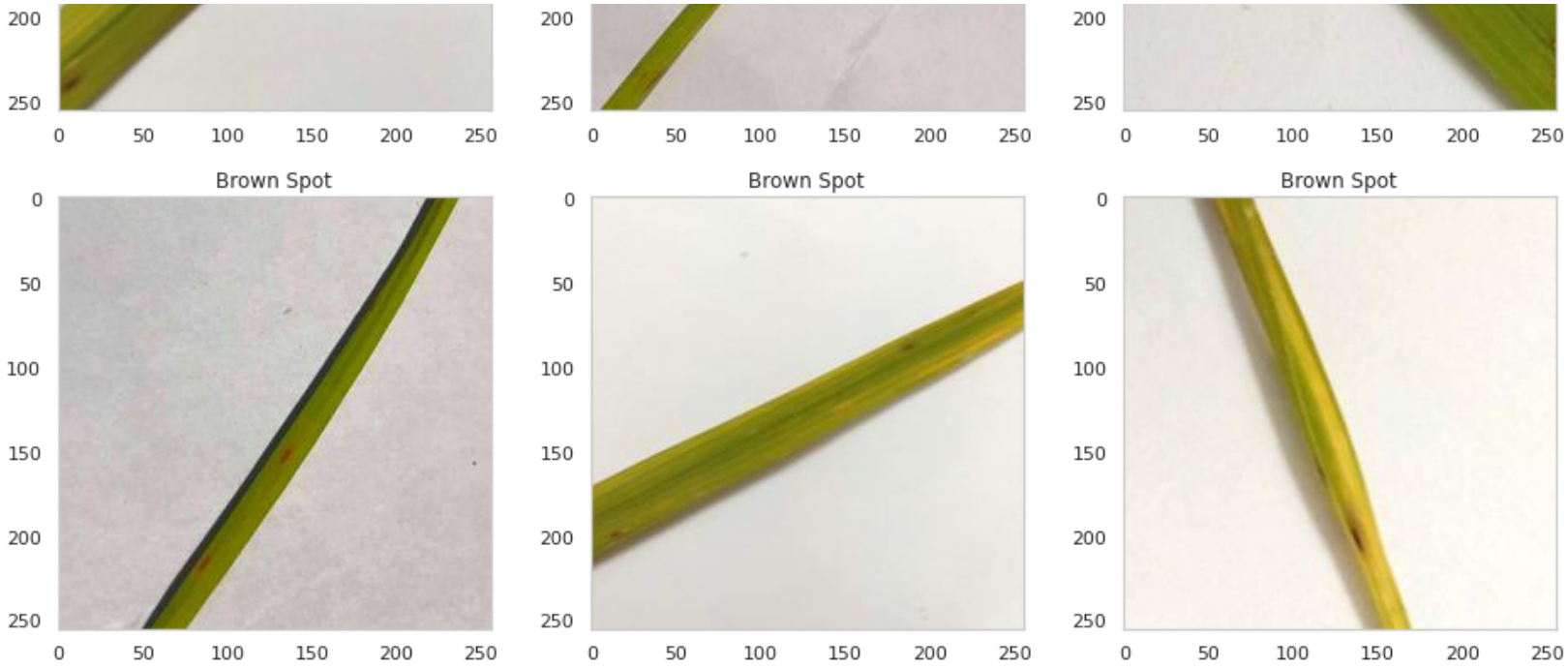




Rice **Brown Spot** is the one of the major fungal diseases in rice in which it is caused by *Bipolarisoryzae*. The fungus can survive in the seed for more than four years and can spread from plant to plant through air. This diseases mainly attacks the crop from seedling stage to milky stage. Brown spot is otherwise called as sesame leaf spot or *Helminthosporiose*. Brown spot causes both quality and quantity losses.

In [ ]:

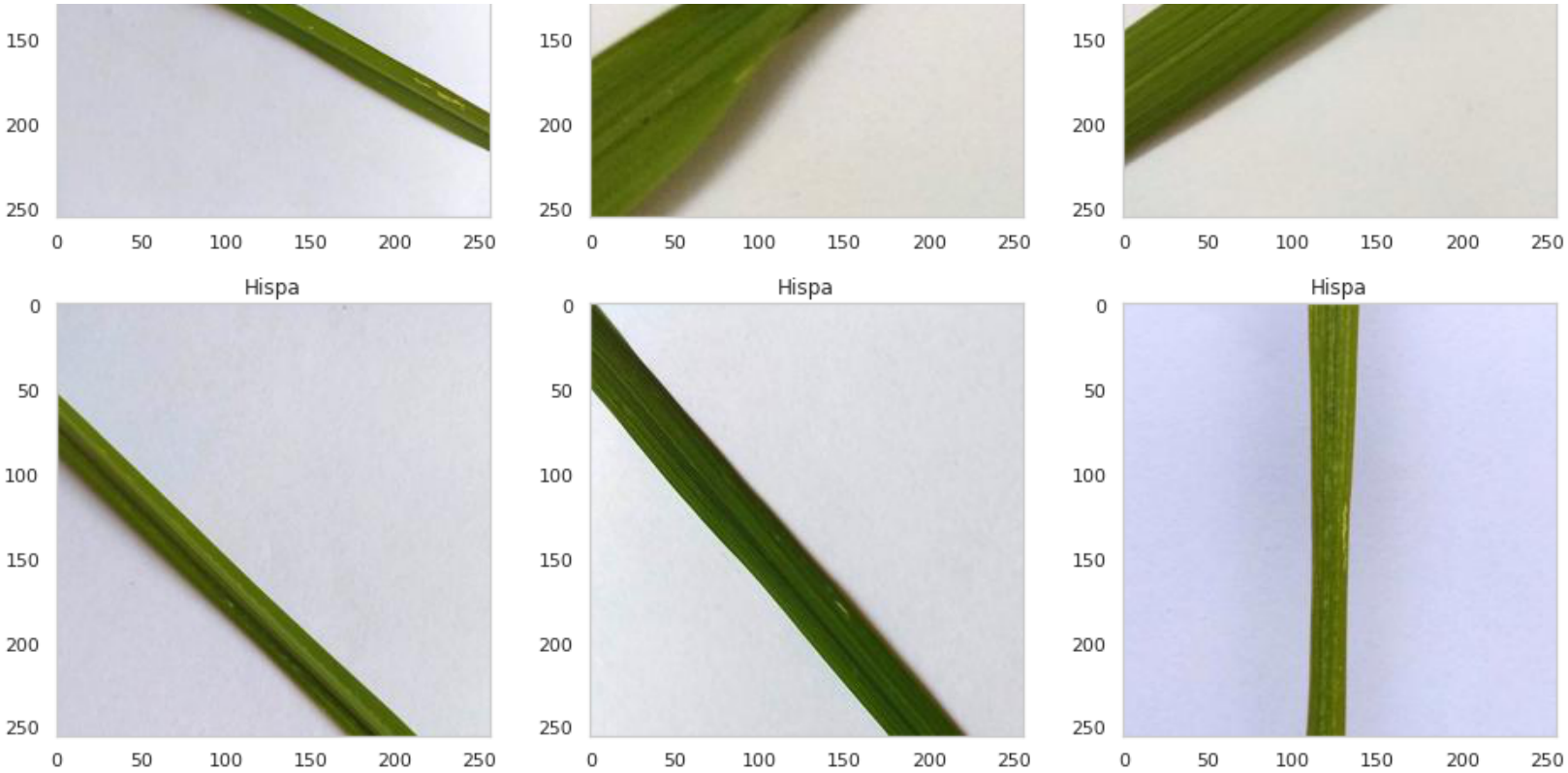
```
# Plot sapmles from brown spot class
plt.figure(figsize=(16,16))
for i,k  in enumerate(brownspot):
    image = Image.open(k)
    plt.subplot(3,3,i+1)
    plt.imshow(image)
    plt.title("Brown Spot")
    plt.grid(False);
```



Rice **Hispa** scrapes the upper surface of leaf blades leaving only the lower epidermis.It also tunnels through the leaf tissues. When damage is severe, plants become less vigorous. The presence of grassy weeds in and near rice fields as alternate hosts harbor and encourage the pest to develop. Heavily fertilized field also encourages the damage.

In [ ]:

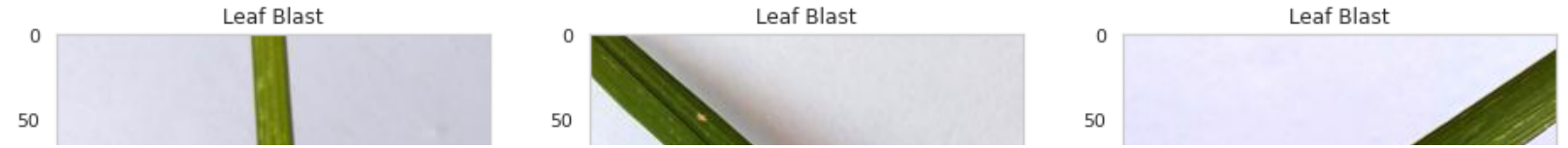
```
# Plot sapmles from hispa class
plt.figure(figsize=(16,16))
for i,k  in enumerate(hispa):
    image = Image.open(k)
    plt.subplot(3,3,i+1)
    plt.imshow(image)
    plt.title("Hispa")
    plt.grid(False);
```

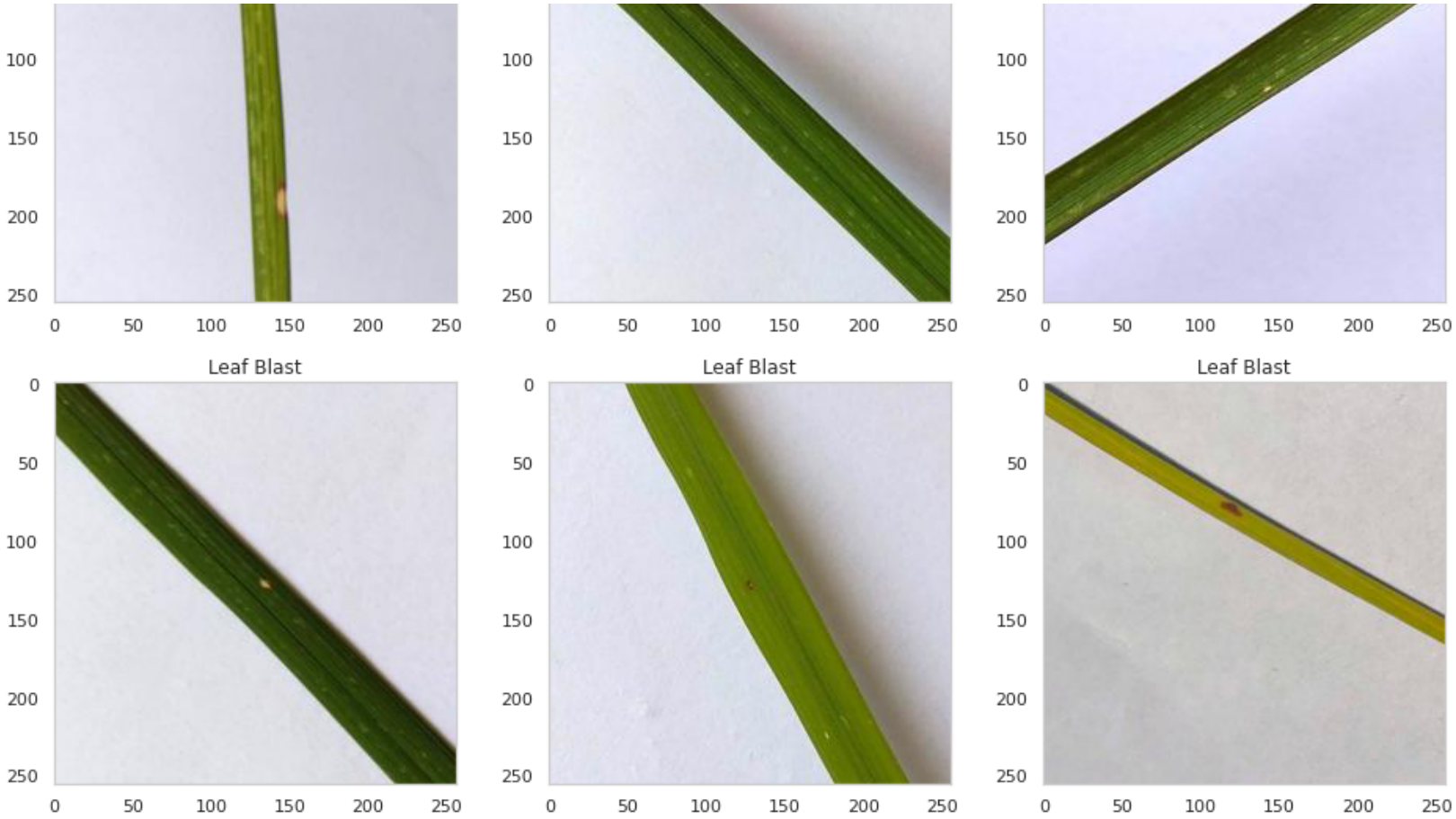


Rice **Blast(leaf and collar)** is caused by the fungus Magnaporthe oryzae. It can affect all above ground parts of a rice plant: leaf, collar, node, neck, parts of panicle, and sometimes leaf sheath. Rice can have blast in all growth stages. However, leaf blast incidence tends to lessen as plants mature and develop adult plant resistance to the disease.

In [ ]:

```
# Plot sapmles from leaf blast class
plt.figure(figsize=(16,16))
for i,k  in enumerate(leafblast):
    image = Image.open(k)
    plt.subplot(3,3,i+1)
    plt.imshow(image)
    plt.title("Leaf Blast")
    plt.grid(False)
```





In [ ]:

```
# Check  size of first images from 2 class
im_1 = Image.open('/content/dataset/train/BrownSpot/shape 1 .jpg')
im_2 = Image.open('/content/dataset/train/Healthy/shape 1 .jpg')
arr_1 = np.array(im_1)
arr_2 =np.array(im_2)
print(arr_1.shape, arr_2.shape)
```

```
(256, 256, 3) (256, 256, 3)
```

These images are in JPEG format and have good resolution with a width and height of 256*256. The image background is also white so there is no need to apply any background subtraction method.

In [ ]:

```
# Create data generators
# Creation of train set
```

```
train_set = ImageDataGenerator(rescale=1./255).flow_from_directory(
                                                train_dir,
                                                batch_size=32,
                                                target_size=(224,224),
                                                class_mode="categorical", # type of problem we're working on
                                                subset = "training",
                                                color_mode='rgb',
                                                seed=127)
# Creation of validation set
val_set = ImageDataGenerator(rescale=1./255).flow_from_directory(
                                                val_dir,
                                                batch_size=32,
                                                target_size=(224,224),
                                                class_mode="categorical", # type of problem we're working on
                                                #subset = "validation",
                                                color_mode='rgb',
                                                seed=127)
# Creation of test set
test_set = ImageDataGenerator(rescale=1./255).flow_from_directory(
                                               test_dir,
                                               batch_size=32,
                                               target_size=(224,224),
                                               class_mode="categorical", # type of problem we're working on
                                               color_mode='rgb',
                                               seed=127,
                                               shuffle=False)
```

```
Found 2683 images belonging to 4 classes.
Found 333 images belonging to 4 classes.
Found 339 images belonging to 4 classes.
```

Model Building

#1. Baseline (dummy) Model

In [ ]:

```
# Set random seed
INPUT_SHAPE = (224, 224, 3) # height, width, colour channels
LOSS_FN =tf.keras.losses.CategoricalCrossentropy()  #loss function for multi-class classification model

# Create a model
base_model = tf.keras.Sequential([
  Flatten(input_shape=INPUT_SHAPE),
  Dense(100, activation='relu'),
```

```
  Dense(50, activation='relu'),
  Dense(4, activation='softmax') # multi class activation output
])

# Compile the model
base_model.compile(loss=LOSS_FN,
              optimizer=tf.keras.optimizers.Adam(),
              metrics=["accuracy"])

# Model summary
base_model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten (Flatten)           (None, 150528)            0

 dense (Dense)               (None, 100)               15052900

 dense_1 (Dense)             (None, 50)                5050

 dense_2 (Dense)             (None, 4)                 204

=================================================================
Total params: 15,058,154
Trainable params: 15,058,154
Non-trainable params: 0
_________________________________________________________________
```

In [ ]:

```
# Fit the baseline model with 15 epochs
history_baseline= base_model.fit(train_set,
                        epochs=15,
                        validation_data=val_set)
```

```
Epoch 1/15
21/21 [==============================] - 7s 293ms/step - loss: 37.8423 - accuracy: 0.3075 - val_loss: 35.5169 - v
al_accuracy: 0.1682
Epoch 2/15
21/21 [==============================] - 6s 279ms/step - loss: 19.9240 - accuracy: 0.2903 - val_loss: 9.7449 - va
l_accuracy: 0.3994
Epoch 3/15
21/21 [==============================] - 6s 276ms/step - loss: 7.0607 - accuracy: 0.3272 - val_loss: 3.0344 - val
_accuracy: 0.2342
Epoch 4/15
```

```
21/21 [==============================] - 6s 276ms/step - loss: 3.5681 - accuracy: 0.3347 - val_loss: 5.0210 - val
_accuracy: 0.2312
Epoch 5/15
21/21 [==============================] - 6s 271ms/step - loss: 5.5305 - accuracy: 0.3235 - val_loss: 5.7286 - val
_accuracy: 0.4444
Epoch 6/15
21/21 [==============================] - 7s 323ms/step - loss: 8.2587 - accuracy: 0.3235 - val_loss: 4.8013 - val
_accuracy: 0.1832
Epoch 7/15
21/21 [==============================] - 6s 272ms/step - loss: 4.8803 - accuracy: 0.3358 - val_loss: 2.1892 - val
_accuracy: 0.2372
Epoch 8/15
21/21 [==============================] - 6s 270ms/step - loss: 3.4714 - accuracy: 0.3600 - val_loss: 1.7269 - val
_accuracy: 0.4625
Epoch 9/15
21/21 [==============================] - 6s 270ms/step - loss: 2.0522 - accuracy: 0.4014 - val_loss: 1.4766 - val
_accuracy: 0.4084
Epoch 10/15
21/21 [==============================] - 6s 269ms/step - loss: 2.5868 - accuracy: 0.3761 - val_loss: 1.8433 - val
_accuracy: 0.2823
Epoch 11/15
21/21 [==============================] - 6s 267ms/step - loss: 4.1623 - accuracy: 0.3630 - val_loss: 4.4979 - val
_accuracy: 0.2733
Epoch 12/15
21/21 [==============================] - 6s 265ms/step - loss: 3.6174 - accuracy: 0.3656 - val_loss: 3.4872 - val
_accuracy: 0.2102
Epoch 13/15
21/21 [==============================] - 6s 268ms/step - loss: 5.8256 - accuracy: 0.3407 - val_loss: 3.7997 - val
_accuracy: 0.2222
Epoch 14/15
21/21 [==============================] - 6s 273ms/step - loss: 4.7643 - accuracy: 0.3250 - val_loss: 1.7192 - val
_accuracy: 0.4685
Epoch 15/15
21/21 [==============================] - 6s 270ms/step - loss: 2.8922 - accuracy: 0.4074 - val_loss: 2.7144 - val
_accuracy: 0.3483
```

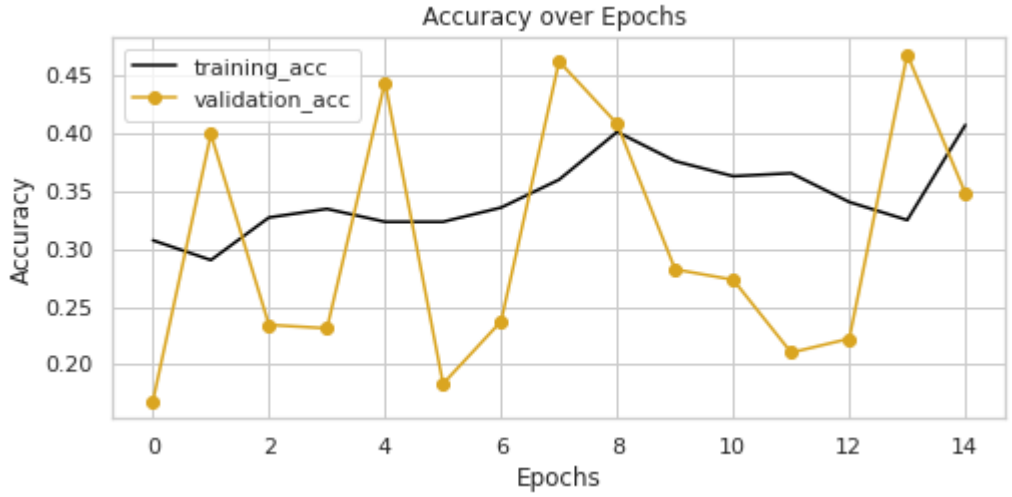
In [ ]:

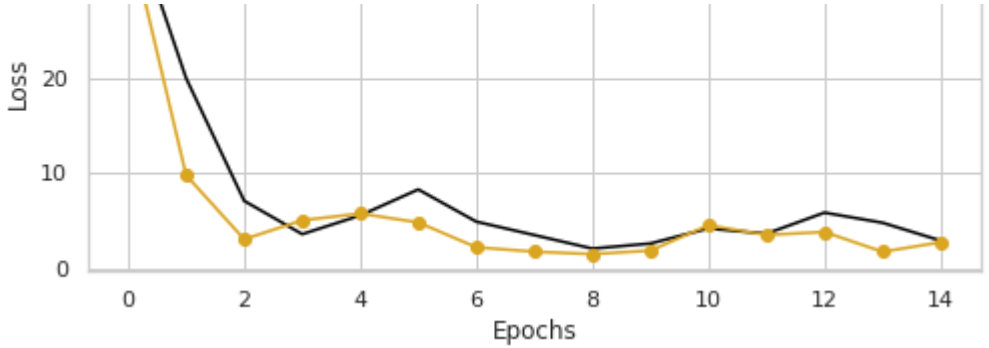
```python
# Plot the training and validation loss curves separately
def plot_learning_curves(history):
  """
  Returns loss curves for training and validation metrics separately.
  """
  epochs = range(len(history.history["loss"]))
  accuracy = history.history["accuracy"]
  val_accuracy = history.history['val_accuracy']
```

```
# Plot accuracy curve
plt.figure(figsize=(8,3.5))
plt.plot(epochs, accuracy, label="training_acc", color="#000000" )
plt.plot(epochs, val_accuracy, label="validation_acc", color="goldenrod", marker='o')
plt.title("Accuracy over Epochs ")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend();

loss = history.history["loss"]
val_loss = history.history["val_loss"]

# Plot loss curve
plt.figure(figsize=(8,3.5))
plt.plot(epochs, loss, label="training_loss",  color="#000000")
plt.plot(epochs, val_loss, label="validation_loss",color="goldenrod", marker='o')
plt.title("Loss over Epochs")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend();
```

In [ ]:

```
plot_learning_curves(history_baseline)
```





Based on the learning curves of baseline model we can see a highly fluctuation of validation accuracy. But validation loss going down - this makes sense. After vanilla model we will try with CNN models

CNN models have shown better performance in various applications including an agriculture. It consists of three layers, wherein the first layer is named as convolution layer, the second layer as pooling layer, and the last layer is so-called fully connected layer. The convolution and pooling layers are responsible for learning of the model and a fully connected layer does the classification.

#2. Model with Convolutional Neural Network (CNN3)

In [ ]:

```
# Build first CNN model  with 3 convolutional layers
model_cnn1 = tf.keras.Sequential([
    Conv2D(filters=64,
           kernel_size=(3,3),
           padding='same', # Add a zero padding to the feature map
           activation='relu',
           input_shape=INPUT_SHAPE),
    MaxPooling2D(pool_size=(2,2)), # Added MaxPool
    Conv2D(filters=128,
           kernel_size=(3,3),
           padding='same',
           activation='relu'),
    MaxPooling2D(pool_size=(2,2)),
    Conv2D(filters=32,
           kernel_size=(3,3),
           activation='relu'),
    MaxPooling2D(pool_size=(2,2)),
    Dropout(0.20),
    Flatten(),
    Dropout(0.20),
```

```
    Dense(4, activation='softmax') ])

# Compile the model
model_cnn1.compile(loss=LOSS_FN,
              optimizer=tf.keras.optimizers.Adam(learning_rate=3e-4),
              metrics=["accuracy"])

# Fit the first CNN model
history_cnn1 = model_cnn1.fit(train_set,
                        epochs=30,
                        steps_per_epoch= len(train_set),
                        validation_data=val_set)
```

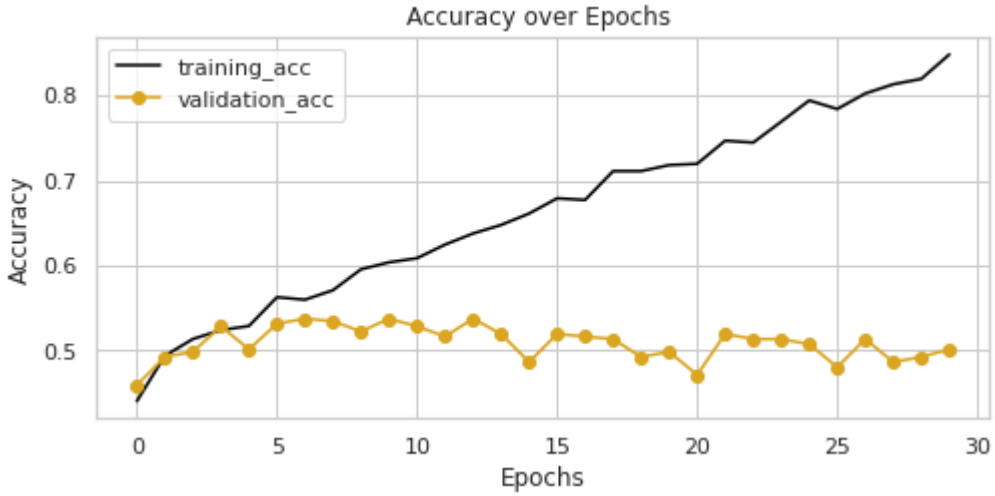
```
Epoch 1/30
84/84 [==============================] - 18s 80ms/step - loss: 1.2880 - accuracy: 0.4406 - val_loss: 1.2096 - val
_accuracy: 0.4595
Epoch 2/30
84/84 [==============================] - 7s 85ms/step - loss: 1.1779 - accuracy: 0.4939 - val_loss: 1.1602 - val_
accuracy: 0.4925
Epoch 3/30
84/84 [==============================] - 5s 63ms/step - loss: 1.1162 - accuracy: 0.5136 - val_loss: 1.1579 - val_
accuracy: 0.4985
Epoch 4/30
84/84 [==============================] - 5s 63ms/step - loss: 1.0860 - accuracy: 0.5240 - val_loss: 1.1217 - val_
accuracy: 0.5285
Epoch 5/30
84/84 [==============================] - 5s 64ms/step - loss: 1.0773 - accuracy: 0.5289 - val_loss: 1.1196 - val_
accuracy: 0.5015
Epoch 6/30
84/84 [==============================] - 5s 64ms/step - loss: 1.0283 - accuracy: 0.5628 - val_loss: 1.1134 - val_
accuracy: 0.5315
Epoch 7/30
84/84 [==============================] - 5s 62ms/step - loss: 1.0091 - accuracy: 0.5598 - val_loss: 1.0925 - val_
accuracy: 0.5375
Epoch 8/30
84/84 [==============================] - 5s 63ms/step - loss: 0.9977 - accuracy: 0.5710 - val_loss: 1.0833 - val_
accuracy: 0.5345
Epoch 9/30
84/84 [==============================] - 5s 61ms/step - loss: 0.9451 - accuracy: 0.5956 - val_loss: 1.1058 - val_
accuracy: 0.5225
Epoch 10/30
84/84 [==============================] - 5s 61ms/step - loss: 0.9219 - accuracy: 0.6038 - val_loss: 1.0829 - val_
accuracy: 0.5375
Epoch 11/30
84/84 [==============================] - 5s 61ms/step - loss: 0.9157 - accuracy: 0.6086 - val_loss: 1.1095 - val_
```

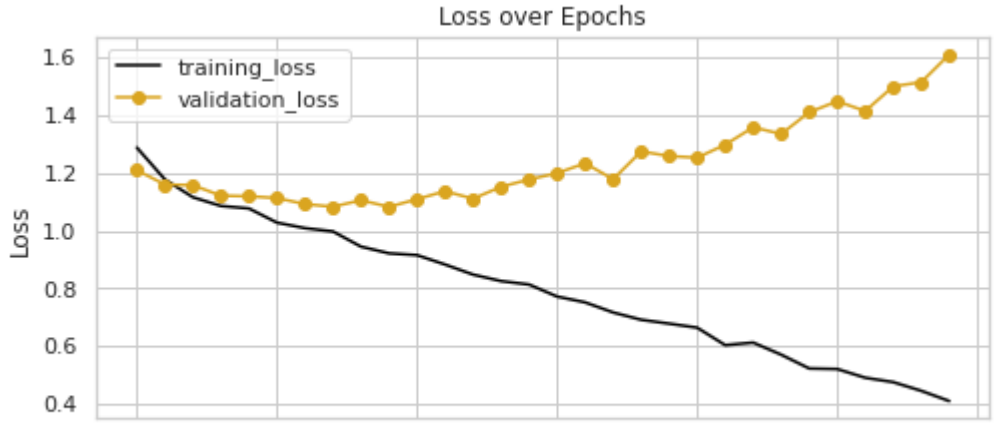
```
accuracy: 0.5285
Epoch 12/30
84/84 [==============================] - 5s 61ms/step - loss: 0.8833 - accuracy: 0.6247 - val_loss: 1.1371 - val_
accuracy: 0.5165
Epoch 13/30
84/84 [==============================] - 5s 62ms/step - loss: 0.8484 - accuracy: 0.6377 - val_loss: 1.1112 - val_
accuracy: 0.5375
Epoch 14/30
84/84 [==============================] - 5s 63ms/step - loss: 0.8258 - accuracy: 0.6478 - val_loss: 1.1524 - val_
accuracy: 0.5195
Epoch 15/30
84/84 [==============================] - 5s 61ms/step - loss: 0.8140 - accuracy: 0.6612 - val_loss: 1.1777 - val_
accuracy: 0.4865
Epoch 16/30
84/84 [==============================] - 5s 62ms/step - loss: 0.7723 - accuracy: 0.6791 - val_loss: 1.1997 - val_
accuracy: 0.5195
Epoch 17/30
84/84 [==============================] - 5s 62ms/step - loss: 0.7522 - accuracy: 0.6772 - val_loss: 1.2338 - val_
accuracy: 0.5165
Epoch 18/30
84/84 [==============================] - 5s 63ms/step - loss: 0.7172 - accuracy: 0.7111 - val_loss: 1.1786 - val_
accuracy: 0.5135
Epoch 19/30
84/84 [==============================] - 5s 62ms/step - loss: 0.6919 - accuracy: 0.7111 - val_loss: 1.2749 - val_
accuracy: 0.4925
Epoch 20/30
84/84 [==============================] - 5s 64ms/step - loss: 0.6785 - accuracy: 0.7182 - val_loss: 1.2587 - val_
accuracy: 0.4985
Epoch 21/30
84/84 [==============================] - 5s 63ms/step - loss: 0.6645 - accuracy: 0.7197 - val_loss: 1.2531 - val_
accuracy: 0.4715
Epoch 22/30
84/84 [==============================] - 5s 64ms/step - loss: 0.6041 - accuracy: 0.7469 - val_loss: 1.2982 - val_
accuracy: 0.5195
Epoch 23/30
84/84 [==============================] - 5s 62ms/step - loss: 0.6128 - accuracy: 0.7447 - val_loss: 1.3584 - val_
accuracy: 0.5135
Epoch 24/30
84/84 [==============================] - 5s 62ms/step - loss: 0.5710 - accuracy: 0.7693 - val_loss: 1.3352 - val_
accuracy: 0.5135
Epoch 25/30
84/84 [==============================] - 5s 62ms/step - loss: 0.5232 - accuracy: 0.7943 - val_loss: 1.4105 - val_
accuracy: 0.5075
Epoch 26/30
84/84 [==============================] - 5s 61ms/step - loss: 0.5213 - accuracy: 0.7842 - val_loss: 1.4486 - val_
accuracy: 0.4805
```

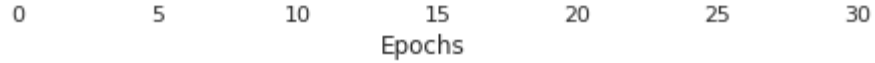
```
Epoch 27/30
84/84 [==============================] - 5s 63ms/step - loss: 0.4911 - accuracy: 0.8025 - val_loss: 1.4146 - val_
accuracy: 0.5135
Epoch 28/30
84/84 [==============================] - 5s 64ms/step - loss: 0.4759 - accuracy: 0.8133 - val_loss: 1.5000 - val_
accuracy: 0.4865
Epoch 29/30
84/84 [==============================] - 5s 63ms/step - loss: 0.4459 - accuracy: 0.8196 - val_loss: 1.5146 - val_
accuracy: 0.4925
Epoch 30/30
84/84 [==============================] - 5s 62ms/step - loss: 0.4103 - accuracy: 0.8483 - val_loss: 1.6104 - val_
accuracy: 0.5015
```

In [ ]:

```
plot_learning_curves(history_cnn1)
```







Despite of having a dropout layers in our first CNN model it is still suffering from overfitting. Thus we will try to combat overfitting by building with r and two conv layer and increasing the value of the dropout layes. In addition the optimizer will be changed to RMSprop from Adam that we used in our first CNN model.

#2. Model with Convolutional Neural Network (CNN2) and RMSprop

In [ ]:

```
# Build CNN2 with bigger dropout and RMSprop optimizer.
model_cnn2 = tf.keras.models.Sequential([
     Conv2D(filters=100,
           kernel_size=(3,3),
           padding='same',
           activation='relu',
           input_shape=(INPUT_SHAPE)),
    MaxPooling2D(pool_size=(2,2)),
    Dropout(0.3),
    Conv2D(filters=250,
           kernel_size=(3,3),
           padding='same',
           activation='relu'),
    MaxPooling2D(pool_size=(2,2)),
    Dropout(0.5),
    Flatten(),
    Dense(4, activation='softmax')
])

# Compile the model
model_cnn2.compile(loss=LOSS_FN,
              optimizer=tf.keras.optimizers.RMSprop(learning_rate=0.001),
              metrics=["accuracy"])

# Fit the  model
history_cnn2 = model_cnn2.fit(train_set,
                         epochs=20,
                        validation_data=val_set )
```

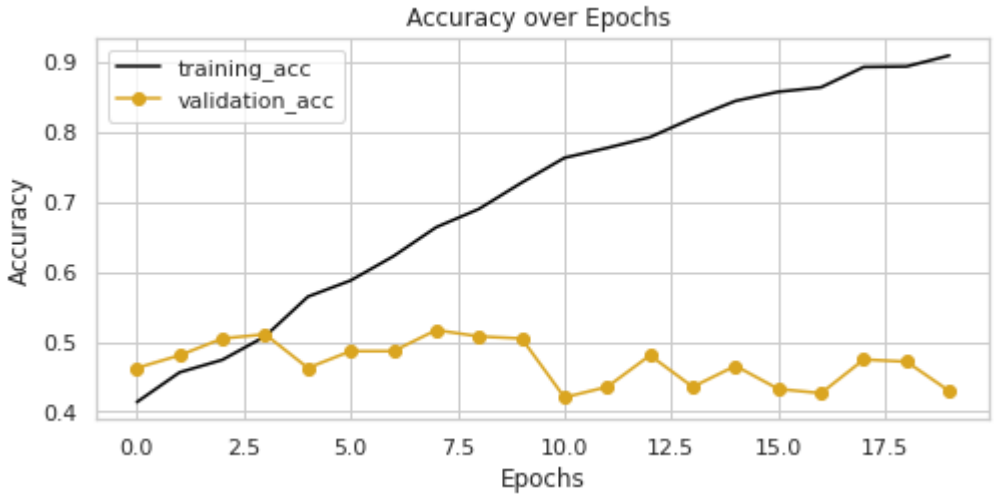
```
Epoch 1/20
84/84 [==============================] - 12s 120ms/step - loss: 5.5095 - accuracy: 0.4137 - val_loss: 1.2314 - va
l_accuracy: 0.4625
Epoch 2/20
84/84 [==============================] - 9s 109ms/step - loss: 1.3641 - accuracy: 0.4562 - val_loss: 1.2429 - val
```

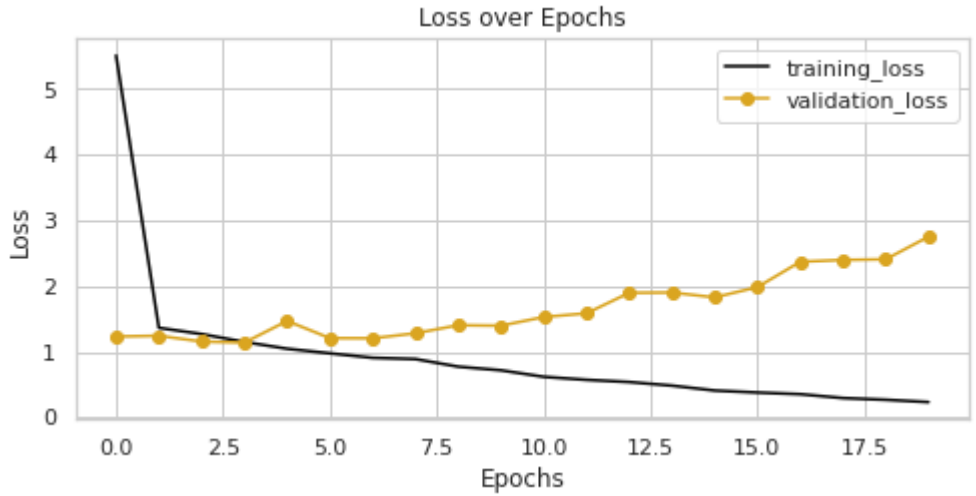
```
84/84 [==============================] - 9s 109ms/step - loss: 1.3041 - accuracy: 0.4502 - val_loss: 1.2429 - val
_accuracy: 0.4805
Epoch 3/20
84/84 [==============================] - 9s 109ms/step - loss: 1.2693 - accuracy: 0.4741 - val_loss: 1.1550 - val
_accuracy: 0.5045
Epoch 4/20
84/84 [==============================] - 9s 112ms/step - loss: 1.1490 - accuracy: 0.5076 - val_loss: 1.1389 - val
_accuracy: 0.5105
Epoch 5/20
84/84 [==============================] - 9s 111ms/step - loss: 1.0456 - accuracy: 0.5643 - val_loss: 1.4689 - val
_accuracy: 0.4625
Epoch 6/20
84/84 [==============================] - 9s 109ms/step - loss: 0.9759 - accuracy: 0.5874 - val_loss: 1.2067 - val
_accuracy: 0.4865
Epoch 7/20
84/84 [==============================] - 9s 109ms/step - loss: 0.9074 - accuracy: 0.6221 - val_loss: 1.2048 - val
_accuracy: 0.4865
Epoch 8/20
84/84 [==============================] - 9s 111ms/step - loss: 0.8887 - accuracy: 0.6634 - val_loss: 1.2808 - val
_accuracy: 0.5165
Epoch 9/20
84/84 [==============================] - 9s 109ms/step - loss: 0.7729 - accuracy: 0.6895 - val_loss: 1.4037 - val
_accuracy: 0.5075
Epoch 10/20
84/84 [==============================] - 9s 110ms/step - loss: 0.7177 - accuracy: 0.7272 - val_loss: 1.3964 - val
_accuracy: 0.5045
Epoch 11/20
84/84 [==============================] - 9s 110ms/step - loss: 0.6183 - accuracy: 0.7626 - val_loss: 1.5285 - val
_accuracy: 0.4204
Epoch 12/20
84/84 [==============================] - 9s 109ms/step - loss: 0.5725 - accuracy: 0.7767 - val_loss: 1.5846 - val
_accuracy: 0.4354
Epoch 13/20
84/84 [==============================] - 9s 109ms/step - loss: 0.5390 - accuracy: 0.7920 - val_loss: 1.8970 - val
_accuracy: 0.4805
Epoch 14/20
84/84 [==============================] - 9s 109ms/step - loss: 0.4841 - accuracy: 0.8192 - val_loss: 1.9011 - val
_accuracy: 0.4354
Epoch 15/20
84/84 [==============================] - 9s 110ms/step - loss: 0.4093 - accuracy: 0.8438 - val_loss: 1.8274 - val
_accuracy: 0.4655
Epoch 16/20
84/84 [==============================] - 9s 109ms/step - loss: 0.3789 - accuracy: 0.8569 - val_loss: 1.9841 - val
_accuracy: 0.4324
Epoch 17/20
84/84 [==============================] - 9s 109ms/step - loss: 0.3539 - accuracy: 0.8632 - val_loss: 2.3705 - val
```

```
_accuracy: 0.4264
Epoch 18/20
84/84 [==============================] - 9s 109ms/step - loss: 0.2946 - accuracy: 0.8923 - val_loss: 2.3972 - val
_accuracy: 0.4745
Epoch 19/20
84/84 [==============================] - 9s 111ms/step - loss: 0.2689 - accuracy: 0.8930 - val_loss: 2.4076 - val
_accuracy: 0.4715
Epoch 20/20
84/84 [==============================] - 9s 109ms/step - loss: 0.2337 - accuracy: 0.9087 - val_loss: 2.7522 - val
_accuracy: 0.4294
```

In [ ]:

```
plot_learning_curves(history_cnn2)
```





Our second CNN model still suffering from the overfitting and validation score did not get improved. Mostly researchers found good results in implementing Batch Normalization after the activation layer. By adding Batch Normalization we reduce the internal covariate shift and instability in distributions of layer activations in deeper networks can reduce the effect of overfitting and works well with generalization data. So we will use Batch Normalization as a *Regularization* technique in our third CNN model.

#3. Model with Convolutional Neural Network (CNN3) and BatchNormalization

In [ ]:

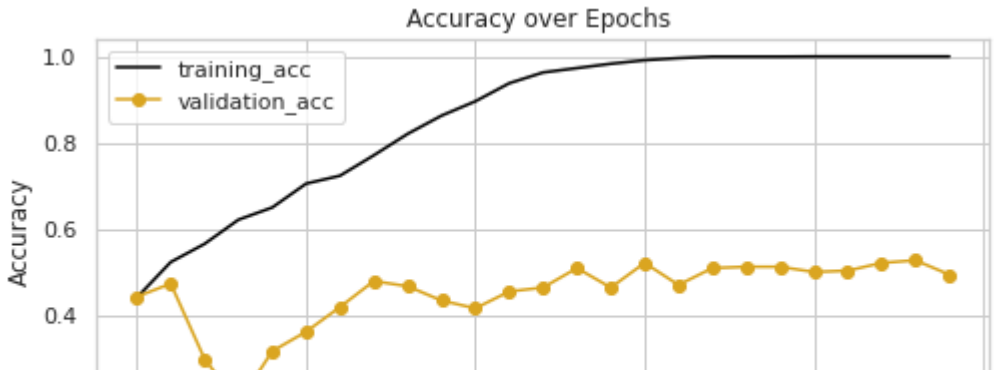
```
# Build CNN3 along with BatchNormalization layer
model_cnn3 = Sequential([
    Conv2D(filters=64,
           kernel_size=(3,3),
           padding='same',
           activation='relu',
           input_shape=INPUT_SHAPE),
    BatchNormalization(),
    MaxPooling2D(pool_size=(2,2)),
    Conv2D(filters=64,
           kernel_size=(3,3),
           padding='same',
           activation='relu'),
    BatchNormalization(),
    MaxPooling2D(pool_size=(2,2)),
    Conv2D(filters=32,
           kernel_size=(3,3),
           activation='relu'),
    BatchNormalization(),
    MaxPooling2D(pool_size=(2,2)),
    Flatten(),
    Dense(4, activation='softmax')
])

# Compile the model
model_cnn3.compile(loss=LOSS_FN,
              optimizer=tf.keras.optimizers.Adam(learning_rate=3e-4),
              metrics=["accuracy"])

# Fit the first CNN model
history_cnn3 = model_cnn3.fit(train_set,
                        epochs=25,
                        validation_data=val_set)
```

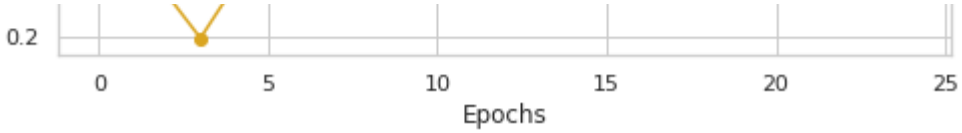
```
Epoch 1/25
84/84 [==============================] - 6s 67ms/step - loss: 1.7579 - accuracy: 0.4424 - val_loss: 1.3370 - val_
accuracy: 0.4444
Epoch 2/25
84/84 [==============================] - 5s 61ms/step - loss: 1.2554 - accuracy: 0.5244 - val_loss: 1.2887 - val_
accuracy: 0.4745
Epoch 3/25
84/84 [==============================] - 5s 63ms/step - loss: 1.1525 - accuracy: 0.5665 - val_loss: 1.4401 - val_
accuracy: 0.3003
Epoch 4/25
84/84 [==============================] - 5s 63ms/step - loss: 0.9905 - accuracy: 0.6224 - val_loss: 1.6021 - val_
accuracy: 0.1982
Epoch 5/25
84/84 [==============================] - 5s 64ms/step - loss: 0.8971 - accuracy: 0.6508 - val_loss: 1.4557 - val_
accuracy: 0.3183
Epoch 6/25
84/84 [==============================] - 5s 62ms/step - loss: 0.7439 - accuracy: 0.7059 - val_loss: 1.4288 - val_
accuracy: 0.3634
Epoch 7/25
84/84 [==============================] - 5s 63ms/step - loss: 0.6867 - accuracy: 0.7242 - val_loss: 1.5538 - val_
accuracy: 0.4204
Epoch 8/25
84/84 [==============================] - 5s 63ms/step - loss: 0.5587 - accuracy: 0.7715 - val_loss: 1.5946 - val_
accuracy: 0.4805
Epoch 9/25
84/84 [==============================] - 5s 63ms/step - loss: 0.4337 - accuracy: 0.8211 - val_loss: 1.7600 - val_
accuracy: 0.4685
Epoch 10/25
84/84 [==============================] - 5s 63ms/step - loss: 0.3530 - accuracy: 0.8632 - val_loss: 1.8927 - val_
accuracy: 0.4354
Epoch 11/25
84/84 [==============================] - 5s 62ms/step - loss: 0.2637 - accuracy: 0.8964 - val_loss: 2.0578 - val_
accuracy: 0.4174
Epoch 12/25
84/84 [==============================] - 5s 62ms/step - loss: 0.1727 - accuracy: 0.9381 - val_loss: 2.2195 - val_
accuracy: 0.4565
Epoch 13/25
84/84 [==============================] - 5s 62ms/step - loss: 0.1209 - accuracy: 0.9631 - val_loss: 2.1506 - val_
accuracy: 0.4655
Epoch 14/25
84/84 [==============================] - 5s 64ms/step - loss: 0.0910 - accuracy: 0.9732 - val_loss: 2.2575 - val_
accuracy: 0.5105
Epoch 15/25
84/84 [==============================] - 5s 63ms/step - loss: 0.0669 - accuracy: 0.9832 - val_loss: 2.2271 - val_
accuracy: 0.4655
```

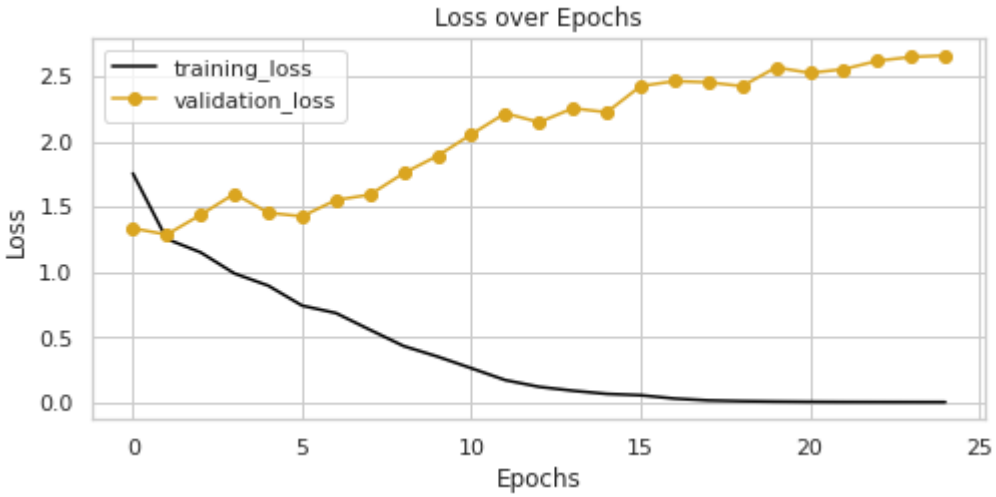
```
Epoch 16/25
84/84 [==============================] - 5s 61ms/step - loss: 0.0564 - accuracy: 0.9914 - val_loss: 2.4276 - val_
accuracy: 0.5225
Epoch 17/25
84/84 [==============================] - 5s 61ms/step - loss: 0.0309 - accuracy: 0.9963 - val_loss: 2.4659 - val_
accuracy: 0.4715
Epoch 18/25
84/84 [==============================] - 5s 61ms/step - loss: 0.0167 - accuracy: 0.9996 - val_loss: 2.4563 - val_
accuracy: 0.5105
Epoch 19/25
84/84 [==============================] - 5s 62ms/step - loss: 0.0113 - accuracy: 0.9996 - val_loss: 2.4272 - val_
accuracy: 0.5135
Epoch 20/25
84/84 [==============================] - 5s 61ms/step - loss: 0.0083 - accuracy: 0.9996 - val_loss: 2.5713 - val_
accuracy: 0.5135
Epoch 21/25
84/84 [==============================] - 5s 62ms/step - loss: 0.0063 - accuracy: 1.0000 - val_loss: 2.5285 - val_
accuracy: 0.5015
Epoch 22/25
84/84 [==============================] - 5s 62ms/step - loss: 0.0052 - accuracy: 1.0000 - val_loss: 2.5575 - val_
accuracy: 0.5045
Epoch 23/25
84/84 [==============================] - 5s 62ms/step - loss: 0.0047 - accuracy: 1.0000 - val_loss: 2.6216 - val_
accuracy: 0.5225
Epoch 24/25
84/84 [==============================] - 5s 61ms/step - loss: 0.0043 - accuracy: 1.0000 - val_loss: 2.6513 - val_
accuracy: 0.5285
Epoch 25/25
84/84 [==============================] - 5s 61ms/step - loss: 0.0040 - accuracy: 1.0000 - val_loss: 2.6624 - val_
accuracy: 0.4955
```

In [ ]:

```
plot_learning_curves(history_cnn3)
```







CNN model with Batchnormalization fits too well to the training set. Model still having a big difficulty with generalization. Our next approach to combat with an overfitting will be a *Data Augmentation* technique. It allows to us artificially increasing the amount of training data by generating new data points from existing data. It can help to improve the model prediction accuracy and reduce data overfitting

Data Augmentation

In [ ]:

```
# Create augmented data generator instance
train_aug_datagen = ImageDataGenerator(rescale=1./255,
                                       brightness_range = (0.5,1.5),
                                       rotation_range=10,
                                       width_shift_range=0.1,
                                       height_shift_range=0.1,
                                       shear_range=0.15,
                                       zoom_range=0.1,
                                       channel_shift_range = 10,
                                       horizontal_flip=True,)

# Augment the whole train images
train_set_augmented = train_aug_datagen.flow_from_directory(train_dir,
                                                            color_mode='rgb',
```

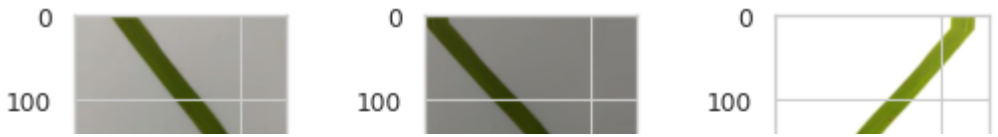
```
                                                    subset='training',
                                                    target_size=(224,224),
                                                    batch_size=128,
                                                    shuffle = True,
                                                    class_mode='categorical')
```

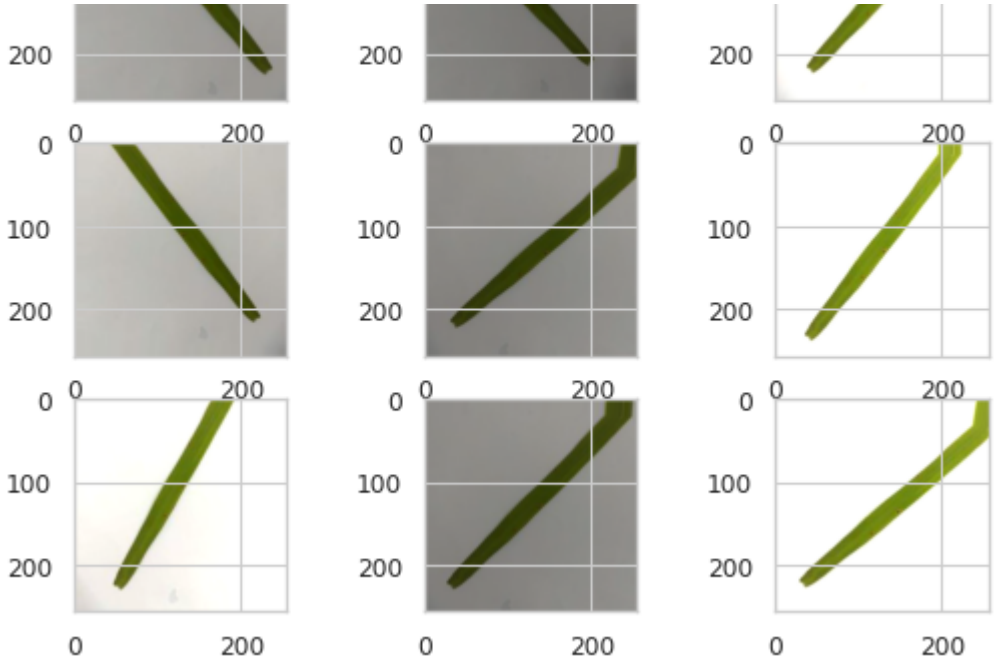
```
Found 2683 images belonging to 4 classes.
```

Let's view few of the augmented images below

In [ ]:

```
# Brown Spot augmented images example
img_bs = load_img('/content/dataset/train/_BrownSpot/shape 100 .jpg')
data = img_to_array(img_bs)
samples = np.expand_dims(data, 0)
# Demonstrating random rotation
datagen = ImageDataGenerator(brightness_range = (0.5,1.5),
                                    rotation_range=10,
                                    width_shift_range=0.1,
                                    height_shift_range=0.1,
                                    shear_range=0.15,
                                    zoom_range=0.1,
                                    channel_shift_range = 10,
                                    horizontal_flip=True,
                                )
it = datagen.flow(samples, batch_size=1)

# Generate samples and plot
plt.figure(figsize=(8, 6), dpi=80)
for i in range(9):
        # define subplot
        plt.subplot(330 + 1 + i)
        # generate batch of images
        batch = it.next()
        # convert to unsigned integers for viewing
        image = batch[0].astype('uint8')
        # plot raw pixel data
        plt.imshow(image)
# show the figure
plt.show();
```

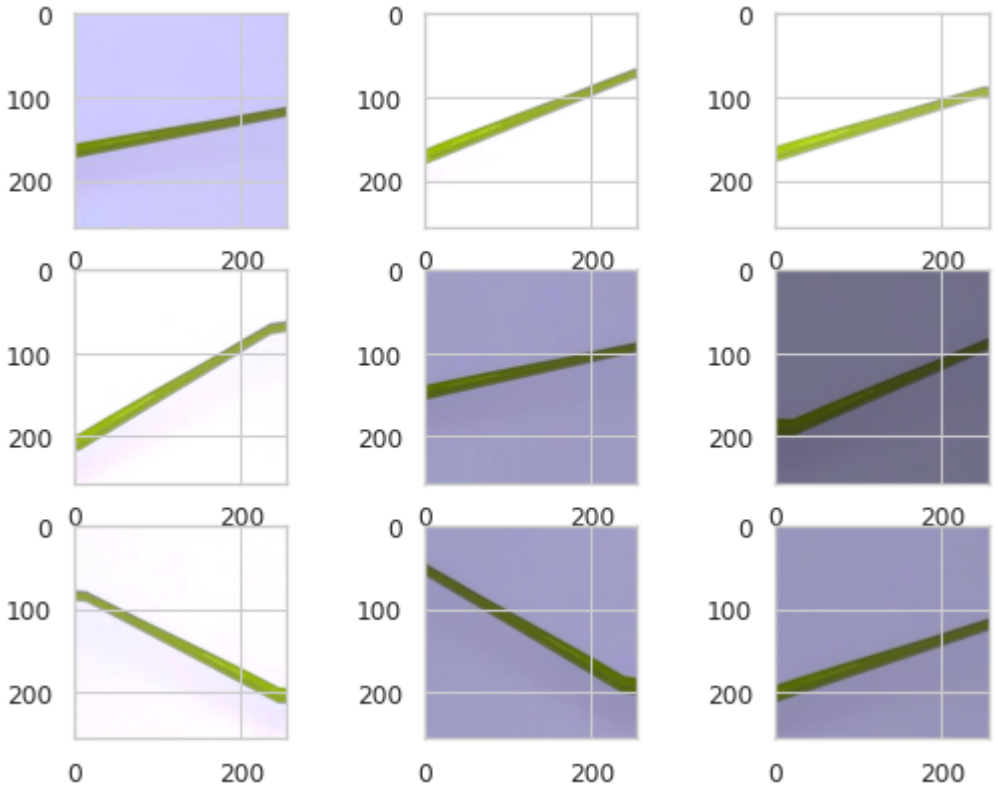




In [ ]:

```
# Hispa augmented images example
img_hispa = load_img("/content/dataset/train/_Hispa/shape 10 .jpg")
data = img_to_array(img_hispa)
samples = np.expand_dims(data, 0)
# Demonstrating random rotation
datagen = ImageDataGenerator(brightness_range = (0.5,1.5),
                                    rotation_range=10,
                                    width_shift_range=0.1,
                                    height_shift_range=0.1,
                                    shear_range=0.15,
                                    zoom_range=0.1,
                                    channel_shift_range = 10,
                                    horizontal_flip=True,
                                )
it = datagen.flow(samples, batch_size=1)
# Generate samples and plot
plt.figure(figsize=(8, 6), dpi=80)
for i in range(9):
        # define subplot
        plt.subplot(330 + 1 + i)
        # generate batch of images
        batch = it.next()
        # convert to unsigned integers for viewing
```

```
        image = batch[0].astype('uint8')
        # plot raw pixel data
        plt.imshow(image)
# show the figure
plt.show();
```



he imbalanced dataset handling has some trade-offs such as whether the accuracy is high but prone to overfit or the accuracy is low with the loss also low. But our target is that we need higher accuracy and lower loss to get a generalized model.Thus, we try to reuse of a pre-trained model on a our case. Namely we will use *Transfer learning* design methodology which is an optimization, a shortcut to saving time or getting better performance. In transfer learning, a machine exploits the knowledge gained from a previous task to improve generalization about another.

#4. Transfer Learning with *ResNet152v2* and *Data Augmentation*

ResNet (Residual Neural Network) was proposed in a 152-layer neural network, which won the championship in the ILSVRC2015 competition. The key idea of ResNet is to add a direct connection channel to the network, which retains a certain proportion of

the output of the previous network layers. That is, the neural network of this layer does not need to learn the entire output, but learns the residual of the previous network output, so ResNet is also called the residual network.

In [ ]:

```
# Create the base pre-trained ResNet152v2 model
base_resnet152v2=tf.keras.applications.ResNet152V2(pooling="avg",
                                          weights="imagenet",
                                          include_top=False,
                                          input_shape=INPUT_SHAPE)

# Set the layers to frozen to keep the weights already made
for layers in base_resnet152v2.layers:
            layers.trainable=False
# Add layers to model
last_output = base_resnet152v2.layers[-1].output
res151v2_x = Flatten()(last_output)
res151v2_x = Dense(128, activation = "relu")(res151v2_x)
res151v2_x = Dense(4, activation = "softmax")(res151v2_x)

# Set the correct inputs and outputs to the model
model_resnet152v2 = tf.keras.Model(base_resnet152v2.input, res151v2_x)

# Compile the model
model_resnet152v2.compile(loss = LOSS_FN,
                          optimizer= tf.keras.optimizers.Adam(),
                          metrics=["accuracy"])

# Return summary of model
#res151v2_final_model.summary()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet152v2_weights_tf_
dim_ordering_tf_kernels_notop.h5
234553344/234545216 [==============================] - 7s 0us/step
234561536/234545216 [==============================] - 7s 0us/step
```

In [ ]:

```
# Check which layers are trainable
for layer_number, layer in enumerate(model_resnet152v2.layers):
  print(layer_number, layer.name, layer.trainable)
```

```
0 input_6 False
1 conv1_pad False
2 conv1_conv False
3 pool1_pad False
4 pool1_pool False
```

```
5 conv2_block1_preact_bn False
6 conv2_block1_preact_relu False
7 conv2_block1_1_conv False
8 conv2_block1_1_bn False
9 conv2_block1_1_relu False
10 conv2_block1_2_pad False
11 conv2_block1_2_conv False
12 conv2_block1_2_bn False
13 conv2_block1_2_relu False
14 conv2_block1_0_conv False
15 conv2_block1_3_conv False
16 conv2_block1_out False
17 conv2_block2_preact_bn False
18 conv2_block2_preact_relu False
19 conv2_block2_1_conv False
20 conv2_block2_1_bn False
21 conv2_block2_1_relu False
22 conv2_block2_2_pad False
23 conv2_block2_2_conv False
24 conv2_block2_2_bn False
25 conv2_block2_2_relu False
26 conv2_block2_3_conv False
27 conv2_block2_out False
28 conv2_block3_preact_bn False
29 conv2_block3_preact_relu False
30 conv2_block3_1_conv False
31 conv2_block3_1_bn False
32 conv2_block3_1_relu False
33 conv2_block3_2_pad False
34 conv2_block3_2_conv False
35 conv2_block3_2_bn False
36 conv2_block3_2_relu False
37 max_pooling2d_22 False
38 conv2_block3_3_conv False
39 conv2_block3_out False
40 conv3_block1_preact_bn False
41 conv3_block1_preact_relu False
42 conv3_block1_1_conv False
43 conv3_block1_1_bn False
44 conv3_block1_1_relu False
45 conv3_block1_2_pad False
46 conv3_block1_2_conv False
47 conv3_block1_2_bn False
48 conv3_block1_2_relu False
49 conv3_block1_0_conv False
50 conv3_block1_3_conv False
```

```
51 conv3_block1_out False
52 conv3_block2_preact_bn False
53 conv3_block2_preact_relu False
54 conv3_block2_1_conv False
55 conv3_block2_1_bn False
56 conv3_block2_1_relu False
57 conv3_block2_2_pad False
58 conv3_block2_2_conv False
59 conv3_block2_2_bn False
60 conv3_block2_2_relu False
61 conv3_block2_3_conv False
62 conv3_block2_out False
63 conv3_block3_preact_bn False
64 conv3_block3_preact_relu False
65 conv3_block3_1_conv False
66 conv3_block3_1_bn False
67 conv3_block3_1_relu False
68 conv3_block3_2_pad False
69 conv3_block3_2_conv False
70 conv3_block3_2_bn False
71 conv3_block3_2_relu False
72 conv3_block3_3_conv False
73 conv3_block3_out False
74 conv3_block4_preact_bn False
75 conv3_block4_preact_relu False
76 conv3_block4_1_conv False
77 conv3_block4_1_bn False
78 conv3_block4_1_relu False
79 conv3_block4_2_pad False
80 conv3_block4_2_conv False
81 conv3_block4_2_bn False
82 conv3_block4_2_relu False
83 conv3_block4_3_conv False
84 conv3_block4_out False
85 conv3_block5_preact_bn False
86 conv3_block5_preact_relu False
87 conv3_block5_1_conv False
88 conv3_block5_1_bn False
89 conv3_block5_1_relu False
90 conv3_block5_2_pad False
91 conv3_block5_2_conv False
92 conv3_block5_2_bn False
93 conv3_block5_2_relu False
94 conv3_block5_3_conv False
95 conv3_block5_out False
96 conv3_block6_preact_bn False
```

```
96 conv3_block6_preact_bn False
97 conv3_block6_preact_relu False
98 conv3_block6_1_conv False
99 conv3_block6_1_bn False
100 conv3_block6_1_relu False
101 conv3_block6_2_pad False
102 conv3_block6_2_conv False
103 conv3_block6_2_bn False
104 conv3_block6_2_relu False
105 conv3_block6_3_conv False
106 conv3_block6_out False
107 conv3_block7_preact_bn False
108 conv3_block7_preact_relu False
109 conv3_block7_1_conv False
110 conv3_block7_1_bn False
111 conv3_block7_1_relu False
112 conv3_block7_2_pad False
113 conv3_block7_2_conv False
114 conv3_block7_2_bn False
115 conv3_block7_2_relu False
116 conv3_block7_3_conv False
117 conv3_block7_out False
118 conv3_block8_preact_bn False
119 conv3_block8_preact_relu False
120 conv3_block8_1_conv False
121 conv3_block8_1_bn False
122 conv3_block8_1_relu False
123 conv3_block8_2_pad False
124 conv3_block8_2_conv False
125 conv3_block8_2_bn False
126 conv3_block8_2_relu False
127 max_pooling2d_23 False
128 conv3_block8_3_conv False
129 conv3_block8_out False
130 conv4_block1_preact_bn False
131 conv4_block1_preact_relu False
132 conv4_block1_1_conv False
133 conv4_block1_1_bn False
134 conv4_block1_1_relu False
135 conv4_block1_2_pad False
136 conv4_block1_2_conv False
137 conv4_block1_2_bn False
138 conv4_block1_2_relu False
139 conv4_block1_0_conv False
140 conv4_block1_3_conv False
141 conv4_block1_out False
```

```
conv4_block2_preact_bn False
conv4_block2_preact_relu False
conv4_block2_1_conv False
conv4_block2_1_bn False
conv4_block2_1_relu False
conv4_block2_2_pad False
conv4_block2_2_conv False
conv4_block2_2_bn False
conv4_block2_2_relu False
conv4_block2_3_conv False
conv4_block2_out False
conv4_block3_preact_bn False
conv4_block3_preact_relu False
conv4_block3_1_conv False
conv4_block3_1_bn False
conv4_block3_1_relu False
conv4_block3_2_pad False
conv4_block3_2_conv False
conv4_block3_2_bn False
conv4_block3_2_relu False
conv4_block3_3_conv False
conv4_block3_out False
conv4_block4_preact_bn False
conv4_block4_preact_relu False
conv4_block4_1_conv False
conv4_block4_1_bn False
conv4_block4_1_relu False
conv4_block4_2_pad False
conv4_block4_2_conv False
conv4_block4_2_bn False
conv4_block4_2_relu False
conv4_block4_3_conv False
conv4_block4_out False
conv4_block5_preact_bn False
conv4_block5_preact_relu False
conv4_block5_1_conv False
conv4_block5_1_bn False
conv4_block5_1_relu False
conv4_block5_2_pad False
conv4_block5_2_conv False
conv4_block5_2_bn False
conv4_block5_2_relu False
conv4_block5_3_conv False
conv4_block5_out False
conv4_block6_preact_bn False
conv4_block6_preact_relu False
```

```
187 conv4_block6_preact_relu False
188 conv4_block6_1_conv False
189 conv4_block6_1_bn False
190 conv4_block6_1_relu False
191 conv4_block6_2_pad False
192 conv4_block6_2_conv False
193 conv4_block6_2_bn False
194 conv4_block6_2_relu False
195 conv4_block6_3_conv False
196 conv4_block6_out False
197 conv4_block7_preact_bn False
198 conv4_block7_preact_relu False
199 conv4_block7_1_conv False
200 conv4_block7_1_bn False
201 conv4_block7_1_relu False
202 conv4_block7_2_pad False
203 conv4_block7_2_conv False
204 conv4_block7_2_bn False
205 conv4_block7_2_relu False
206 conv4_block7_3_conv False
207 conv4_block7_out False
208 conv4_block8_preact_bn False
209 conv4_block8_preact_relu False
210 conv4_block8_1_conv False
211 conv4_block8_1_bn False
212 conv4_block8_1_relu False
213 conv4_block8_2_pad False
214 conv4_block8_2_conv False
215 conv4_block8_2_bn False
216 conv4_block8_2_relu False
217 conv4_block8_3_conv False
218 conv4_block8_out False
219 conv4_block9_preact_bn False
220 conv4_block9_preact_relu False
221 conv4_block9_1_conv False
222 conv4_block9_1_bn False
223 conv4_block9_1_relu False
224 conv4_block9_2_pad False
225 conv4_block9_2_conv False
226 conv4_block9_2_bn False
227 conv4_block9_2_relu False
228 conv4_block9_3_conv False
229 conv4_block9_out False
230 conv4_block10_preact_bn False
231 conv4_block10_preact_relu False
232 conv4_block10_1_conv False
```

```
233 conv4_block10_1_bn False
234 conv4_block10_1_relu False
235 conv4_block10_2_pad False
236 conv4_block10_2_conv False
237 conv4_block10_2_bn False
238 conv4_block10_2_relu False
239 conv4_block10_3_conv False
240 conv4_block10_out False
241 conv4_block11_preact_bn False
242 conv4_block11_preact_relu False
243 conv4_block11_1_conv False
244 conv4_block11_1_bn False
245 conv4_block11_1_relu False
246 conv4_block11_2_pad False
247 conv4_block11_2_conv False
248 conv4_block11_2_bn False
249 conv4_block11_2_relu False
250 conv4_block11_3_conv False
251 conv4_block11_out False
252 conv4_block12_preact_bn False
253 conv4_block12_preact_relu False
254 conv4_block12_1_conv False
255 conv4_block12_1_bn False
256 conv4_block12_1_relu False
257 conv4_block12_2_pad False
258 conv4_block12_2_conv False
259 conv4_block12_2_bn False
260 conv4_block12_2_relu False
261 conv4_block12_3_conv False
262 conv4_block12_out False
263 conv4_block13_preact_bn False
264 conv4_block13_preact_relu False
265 conv4_block13_1_conv False
266 conv4_block13_1_bn False
267 conv4_block13_1_relu False
268 conv4_block13_2_pad False
269 conv4_block13_2_conv False
270 conv4_block13_2_bn False
271 conv4_block13_2_relu False
272 conv4_block13_3_conv False
273 conv4_block13_out False
274 conv4_block14_preact_bn False
275 conv4_block14_preact_relu False
276 conv4_block14_1_conv False
277 conv4_block14_1_bn False
278 conv4_block14_1_relu False
```

```
279 conv4_block14_2_pad False
280 conv4_block14_2_conv False
281 conv4_block14_2_bn False
282 conv4_block14_2_relu False
283 conv4_block14_3_conv False
284 conv4_block14_out False
285 conv4_block15_preact_bn False
286 conv4_block15_preact_relu False
287 conv4_block15_1_conv False
288 conv4_block15_1_bn False
289 conv4_block15_1_relu False
290 conv4_block15_2_pad False
291 conv4_block15_2_conv False
292 conv4_block15_2_bn False
293 conv4_block15_2_relu False
294 conv4_block15_3_conv False
295 conv4_block15_out False
296 conv4_block16_preact_bn False
297 conv4_block16_preact_relu False
298 conv4_block16_1_conv False
299 conv4_block16_1_bn False
300 conv4_block16_1_relu False
301 conv4_block16_2_pad False
302 conv4_block16_2_conv False
303 conv4_block16_2_bn False
304 conv4_block16_2_relu False
305 conv4_block16_3_conv False
306 conv4_block16_out False
307 conv4_block17_preact_bn False
308 conv4_block17_preact_relu False
309 conv4_block17_1_conv False
310 conv4_block17_1_bn False
311 conv4_block17_1_relu False
312 conv4_block17_2_pad False
313 conv4_block17_2_conv False
314 conv4_block17_2_bn False
315 conv4_block17_2_relu False
316 conv4_block17_3_conv False
317 conv4_block17_out False
318 conv4_block18_preact_bn False
319 conv4_block18_preact_relu False
320 conv4_block18_1_conv False
321 conv4_block18_1_bn False
322 conv4_block18_1_relu False
323 conv4_block18_2_pad False
```

```
conv4_block18_2_conv False
conv4_block18_2_bn False
conv4_block18_2_relu False
conv4_block18_3_conv False
conv4_block18_out False
conv4_block19_preact_bn False
conv4_block19_preact_relu False
conv4_block19_1_conv False
conv4_block19_1_bn False
conv4_block19_1_relu False
conv4_block19_2_pad False
conv4_block19_2_conv False
conv4_block19_2_bn False
conv4_block19_2_relu False
conv4_block19_3_conv False
conv4_block19_out False
conv4_block20_preact_bn False
conv4_block20_preact_relu False
conv4_block20_1_conv False
conv4_block20_1_bn False
conv4_block20_1_relu False
conv4_block20_2_pad False
conv4_block20_2_conv False
conv4_block20_2_bn False
conv4_block20_2_relu False
conv4_block20_3_conv False
conv4_block20_out False
conv4_block21_preact_bn False
conv4_block21_preact_relu False
conv4_block21_1_conv False
conv4_block21_1_bn False
conv4_block21_1_relu False
conv4_block21_2_pad False
conv4_block21_2_conv False
conv4_block21_2_bn False
conv4_block21_2_relu False
conv4_block21_3_conv False
conv4_block21_out False
conv4_block22_preact_bn False
conv4_block22_preact_relu False
conv4_block22_1_conv False
conv4_block22_1_bn False
conv4_block22_1_relu False
conv4_block22_2_pad False
conv4_block22_2_conv False
conv4_block22_2_bn False
```

```
370 conv4_block22_2_relu False
371 conv4_block22_3_conv False
372 conv4_block22_out False
373 conv4_block23_preact_bn False
374 conv4_block23_preact_relu False
375 conv4_block23_1_conv False
376 conv4_block23_1_bn False
377 conv4_block23_1_relu False
378 conv4_block23_2_pad False
379 conv4_block23_2_conv False
380 conv4_block23_2_bn False
381 conv4_block23_2_relu False
382 conv4_block23_3_conv False
383 conv4_block23_out False
384 conv4_block24_preact_bn False
385 conv4_block24_preact_relu False
386 conv4_block24_1_conv False
387 conv4_block24_1_bn False
388 conv4_block24_1_relu False
389 conv4_block24_2_pad False
390 conv4_block24_2_conv False
391 conv4_block24_2_bn False
392 conv4_block24_2_relu False
393 conv4_block24_3_conv False
394 conv4_block24_out False
395 conv4_block25_preact_bn False
396 conv4_block25_preact_relu False
397 conv4_block25_1_conv False
398 conv4_block25_1_bn False
399 conv4_block25_1_relu False
400 conv4_block25_2_pad False
401 conv4_block25_2_conv False
402 conv4_block25_2_bn False
403 conv4_block25_2_relu False
404 conv4_block25_3_conv False
405 conv4_block25_out False
406 conv4_block26_preact_bn False
407 conv4_block26_preact_relu False
408 conv4_block26_1_conv False
409 conv4_block26_1_bn False
410 conv4_block26_1_relu False
411 conv4_block26_2_pad False
412 conv4_block26_2_conv False
413 conv4_block26_2_bn False
414 conv4_block26_2_relu False
```

```
conv4_block26_3_conv False
conv4_block26_out False
conv4_block27_preact_bn False
conv4_block27_preact_relu False
conv4_block27_1_conv False
conv4_block27_1_bn False
conv4_block27_1_relu False
conv4_block27_2_pad False
conv4_block27_2_conv False
conv4_block27_2_bn False
conv4_block27_2_relu False
conv4_block27_3_conv False
conv4_block27_out False
conv4_block28_preact_bn False
conv4_block28_preact_relu False
conv4_block28_1_conv False
conv4_block28_1_bn False
conv4_block28_1_relu False
conv4_block28_2_pad False
conv4_block28_2_conv False
conv4_block28_2_bn False
conv4_block28_2_relu False
conv4_block28_3_conv False
conv4_block28_out False
conv4_block29_preact_bn False
conv4_block29_preact_relu False
conv4_block29_1_conv False
conv4_block29_1_bn False
conv4_block29_1_relu False
conv4_block29_2_pad False
conv4_block29_2_conv False
conv4_block29_2_bn False
conv4_block29_2_relu False
conv4_block29_3_conv False
conv4_block29_out False
conv4_block30_preact_bn False
conv4_block30_preact_relu False
conv4_block30_1_conv False
conv4_block30_1_bn False
conv4_block30_1_relu False
conv4_block30_2_pad False
conv4_block30_2_conv False
conv4_block30_2_bn False
conv4_block30_2_relu False
conv4_block30_3_conv False
conv4_block30_out False
```

```
461 conv4_block31_preact_bn False
462 conv4_block31_preact_relu False
463 conv4_block31_1_conv False
464 conv4_block31_1_bn False
465 conv4_block31_1_relu False
466 conv4_block31_2_pad False
467 conv4_block31_2_conv False
468 conv4_block31_2_bn False
469 conv4_block31_2_relu False
470 conv4_block31_3_conv False
471 conv4_block31_out False
472 conv4_block32_preact_bn False
473 conv4_block32_preact_relu False
474 conv4_block32_1_conv False
475 conv4_block32_1_bn False
476 conv4_block32_1_relu False
477 conv4_block32_2_pad False
478 conv4_block32_2_conv False
479 conv4_block32_2_bn False
480 conv4_block32_2_relu False
481 conv4_block32_3_conv False
482 conv4_block32_out False
483 conv4_block33_preact_bn False
484 conv4_block33_preact_relu False
485 conv4_block33_1_conv False
486 conv4_block33_1_bn False
487 conv4_block33_1_relu False
488 conv4_block33_2_pad False
489 conv4_block33_2_conv False
490 conv4_block33_2_bn False
491 conv4_block33_2_relu False
492 conv4_block33_3_conv False
493 conv4_block33_out False
494 conv4_block34_preact_bn False
495 conv4_block34_preact_relu False
496 conv4_block34_1_conv False
497 conv4_block34_1_bn False
498 conv4_block34_1_relu False
499 conv4_block34_2_pad False
500 conv4_block34_2_conv False
501 conv4_block34_2_bn False
502 conv4_block34_2_relu False
503 conv4_block34_3_conv False
504 conv4_block34_out False
505 conv4_block35_preact_bn False
```

```
506 conv4_block35_preact_relu False
507 conv4_block35_1_conv False
508 conv4_block35_1_bn False
509 conv4_block35_1_relu False
510 conv4_block35_2_pad False
511 conv4_block35_2_conv False
512 conv4_block35_2_bn False
513 conv4_block35_2_relu False
514 conv4_block35_3_conv False
515 conv4_block35_out False
516 conv4_block36_preact_bn False
517 conv4_block36_preact_relu False
518 conv4_block36_1_conv False
519 conv4_block36_1_bn False
520 conv4_block36_1_relu False
521 conv4_block36_2_pad False
522 conv4_block36_2_conv False
523 conv4_block36_2_bn False
524 conv4_block36_2_relu False
525 max_pooling2d_24 False
526 conv4_block36_3_conv False
527 conv4_block36_out False
528 conv5_block1_preact_bn False
529 conv5_block1_preact_relu False
530 conv5_block1_1_conv False
531 conv5_block1_1_bn False
532 conv5_block1_1_relu False
533 conv5_block1_2_pad False
534 conv5_block1_2_conv False
535 conv5_block1_2_bn False
536 conv5_block1_2_relu False
537 conv5_block1_0_conv False
538 conv5_block1_3_conv False
539 conv5_block1_out False
540 conv5_block2_preact_bn False
541 conv5_block2_preact_relu False
542 conv5_block2_1_conv False
543 conv5_block2_1_bn False
544 conv5_block2_1_relu False
545 conv5_block2_2_pad False
546 conv5_block2_2_conv False
547 conv5_block2_2_bn False
548 conv5_block2_2_relu False
549 conv5_block2_3_conv False
550 conv5_block2_out False
551 conv5_block3_preact_bn False
```

```
552 conv5_block3_preact_relu False
553 conv5_block3_1_conv False
554 conv5_block3_1_bn False
555 conv5_block3_1_relu False
556 conv5_block3_2_pad False
557 conv5_block3_2_conv False
558 conv5_block3_2_bn False
559 conv5_block3_2_relu False
560 conv5_block3_3_conv False
561 conv5_block3_out False
562 post_bn False
563 post_relu False
564 avg_pool False
565 flatten_13 True
566 dense_24 True
567 dense_25 True
```

In [ ]:

```
# Fit the model ResNet152v2
history_res151v2 = model_resnet152v2.fit(train_set_augmented,
                                         epochs=50,
                                         validation_data=val_set,
                                         verbose=2)
```

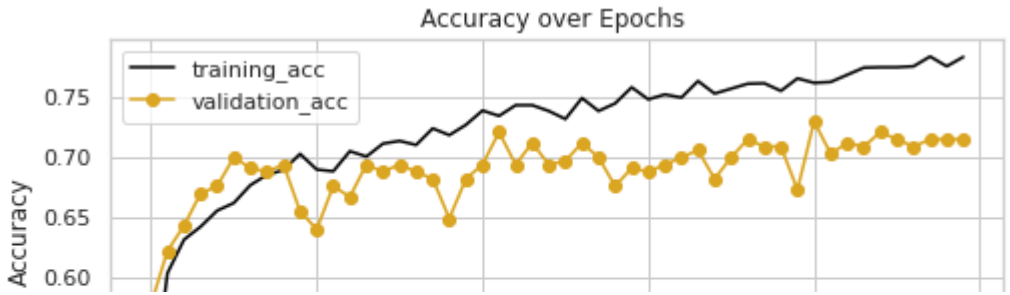
```
Epoch 1/50
21/21 - 44s - loss: 1.2036 - accuracy: 0.4946 - val_loss: 1.0205 - val_accuracy: 0.5766 - 44s/epoch - 2s/step
Epoch 2/50
21/21 - 34s - loss: 0.9881 - accuracy: 0.6031 - val_loss: 0.8955 - val_accuracy: 0.6216 - 34s/epoch - 2s/step
Epoch 3/50
21/21 - 34s - loss: 0.9190 - accuracy: 0.6310 - val_loss: 0.8540 - val_accuracy: 0.6426 - 34s/epoch - 2s/step
Epoch 4/50
21/21 - 34s - loss: 0.8951 - accuracy: 0.6418 - val_loss: 0.8295 - val_accuracy: 0.6697 - 34s/epoch - 2s/step
Epoch 5/50
21/21 - 34s - loss: 0.8748 - accuracy: 0.6552 - val_loss: 0.8314 - val_accuracy: 0.6757 - 34s/epoch - 2s/step
Epoch 6/50
21/21 - 34s - loss: 0.8537 - accuracy: 0.6616 - val_loss: 0.7941 - val_accuracy: 0.6997 - 34s/epoch - 2s/step
Epoch 7/50
21/21 - 34s - loss: 0.8309 - accuracy: 0.6765 - val_loss: 0.7874 - val_accuracy: 0.6907 - 34s/epoch - 2s/step
Epoch 8/50
21/21 - 34s - loss: 0.8099 - accuracy: 0.6851 - val_loss: 0.7771 - val_accuracy: 0.6877 - 34s/epoch - 2s/step
Epoch 9/50
21/21 - 34s - loss: 0.7838 - accuracy: 0.6892 - val_loss: 0.7769 - val_accuracy: 0.6937 - 34s/epoch - 2s/step
Epoch 10/50
21/21 - 34s - loss: 0.7641 - accuracy: 0.7026 - val_loss: 0.8047 - val_accuracy: 0.6547 - 34s/epoch - 2s/step
Epoch 11/50
21/21 - 34s - loss: 0.7875 - accuracy: 0.6895 - val_loss: 0.8084 - val_accuracy: 0.6396 - 34s/epoch - 2s/step
```

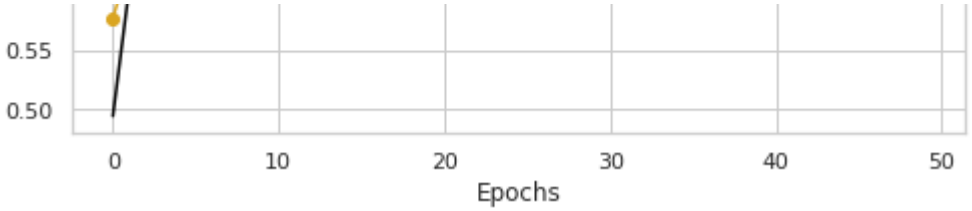
```
Epoch 12/50
21/21 - 34s - loss: 0.7917 - accuracy: 0.6880 - val_loss: 0.7765 - val_accuracy: 0.6757 - 34s/epoch - 2s/step
Epoch 13/50
21/21 - 34s - loss: 0.7522 - accuracy: 0.7048 - val_loss: 0.8099 - val_accuracy: 0.6667 - 34s/epoch - 2s/step
Epoch 14/50
21/21 - 34s - loss: 0.7566 - accuracy: 0.7003 - val_loss: 0.7439 - val_accuracy: 0.6937 - 34s/epoch - 2s/step
Epoch 15/50
21/21 - 34s - loss: 0.7408 - accuracy: 0.7111 - val_loss: 0.7530 - val_accuracy: 0.6877 - 34s/epoch - 2s/step
Epoch 16/50
21/21 - 34s - loss: 0.7252 - accuracy: 0.7134 - val_loss: 0.7528 - val_accuracy: 0.6937 - 34s/epoch - 2s/step
Epoch 17/50
21/21 - 34s - loss: 0.7084 - accuracy: 0.7100 - val_loss: 0.7351 - val_accuracy: 0.6877 - 34s/epoch - 2s/step
Epoch 18/50
21/21 - 34s - loss: 0.7062 - accuracy: 0.7238 - val_loss: 0.7542 - val_accuracy: 0.6817 - 34s/epoch - 2s/step
Epoch 19/50
21/21 - 34s - loss: 0.7337 - accuracy: 0.7182 - val_loss: 0.7838 - val_accuracy: 0.6486 - 34s/epoch - 2s/step
Epoch 20/50
21/21 - 34s - loss: 0.7097 - accuracy: 0.7268 - val_loss: 0.7630 - val_accuracy: 0.6817 - 34s/epoch - 2s/step
Epoch 21/50
21/21 - 34s - loss: 0.6876 - accuracy: 0.7387 - val_loss: 0.7602 - val_accuracy: 0.6937 - 34s/epoch - 2s/step
Epoch 22/50
21/21 - 34s - loss: 0.6874 - accuracy: 0.7343 - val_loss: 0.7336 - val_accuracy: 0.7207 - 34s/epoch - 2s/step
Epoch 23/50
21/21 - 34s - loss: 0.6811 - accuracy: 0.7432 - val_loss: 0.7638 - val_accuracy: 0.6937 - 34s/epoch - 2s/step
Epoch 24/50
21/21 - 33s - loss: 0.6813 - accuracy: 0.7432 - val_loss: 0.7371 - val_accuracy: 0.7117 - 33s/epoch - 2s/step
Epoch 25/50
21/21 - 34s - loss: 0.6626 - accuracy: 0.7384 - val_loss: 0.7442 - val_accuracy: 0.6937 - 34s/epoch - 2s/step
Epoch 26/50
21/21 - 34s - loss: 0.6767 - accuracy: 0.7316 - val_loss: 0.7444 - val_accuracy: 0.6967 - 34s/epoch - 2s/step
Epoch 27/50
21/21 - 34s - loss: 0.6558 - accuracy: 0.7492 - val_loss: 0.7479 - val_accuracy: 0.7117 - 34s/epoch - 2s/step
Epoch 28/50
21/21 - 34s - loss: 0.6685 - accuracy: 0.7384 - val_loss: 0.7617 - val_accuracy: 0.6997 - 34s/epoch - 2s/step
Epoch 29/50
21/21 - 34s - loss: 0.6602 - accuracy: 0.7447 - val_loss: 0.7673 - val_accuracy: 0.6757 - 34s/epoch - 2s/step
Epoch 30/50
21/21 - 34s - loss: 0.6308 - accuracy: 0.7581 - val_loss: 0.7519 - val_accuracy: 0.6907 - 34s/epoch - 2s/step
Epoch 31/50
21/21 - 34s - loss: 0.6372 - accuracy: 0.7480 - val_loss: 0.7686 - val_accuracy: 0.6877 - 34s/epoch - 2s/step
Epoch 32/50
21/21 - 34s - loss: 0.6369 - accuracy: 0.7521 - val_loss: 0.7682 - val_accuracy: 0.6937 - 34s/epoch - 2s/step
Epoch 33/50
21/21 - 33s - loss: 0.6243 - accuracy: 0.7495 - val_loss: 0.7599 - val_accuracy: 0.6997 - 33s/epoch - 2s/step
Epoch 34/50
```

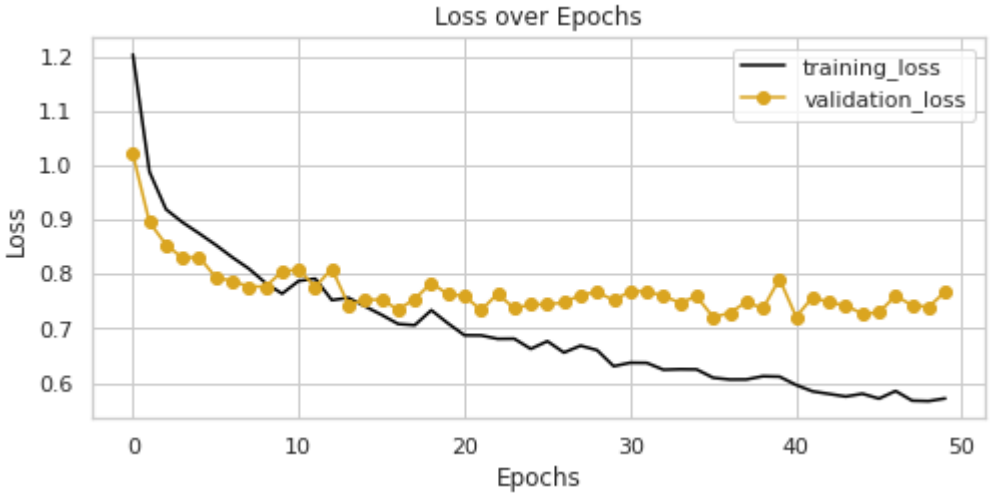
```
21/21 - 34s - loss: 0.6252 - accuracy: 0.7633 - val_loss: 0.7473 - val_accuracy: 0.7057 - 34s/epoch - 2s/step
Epoch 35/50
21/21 - 34s - loss: 0.6249 - accuracy: 0.7529 - val_loss: 0.7595 - val_accuracy: 0.6817 - 34s/epoch - 2s/step
Epoch 36/50
21/21 - 34s - loss: 0.6096 - accuracy: 0.7570 - val_loss: 0.7208 - val_accuracy: 0.6997 - 34s/epoch - 2s/step
Epoch 37/50
21/21 - 34s - loss: 0.6064 - accuracy: 0.7611 - val_loss: 0.7286 - val_accuracy: 0.7147 - 34s/epoch - 2s/step
Epoch 38/50
21/21 - 34s - loss: 0.6065 - accuracy: 0.7615 - val_loss: 0.7490 - val_accuracy: 0.7087 - 34s/epoch - 2s/step
Epoch 39/50
21/21 - 34s - loss: 0.6124 - accuracy: 0.7551 - val_loss: 0.7367 - val_accuracy: 0.7087 - 34s/epoch - 2s/step
Epoch 40/50
21/21 - 34s - loss: 0.6116 - accuracy: 0.7656 - val_loss: 0.7913 - val_accuracy: 0.6727 - 34s/epoch - 2s/step
Epoch 41/50
21/21 - 34s - loss: 0.5963 - accuracy: 0.7618 - val_loss: 0.7215 - val_accuracy: 0.7297 - 34s/epoch - 2s/step
Epoch 42/50
21/21 - 34s - loss: 0.5846 - accuracy: 0.7626 - val_loss: 0.7576 - val_accuracy: 0.7027 - 34s/epoch - 2s/step
Epoch 43/50
21/21 - 34s - loss: 0.5798 - accuracy: 0.7685 - val_loss: 0.7479 - val_accuracy: 0.7117 - 34s/epoch - 2s/step
Epoch 44/50
21/21 - 34s - loss: 0.5751 - accuracy: 0.7745 - val_loss: 0.7425 - val_accuracy: 0.7087 - 34s/epoch - 2s/step
Epoch 45/50
21/21 - 34s - loss: 0.5802 - accuracy: 0.7749 - val_loss: 0.7268 - val_accuracy: 0.7207 - 34s/epoch - 2s/step
Epoch 46/50
21/21 - 34s - loss: 0.5712 - accuracy: 0.7749 - val_loss: 0.7297 - val_accuracy: 0.7147 - 34s/epoch - 2s/step
Epoch 47/50
21/21 - 34s - loss: 0.5853 - accuracy: 0.7756 - val_loss: 0.7602 - val_accuracy: 0.7087 - 34s/epoch - 2s/step
Epoch 48/50
21/21 - 34s - loss: 0.5675 - accuracy: 0.7838 - val_loss: 0.7426 - val_accuracy: 0.7147 - 34s/epoch - 2s/step
Epoch 49/50
21/21 - 34s - loss: 0.5667 - accuracy: 0.7756 - val_loss: 0.7398 - val_accuracy: 0.7147 - 34s/epoch - 2s/step
Epoch 50/50
21/21 - 34s - loss: 0.5717 - accuracy: 0.7835 - val_loss: 0.7679 - val_accuracy: 0.7147 - 34s/epoch - 2s/step
```

In [ ]:

```
plot_learning_curves(history_res151v2)
```





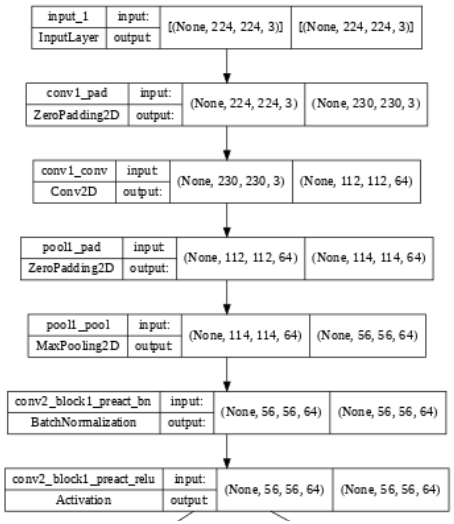


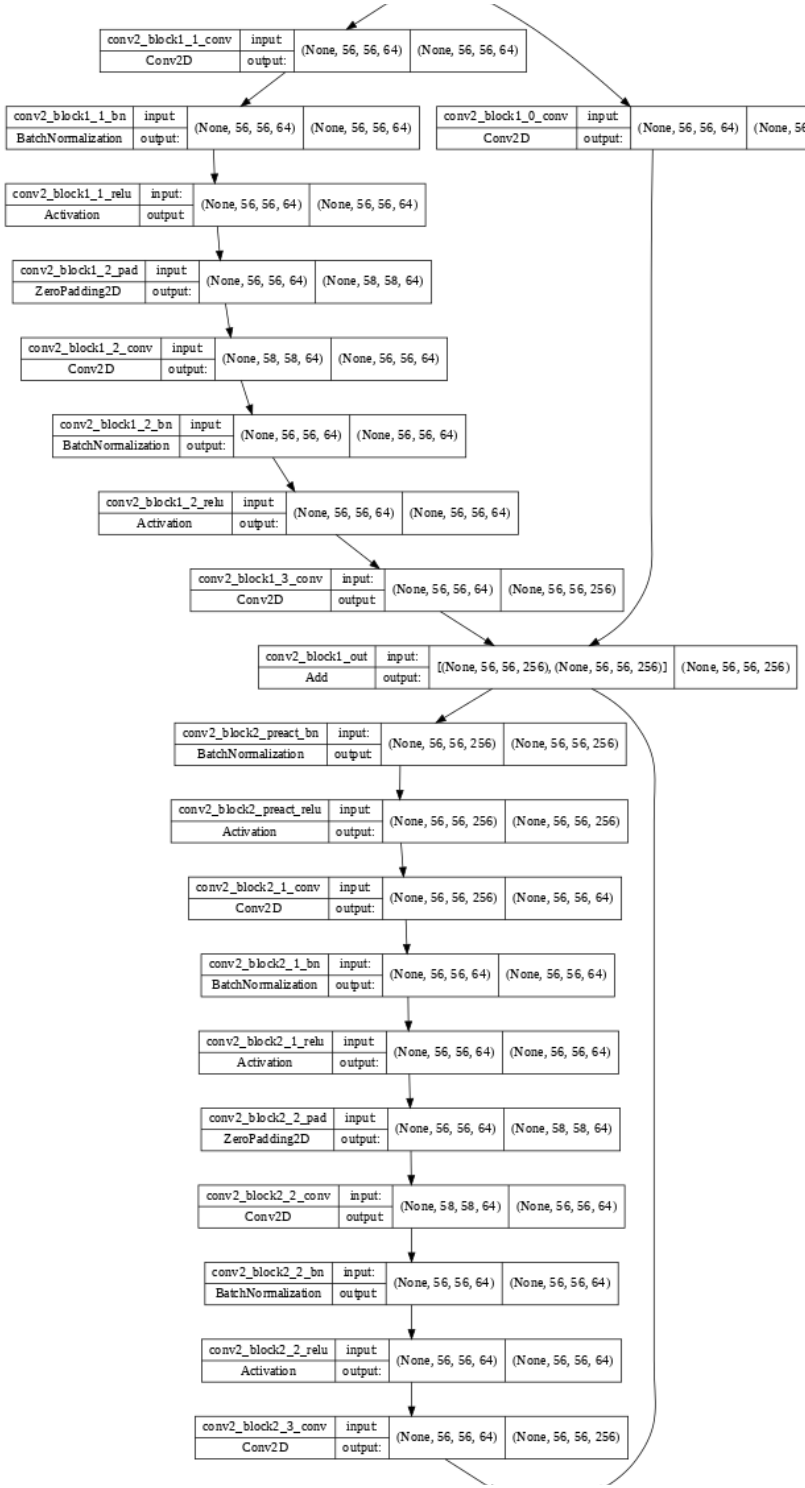
In [ ]:

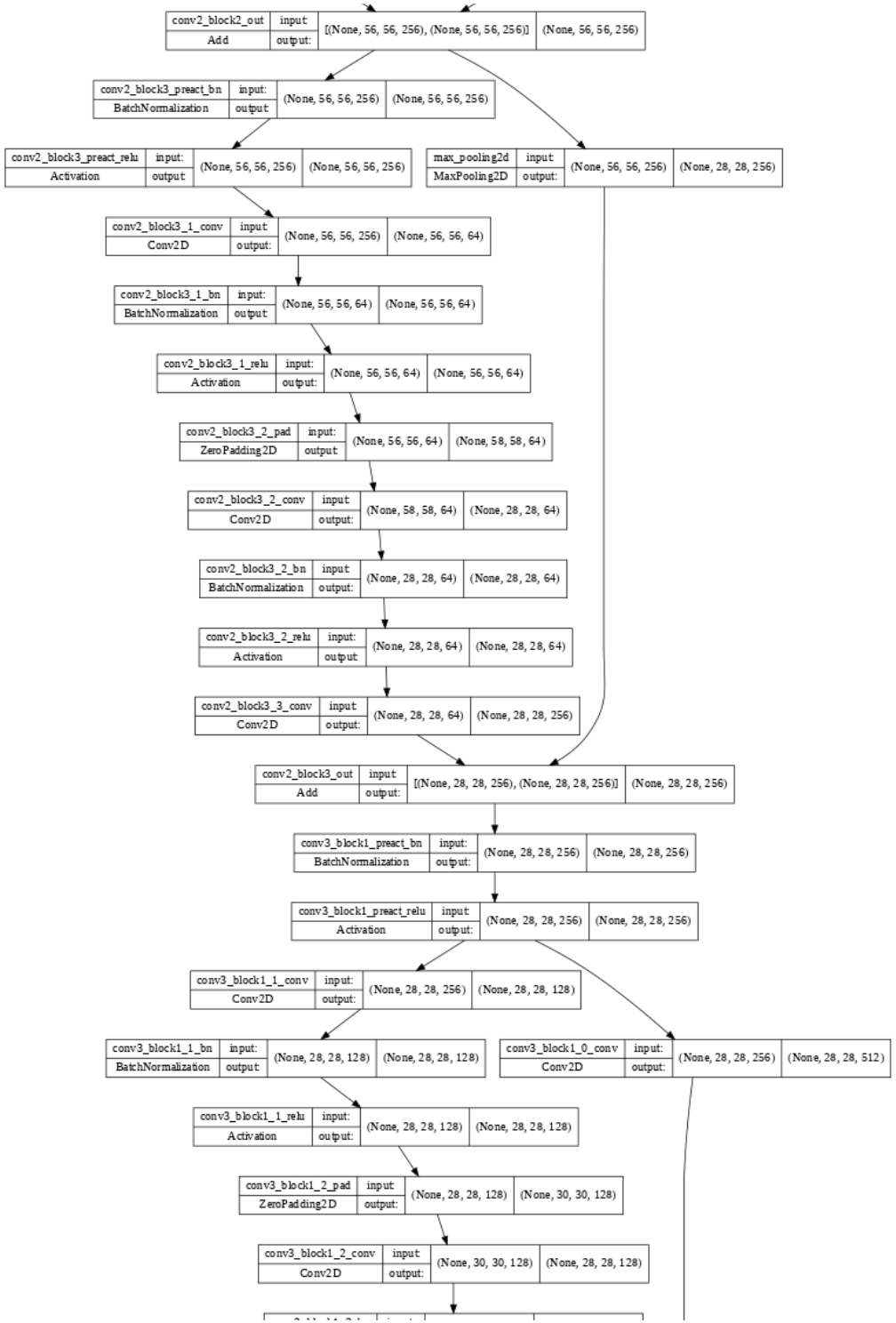
```
# Plot model architecture
tf.keras.utils.plot_model(model_resnet152v2, show_shapes=True, show_layer_names=True)
```

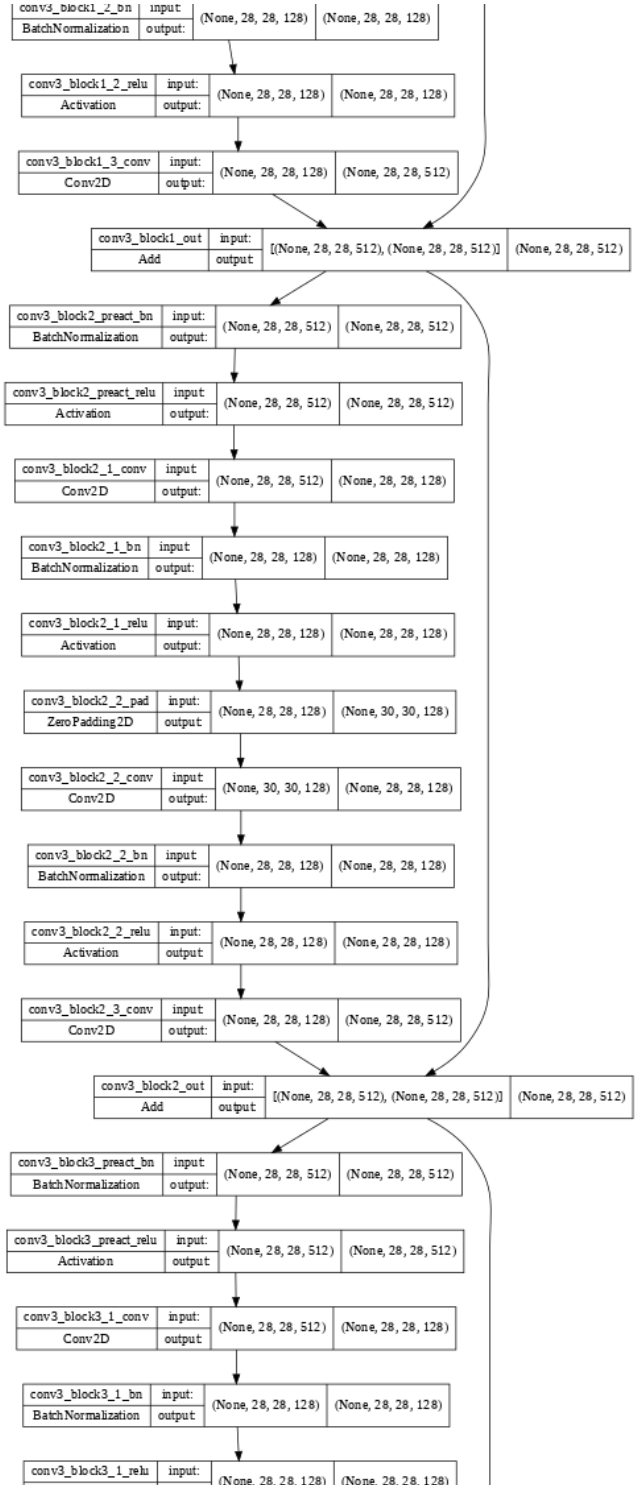
```
dot: graph is too large for cairo-renderer bitmaps. Scaling by 0.5281 to fit
```

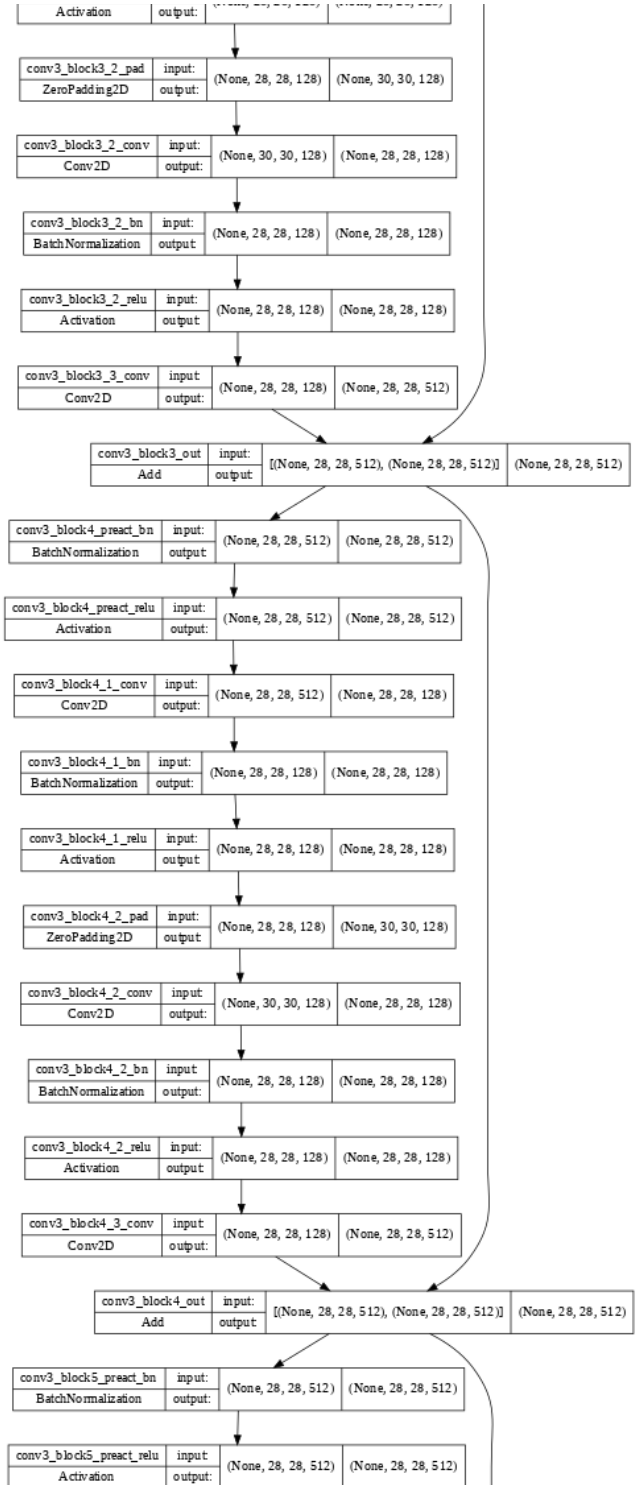
Out[ ]:

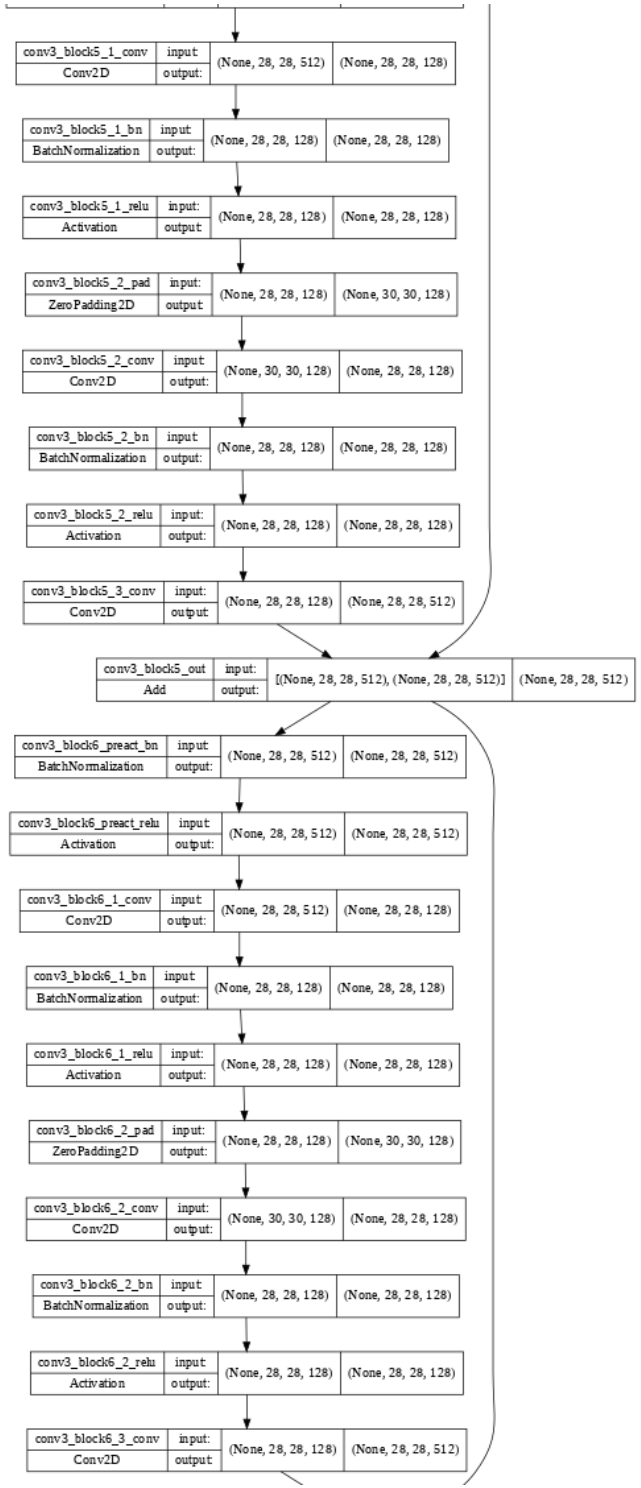


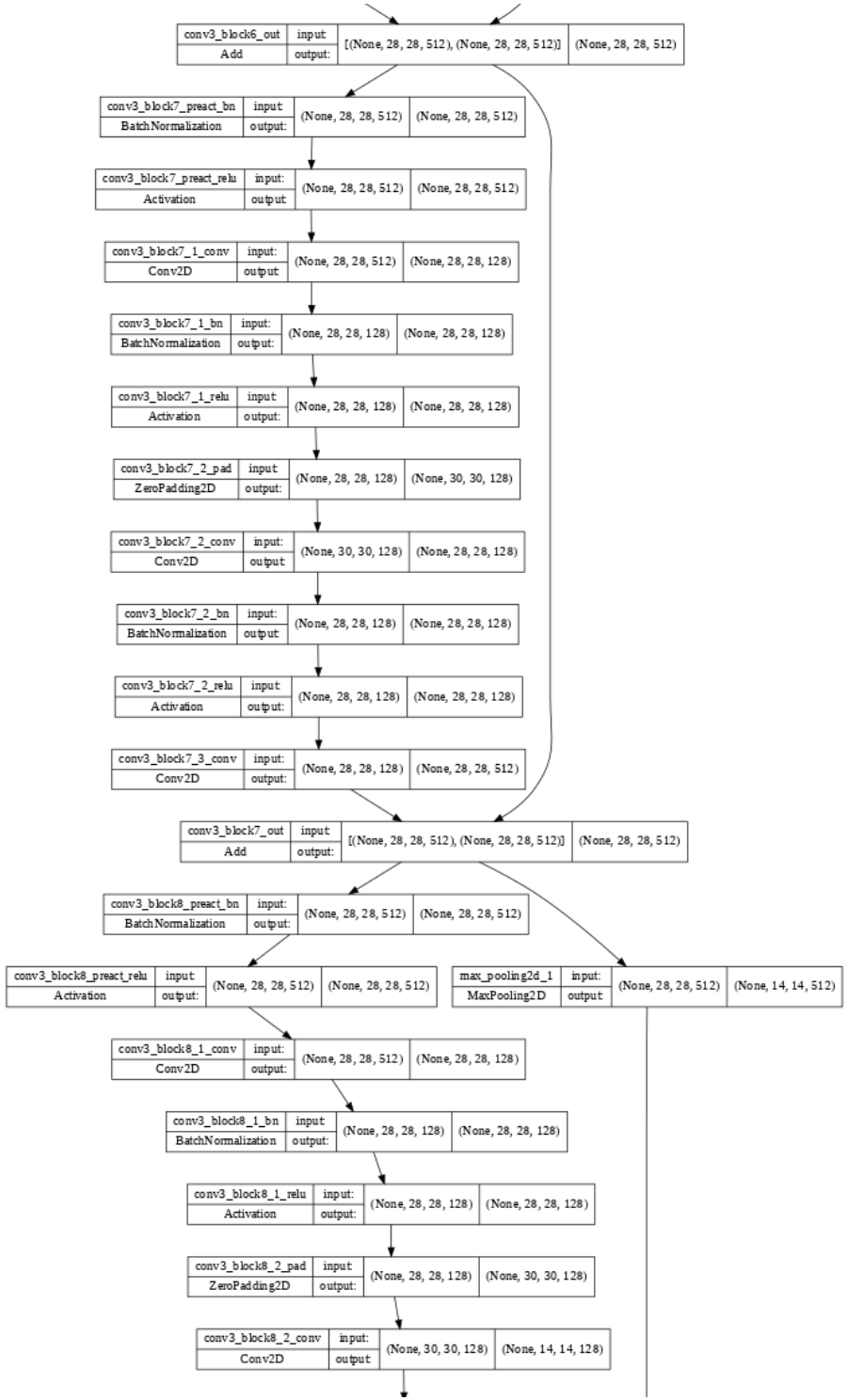

conv2_block1_1_conv | Conv2D | input: | output: | (None, 56, 56, 64) | (None, 56, 56, 64)
conv2_block1_1_bn | BatchNormalization | input: | output: | (None, 56, 56, 64) | (None, 56, 56, 64)
conv2_block1_0_conv | Conv2D | input: | output: | (None, 56, 56, 64) | (None, 56, 56, 256)
conv2_block1_1_relu | Activation | input: | output: | (None, 56, 56, 64) | (None, 56, 56, 64)
conv2_block1_2_pad | ZeroPadding2D | input: | output: | (None, 56, 56, 64) | (None, 58, 58, 64)
conv2_block1_2_conv | Conv2D | input: | output: | (None, 58, 58, 64) | (None, 56, 56, 64)
conv2_block1_2_bn | BatchNormalization | input: | output: | (None, 56, 56, 64) | (None, 56, 56, 64)
conv2_block1_2_relu | Activation | input: | output: | (None, 56, 56, 64) | (None, 56, 56, 64)
conv2_block1_3_conv | Conv2D | input: | output | (None, 56, 56, 64) | (None, 56, 56, 256)
conv2_block1_out | Add | input: | output: | [(None, 56, 56, 256), (None, 56, 56, 256)] | (None, 56, 56, 256)
conv2_block2_preact_bn | BatchNormalization | input: | output | (None, 56, 56, 256) | (None, 56, 56, 256)
conv2_block2_preact_relu | Activation | input: | output: | (None, 56, 56, 256) | (None, 56, 56, 256)
conv2_block2_1_conv | Conv2D | input: | output: | (None, 56, 56, 256) | (None, 56, 56, 64)
conv2_block2_1_bn | BatchNormalization | input: | output: | (None, 56, 56, 64) | (None, 56, 56, 64)
conv2_block2_1_relu | Activation | input: | output: | (None, 56, 56, 64) | (None, 56, 56, 64)
conv2_block2_2_pad | ZeroPadding2D | input: | output: | (None, 56, 56, 64) | (None, 58, 58, 64)
conv2_block2_2_conv | Conv2D | input: | output | (None, 58, 58, 64) | (None, 56, 56, 64)
conv2_block2_2_bn | BatchNormalization | input: | output | (None, 56, 56, 64) | (None, 56, 56, 64)
conv2_block2_2_relu | Activation | input: | output: | (None, 56, 56, 64) | (None, 56, 56, 64)
conv2_block2_3_conv | Conv2D | input: | output: | (None, 56, 56, 64) | (None, 56, 56, 256)

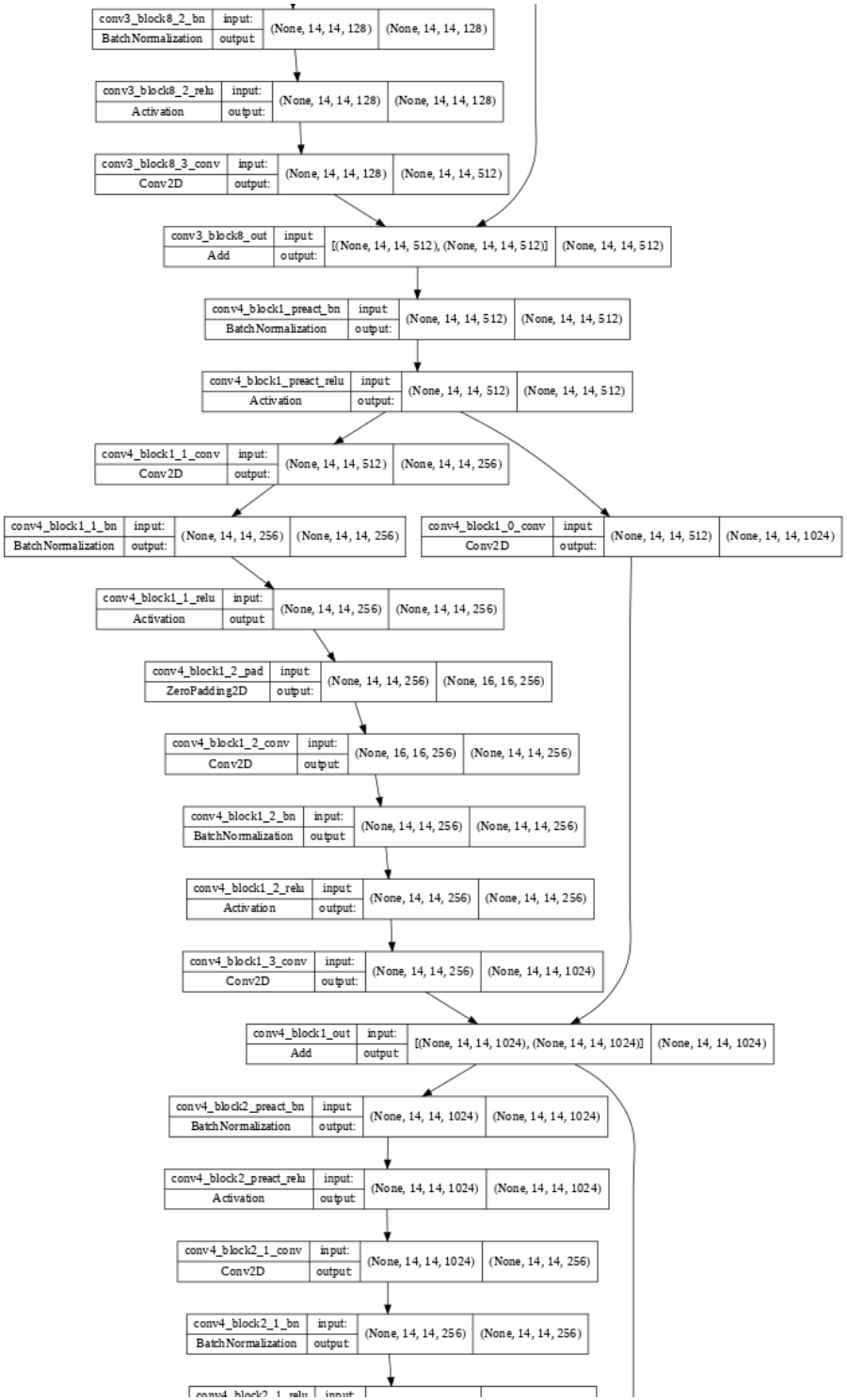

conv2_block2_out | input: | [(None, 56, 56, 256), (None, 56, 56, 256)] | (None, 56, 56, 256)
Add | output:
conv2_block3_preact_bn | input: | (None, 56, 56, 256) | (None, 56, 56, 256)
BatchNormalization | output:
conv2_block3_preact_relu | input: | (None, 56, 56, 256) | (None, 56, 56, 256)
Activation | output:
max_pooling2d | input: | (None, 56, 56, 256) | (None, 28, 28, 256)
MaxPooling2D | output:
conv2_block3_1_conv | input: | (None, 56, 56, 256) | (None, 56, 56, 64)
Conv2D | output:
conv2_block3_1_bn | input: | (None, 56, 56, 64) | (None, 56, 56, 64)
BatchNormalization | output:
conv2_block3_1_relu | input: | (None, 56, 56, 64) | (None, 56, 56, 64)
Activation | output:
conv2_block3_2_pad | input: | (None, 56, 56, 64) | (None, 58, 58, 64)
ZeroPadding2D | output:
conv2_block3_2_conv | input: | (None, 58, 58, 64) | (None, 28, 28, 64)
Conv2D | output:
conv2_block3_2_bn | input: | (None, 28, 28, 64) | (None, 28, 28, 64)
BatchNormalization | output:
conv2_block3_2_relu | input: | (None, 28, 28, 64) | (None, 28, 28, 64)
Activation | output:
conv2_block3_3_conv | input: | (None, 28, 28, 64) | (None, 28, 28, 256)
Conv2D | output:
conv2_block3_out | input: | [(None, 28, 28, 256), (None, 28, 28, 256)] | (None, 28, 28, 256)
Add | output:
conv3_block1_preact_bn | input: | (None, 28, 28, 256) | (None, 28, 28, 256)
BatchNormalization | output:
conv3_block1_preact_relu | input: | (None, 28, 28, 256) | (None, 28, 28, 256)
Activation | output:
conv3_block1_1_conv | input: | (None, 28, 28, 256) | (None, 28, 28, 128)
Conv2D | output:
conv3_block1_1_bn | input: | (None, 28, 28, 128) | (None, 28, 28, 128)
BatchNormalization | output:
conv3_block1_0_conv | input: | (None, 28, 28, 256) | (None, 28, 28, 512)
Conv2D | output:
conv3_block1_1_relu | input: | (None, 28, 28, 128) | (None, 28, 28, 128)
Activation | output:
conv3_block1_2_pad | input: | (None, 28, 28, 128) | (None, 30, 30, 128)
ZeroPadding2D | output:
conv3_block1_2_conv | input: | (None, 30, 30, 128) | (None, 28, 28, 128)
Conv2D | output:

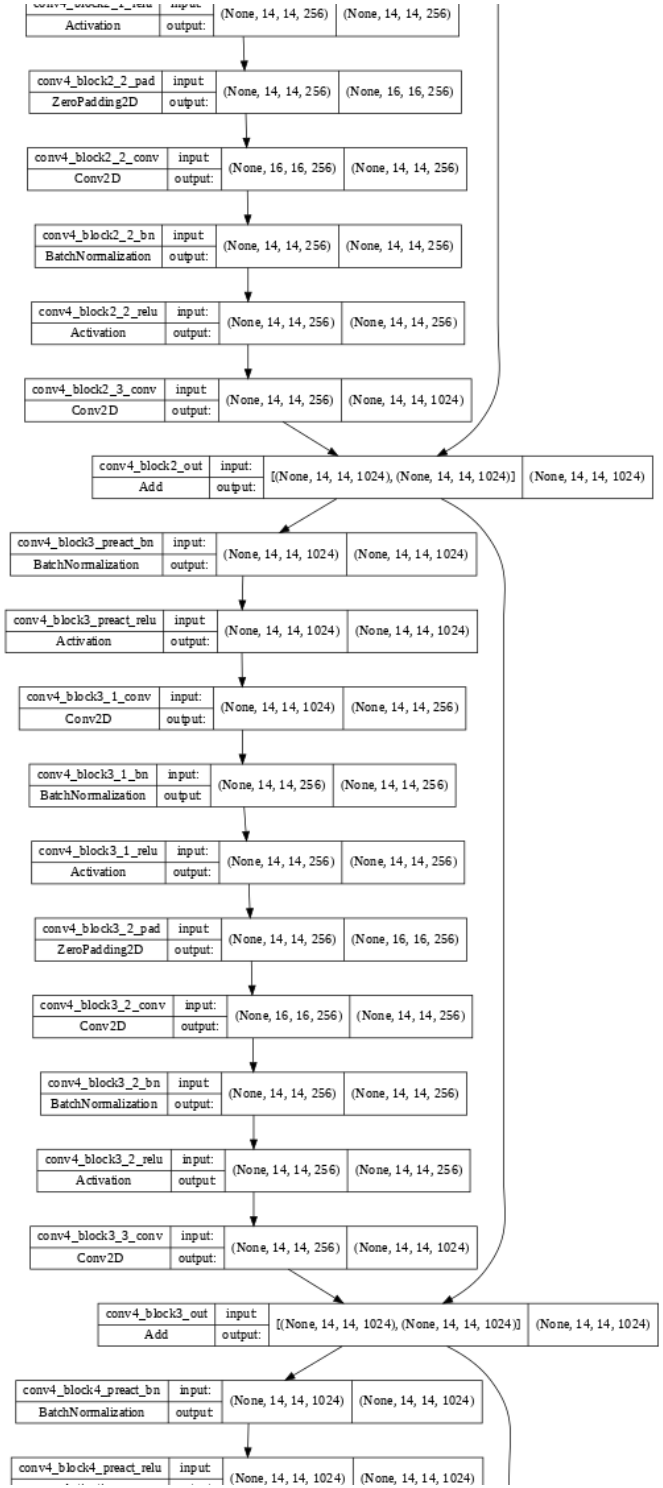

conv3_block1_2_bn | BatchNormalization | input: | output: | (None, 28, 28, 128) | (None, 28, 28, 128)
conv3_block1_2_relu | Activation | input: | output: | (None, 28, 28, 128) | (None, 28, 28, 128)
conv3_block1_3_conv | Conv2D | input: | output: | (None, 28, 28, 128) | (None, 28, 28, 512)
conv3_block1_out | Add | input: | output | [(None, 28, 28, 512), (None, 28, 28, 512)] | (None, 28, 28, 512)
conv3_block2_preact_bn | BatchNormalization | input: | output: | (None, 28, 28, 512) | (None, 28, 28, 512)
conv3_block2_preact_relu | Activation | input: | output: | (None, 28, 28, 512) | (None, 28, 28, 512)
conv3_block2_1_conv | Conv2D | input: | output: | (None, 28, 28, 512) | (None, 28, 28, 128)
conv3_block2_1_bn | BatchNormalization | input: | output: | (None, 28, 28, 128) | (None, 28, 28, 128)
conv3_block2_1_relu | Activation | input: | output: | (None, 28, 28, 128) | (None, 28, 28, 128)
conv3_block2_2_pad | ZeroPadding2D | input: | output: | (None, 28, 28, 128) | (None, 30, 30, 128)
conv3_block2_2_conv | Conv2D | input: | output: | (None, 30, 30, 128) | (None, 28, 28, 128)
conv3_block2_2_bn | BatchNormalization | input: | output: | (None, 28, 28, 128) | (None, 28, 28, 128)
conv3_block2_2_relu | Activation | input: | output: | (None, 28, 28, 128) | (None, 28, 28, 128)
conv3_block2_3_conv | Conv2D | input: | output: | (None, 28, 28, 128) | (None, 28, 28, 512)
conv3_block2_out | Add | input: | output | [(None, 28, 28, 512), (None, 28, 28, 512)] | (None, 28, 28, 512)
conv3_block3_preact_bn | BatchNormalization | input: | output: | (None, 28, 28, 512) | (None, 28, 28, 512)
conv3_block3_preact_relu | Activation | input: | output | (None, 28, 28, 512) | (None, 28, 28, 512)
conv3_block3_1_conv | Conv2D | input: | output: | (None, 28, 28, 512) | (None, 28, 28, 128)
conv3_block3_1_bn | BatchNormalization | input: | output | (None, 28, 28, 128) | (None, 28, 28, 128)
conv3_block3_1_relu | input:

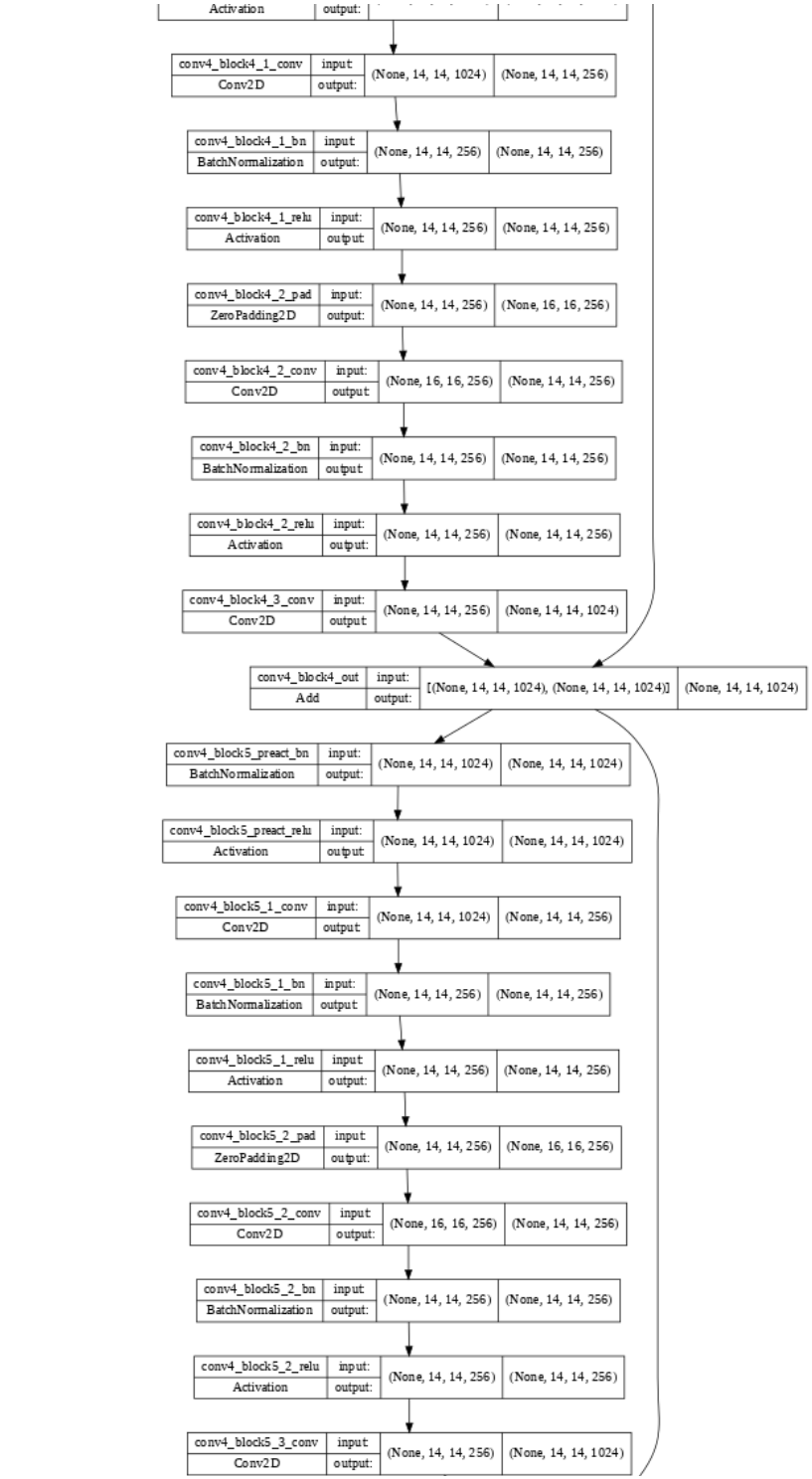

Activation | output: | (None, 28, 28, 128)
conv3_block3_2_pad | ZeroPadding2D | input: | output: | (None, 28, 28, 128) | (None, 30, 30, 128)
conv3_block3_2_conv | Conv2D | input: | output: | (None, 30, 30, 128) | (None, 28, 28, 128)
conv3_block3_2_bn | BatchNormalization | input: | output: | (None, 28, 28, 128) | (None, 28, 28, 128)
conv3_block3_2_relu | Activation | input: | output: | (None, 28, 28, 128) | (None, 28, 28, 128)
conv3_block3_3_conv | Conv2D | input: | output: | (None, 28, 28, 128) | (None, 28, 28, 512)
conv3_block3_out | Add | input: | output: | [(None, 28, 28, 512), (None, 28, 28, 512)] | (None, 28, 28, 512)
conv3_block4_preact_bn | BatchNormalization | input: | output: | (None, 28, 28, 512) | (None, 28, 28, 512)
conv3_block4_preact_relu | Activation | input: | output: | (None, 28, 28, 512) | (None, 28, 28, 512)
conv3_block4_1_conv | Conv2D | input: | output: | (None, 28, 28, 512) | (None, 28, 28, 128)
conv3_block4_1_bn | BatchNormalization | input: | output: | (None, 28, 28, 128) | (None, 28, 28, 128)
conv3_block4_1_relu | Activation | input: | output: | (None, 28, 28, 128) | (None, 28, 28, 128)
conv3_block4_2_pad | ZeroPadding2D | input: | output: | (None, 28, 28, 128) | (None, 30, 30, 128)
conv3_block4_2_conv | Conv2D | input: | output: | (None, 30, 30, 128) | (None, 28, 28, 128)
conv3_block4_2_bn | BatchNormalization | input: | output: | (None, 28, 28, 128) | (None, 28, 28, 128)
conv3_block4_2_relu | Activation | input: | output: | (None, 28, 28, 128) | (None, 28, 28, 128)
conv3_block4_3_conv | Conv2D | input: | output: | (None, 28, 28, 128) | (None, 28, 28, 512)
conv3_block4_out | Add | input: | output: | [(None, 28, 28, 512), (None, 28, 28, 512)] | (None, 28, 28, 512)
conv3_block5_preact_bn | BatchNormalization | input: | output: | (None, 28, 28, 512) | (None, 28, 28, 512)
conv3_block5_preact_relu | Activation | input: | output: | (None, 28, 28, 512) | (None, 28, 28, 512)

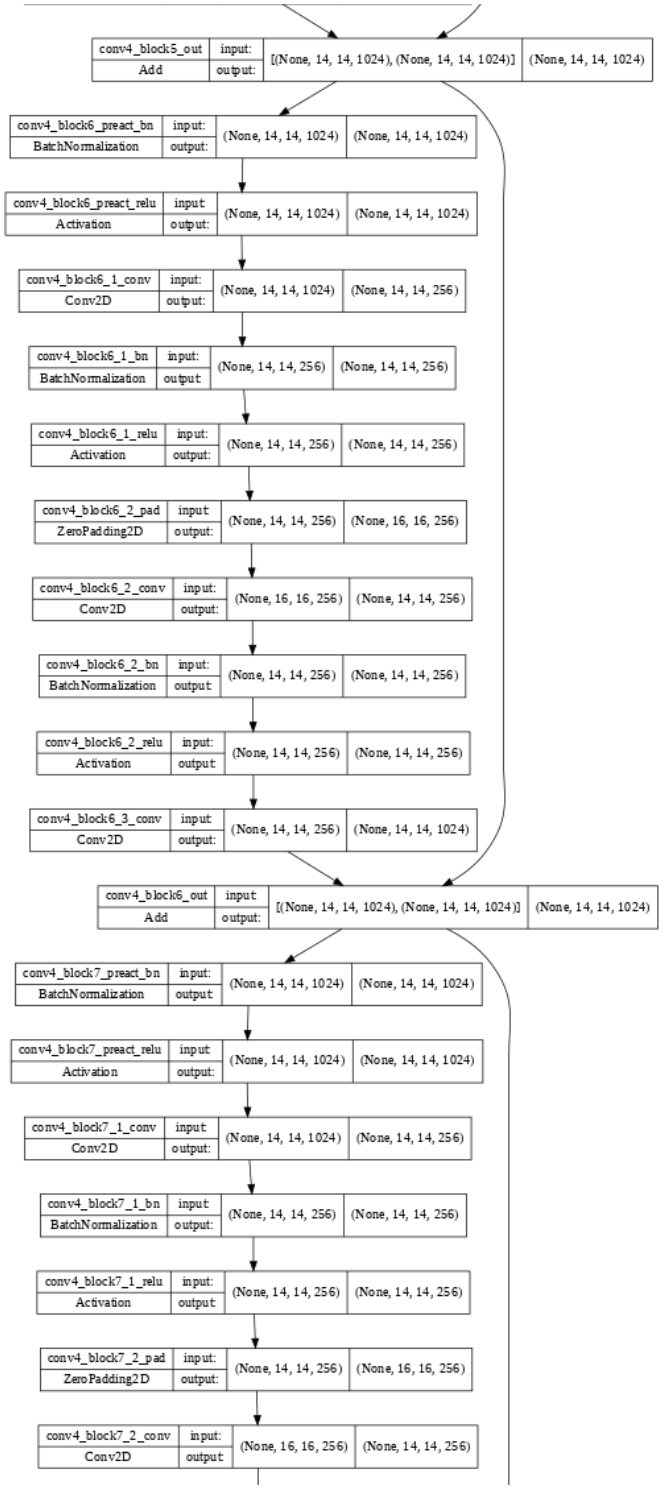

conv3_block5_1_conv | Conv2D | input: | output: | (None, 28, 28, 512) | (None, 28, 28, 128)
conv3_block5_1_bn | BatchNormalization | input: | output: | (None, 28, 28, 128) | (None, 28, 28, 128)
conv3_block5_1_relu | Activation | input: | output: | (None, 28, 28, 128) | (None, 28, 28, 128)
conv3_block5_2_pad | ZeroPadding2D | input: | output: | (None, 28, 28, 128) | (None, 30, 30, 128)
conv3_block5_2_conv | Conv2D | input: | output: | (None, 30, 30, 128) | (None, 28, 28, 128)
conv3_block5_2_bn | BatchNormalization | input: | output: | (None, 28, 28, 128) | (None, 28, 28, 128)
conv3_block5_2_relu | Activation | input: | output: | (None, 28, 28, 128) | (None, 28, 28, 128)
conv3_block5_3_conv | Conv2D | input: | output: | (None, 28, 28, 128) | (None, 28, 28, 512)
conv3_block5_out | Add | input: | output: | [(None, 28, 28, 512), (None, 28, 28, 512)] | (None, 28, 28, 512)
conv3_block6_preact_bn | BatchNormalization | input: | output: | (None, 28, 28, 512) | (None, 28, 28, 512)
conv3_block6_preact_relu | Activation | input: | output: | (None, 28, 28, 512) | (None, 28, 28, 512)
conv3_block6_1_conv | Conv2D | input: | output: | (None, 28, 28, 512) | (None, 28, 28, 128)
conv3_block6_1_bn | BatchNormalization | input: | output: | (None, 28, 28, 128) | (None, 28, 28, 128)
conv3_block6_1_relu | Activation | input: | output: | (None, 28, 28, 128) | (None, 28, 28, 128)
conv3_block6_2_pad | ZeroPadding2D | input: | output: | (None, 28, 28, 128) | (None, 30, 30, 128)
conv3_block6_2_conv | Conv2D | input: | output: | (None, 30, 30, 128) | (None, 28, 28, 128)
conv3_block6_2_bn | BatchNormalization | input: | output: | (None, 28, 28, 128) | (None, 28, 28, 128)
conv3_block6_2_relu | Activation | input: | output: | (None, 28, 28, 128) | (None, 28, 28, 128)
conv3_block6_3_conv | Conv2D | input: | output: | (None, 28, 28, 128) | (None, 28, 28, 512)

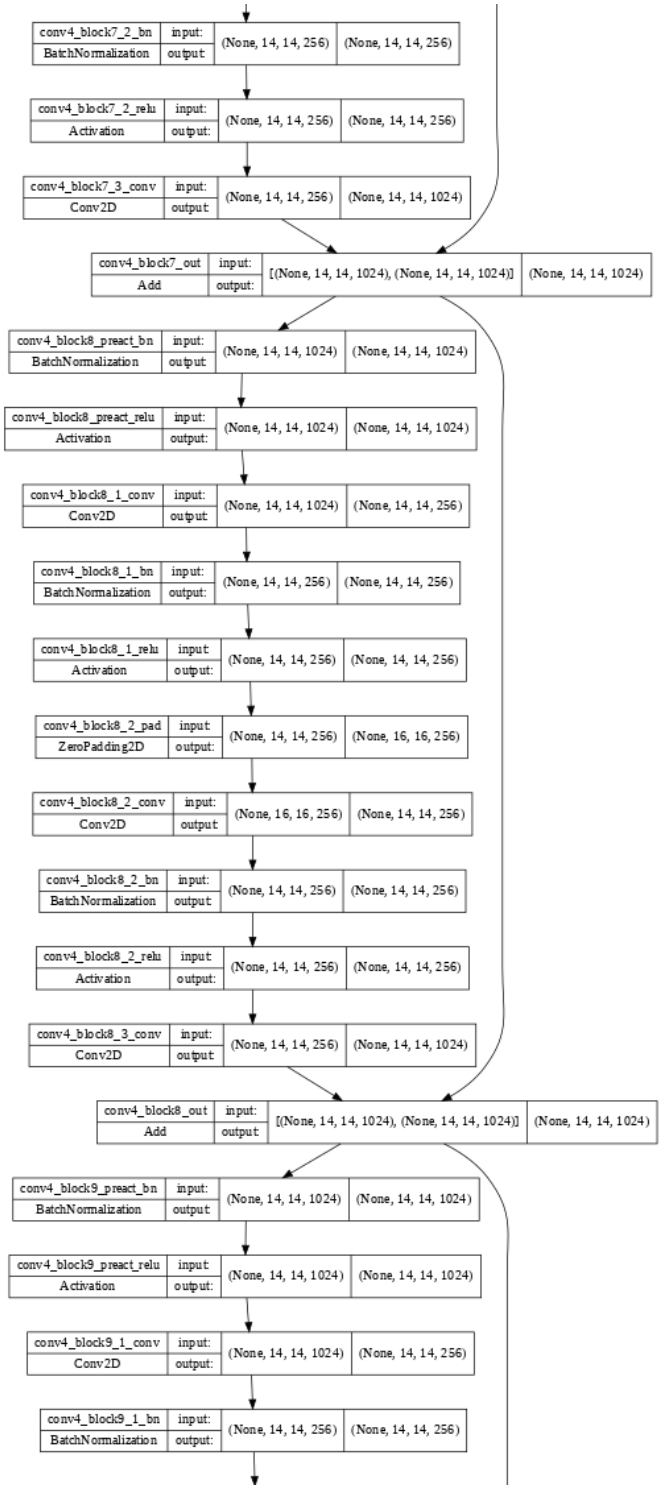

conv3_block6_out | Add | input: | output: | [(None, 28, 28, 512), (None, 28, 28, 512)] | (None, 28, 28, 512)
conv3_block7_preact_bn | BatchNormalization | input: | output: | (None, 28, 28, 512) | (None, 28, 28, 512)
conv3_block7_preact_relu | Activation | input: | output: | (None, 28, 28, 512) | (None, 28, 28, 512)
conv3_block7_1_conv | Conv2D | input: | output: | (None, 28, 28, 512) | (None, 28, 28, 128)
conv3_block7_1_bn | BatchNormalization | input: | output: | (None, 28, 28, 128) | (None, 28, 28, 128)
conv3_block7_1_relu | Activation | input: | output: | (None, 28, 28, 128) | (None, 28, 28, 128)
conv3_block7_2_pad | ZeroPadding2D | input: | output: | (None, 28, 28, 128) | (None, 30, 30, 128)
conv3_block7_2_conv | Conv2D | input: | output: | (None, 30, 30, 128) | (None, 28, 28, 128)
conv3_block7_2_bn | BatchNormalization | input: | output: | (None, 28, 28, 128) | (None, 28, 28, 128)
conv3_block7_2_relu | Activation | input: | output: | (None, 28, 28, 128) | (None, 28, 28, 128)
conv3_block7_3_conv | Conv2D | input: | output: | (None, 28, 28, 128) | (None, 28, 28, 512)
conv3_block7_out | Add | input: | output: | [(None, 28, 28, 512), (None, 28, 28, 512)] | (None, 28, 28, 512)
conv3_block8_preact_bn | BatchNormalization | input: | output: | (None, 28, 28, 512) | (None, 28, 28, 512)
conv3_block8_preact_relu | Activation | input: | output: | (None, 28, 28, 512) | (None, 28, 28, 512)
max_pooling2d_1 | MaxPooling2D | input: | output: | (None, 28, 28, 512) | (None, 14, 14, 512)
conv3_block8_1_conv | Conv2D | input: | output: | (None, 28, 28, 512) | (None, 28, 28, 128)
conv3_block8_1_bn | BatchNormalization | input: | output: | (None, 28, 28, 128) | (None, 28, 28, 128)
conv3_block8_1_relu | Activation | input: | output: | (None, 28, 28, 128) | (None, 28, 28, 128)
conv3_block8_2_pad | ZeroPadding2D | input: | output: | (None, 28, 28, 128) | (None, 30, 30, 128)
conv3_block8_2_conv | Conv2D | input: | output | (None, 30, 30, 128) | (None, 14, 14, 128)

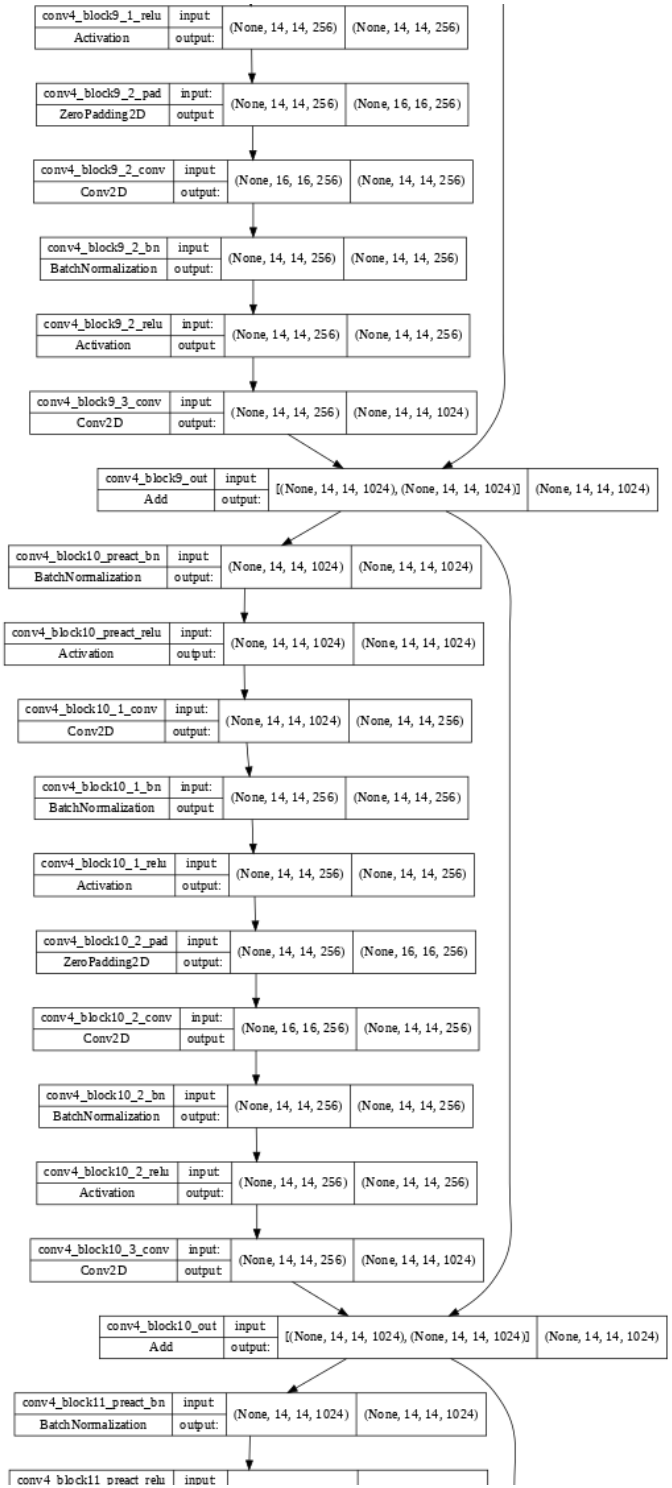

conv3_block8_2_bn | BatchNormalization | input: | output: | (None, 14, 14, 128) | (None, 14, 14, 128)
conv3_block8_2_relu | Activation | input: | output: | (None, 14, 14, 128) | (None, 14, 14, 128)
conv3_block8_3_conv | Conv2D | input: | output: | (None, 14, 14, 128) | (None, 14, 14, 512)
conv3_block8_out | Add | input: | output: | [(None, 14, 14, 512), (None, 14, 14, 512)] | (None, 14, 14, 512)
conv4_block1_preact_bn | BatchNormalization | input: | output: | (None, 14, 14, 512) | (None, 14, 14, 512)
conv4_block1_preact_relu | Activation | input: | output: | (None, 14, 14, 512) | (None, 14, 14, 512)
conv4_block1_1_conv | Conv2D | input: | output: | (None, 14, 14, 512) | (None, 14, 14, 256)
conv4_block1_1_bn | BatchNormalization | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block1_0_conv | Conv2D | input | output: | (None, 14, 14, 512) | (None, 14, 14, 1024)
conv4_block1_1_relu | Activation | input: | output | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block1_2_pad | ZeroPadding2D | input | output: | (None, 14, 14, 256) | (None, 16, 16, 256)
conv4_block1_2_conv | Conv2D | input: | output | (None, 16, 16, 256) | (None, 14, 14, 256)
conv4_block1_2_bn | BatchNormalization | input: | output | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block1_2_relu | Activation | input | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block1_3_conv | Conv2D | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 1024)
conv4_block1_out | Add | input: | output | [(None, 14, 14, 1024), (None, 14, 14, 1024)] | (None, 14, 14, 1024)
conv4_block2_preact_bn | BatchNormalization | input | output: | (None, 14, 14, 1024) | (None, 14, 14, 1024)
conv4_block2_preact_relu | Activation | input: | output | (None, 14, 14, 1024) | (None, 14, 14, 1024)
conv4_block2_1_conv | Conv2D | input: | output | (None, 14, 14, 1024) | (None, 14, 14, 256)
conv4_block2_1_bn | BatchNormalization | input: | output | (None, 14, 14, 256) | (None, 14, 14, 256)

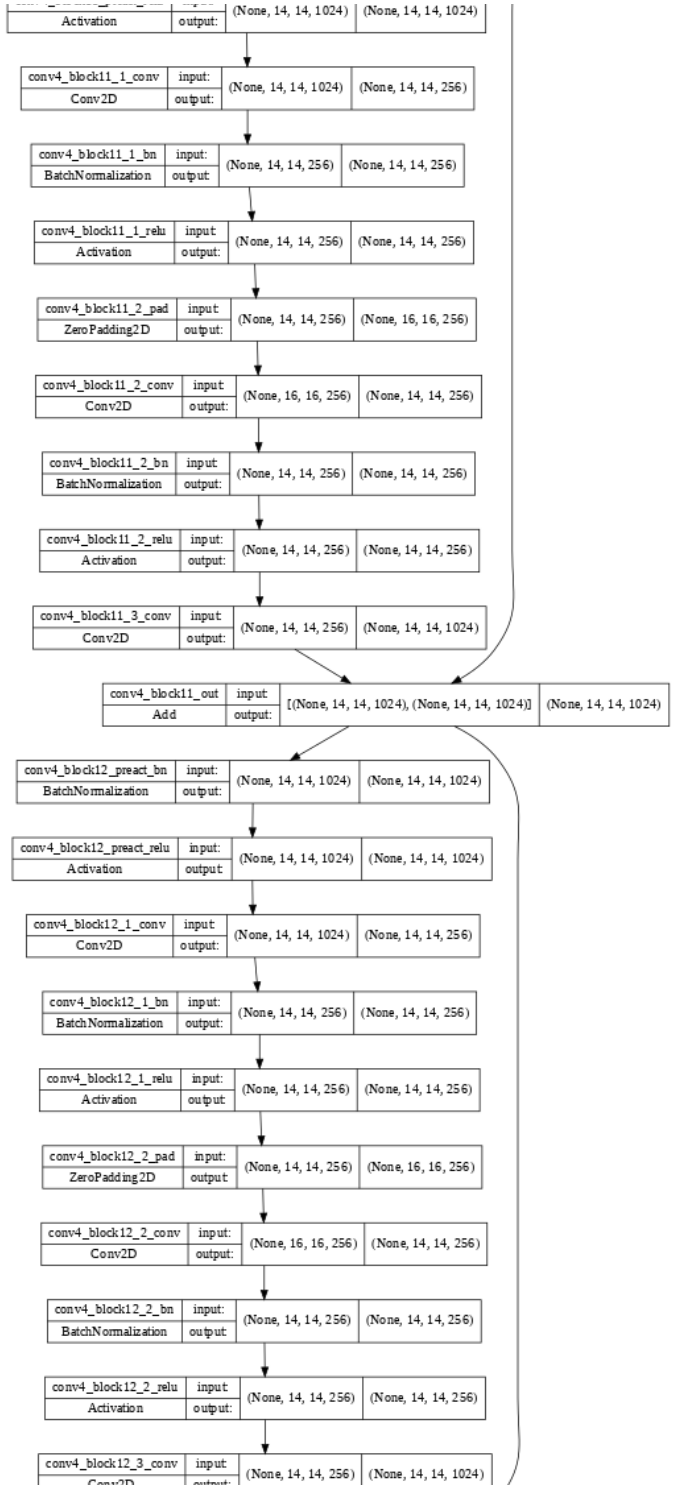

Activation | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block2_2_pad | ZeroPadding2D | input: | output: | (None, 14, 14, 256) | (None, 16, 16, 256)
conv4_block2_2_conv | Conv2D | input: | output: | (None, 16, 16, 256) | (None, 14, 14, 256)
conv4_block2_2_bn | BatchNormalization | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block2_2_relu | Activation | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block2_3_conv | Conv2D | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 1024)
conv4_block2_out | Add | input: | output: | [(None, 14, 14, 1024), (None, 14, 14, 1024)] | (None, 14, 14, 1024)
conv4_block3_preact_bn | BatchNormalization | input: | output: | (None, 14, 14, 1024) | (None, 14, 14, 1024)
conv4_block3_preact_relu | Activation | input: | output: | (None, 14, 14, 1024) | (None, 14, 14, 1024)
conv4_block3_1_conv | Conv2D | input: | output: | (None, 14, 14, 1024) | (None, 14, 14, 256)
conv4_block3_1_bn | BatchNormalization | input: | output | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block3_1_relu | Activation | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block3_2_pad | ZeroPadding2D | input | output: | (None, 14, 14, 256) | (None, 16, 16, 256)
conv4_block3_2_conv | Conv2D | input: | output: | (None, 16, 16, 256) | (None, 14, 14, 256)
conv4_block3_2_bn | BatchNormalization | input | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block3_2_relu | Activation | input: | output | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block3_3_conv | Conv2D | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 1024)
conv4_block3_out | Add | input | output: | [(None, 14, 14, 1024), (None, 14, 14, 1024)] | (None, 14, 14, 1024)
conv4_block4_preact_bn | BatchNormalization | input: | output | (None, 14, 14, 1024) | (None, 14, 14, 1024)
conv4_block4_preact_relu | input: | (None, 14, 14, 1024) | (None, 14, 14, 1024)

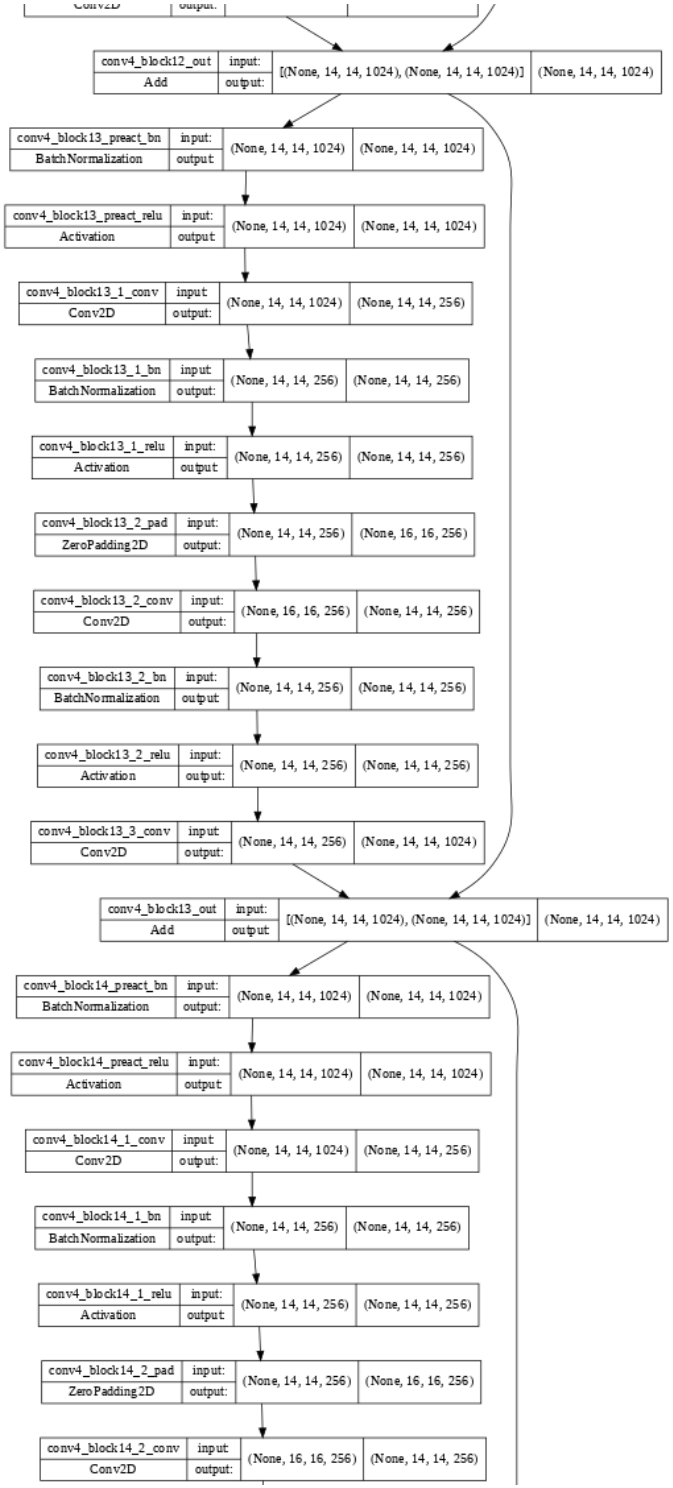

Activation | output: |
conv4_block4_1_conv | Conv2D | input: | output: | (None, 14, 14, 1024) | (None, 14, 14, 256)
conv4_block4_1_bn | BatchNormalization | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block4_1_relu | Activation | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block4_2_pad | ZeroPadding2D | input: | output: | (None, 14, 14, 256) | (None, 16, 16, 256)
conv4_block4_2_conv | Conv2D | input: | output: | (None, 16, 16, 256) | (None, 14, 14, 256)
conv4_block4_2_bn | BatchNormalization | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block4_2_relu | Activation | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block4_3_conv | Conv2D | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 1024)
conv4_block4_out | Add | input: | output: | [(None, 14, 14, 1024), (None, 14, 14, 1024)] | (None, 14, 14, 1024)
conv4_block5_preact_bn | BatchNormalization | input: | output: | (None, 14, 14, 1024) | (None, 14, 14, 1024)
conv4_block5_preact_relu | Activation | input: | output: | (None, 14, 14, 1024) | (None, 14, 14, 1024)
conv4_block5_1_conv | Conv2D | input: | output: | (None, 14, 14, 1024) | (None, 14, 14, 256)
conv4_block5_1_bn | BatchNormalization | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block5_1_relu | Activation | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block5_2_pad | ZeroPadding2D | input: | output: | (None, 14, 14, 256) | (None, 16, 16, 256)
conv4_block5_2_conv | Conv2D | input: | output: | (None, 16, 16, 256) | (None, 14, 14, 256)
conv4_block5_2_bn | BatchNormalization | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block5_2_relu | Activation | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block5_3_conv | Conv2D | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 1024)

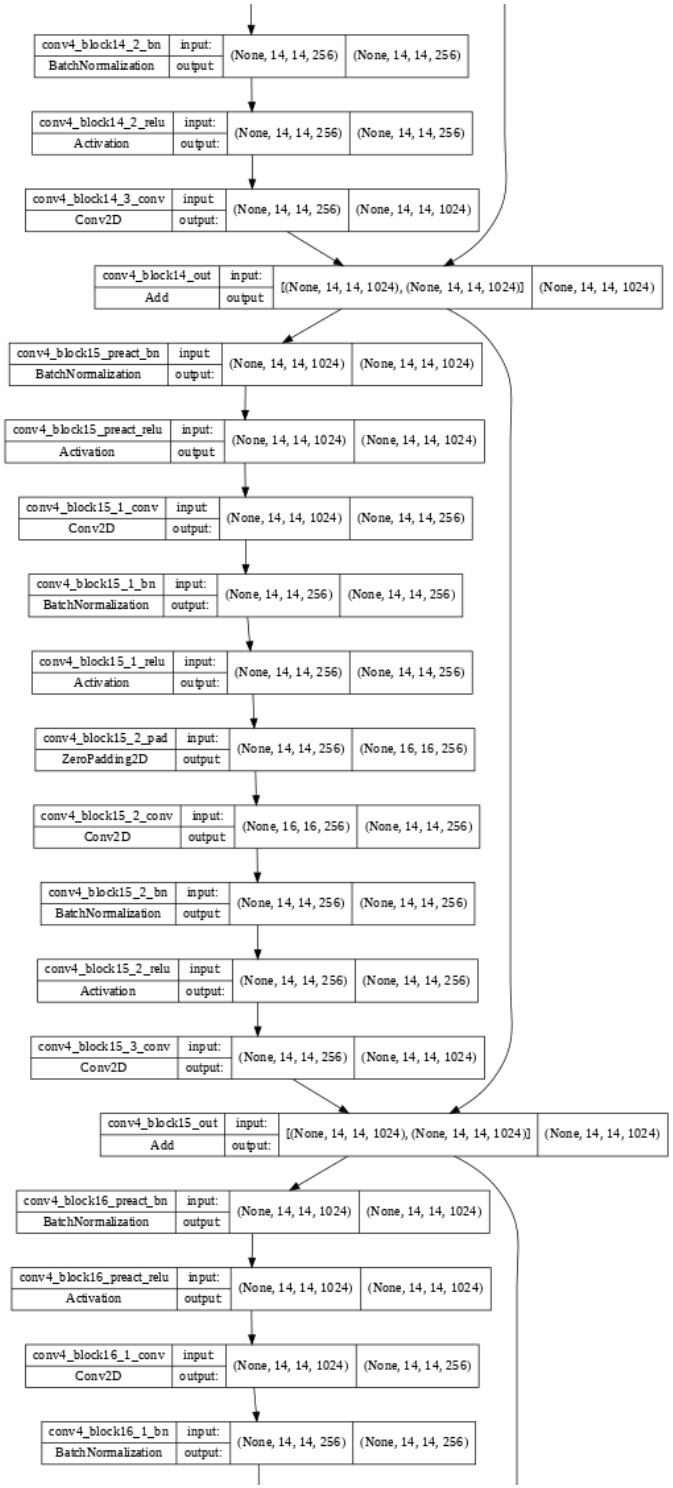

conv4_block5_out | Add | input: | output: | [(None, 14, 14, 1024), (None, 14, 14, 1024)] | (None, 14, 14, 1024)
conv4_block6_preact_bn | BatchNormalization | input: | output: | (None, 14, 14, 1024) | (None, 14, 14, 1024)
conv4_block6_preact_relu | Activation | input | output: | (None, 14, 14, 1024) | (None, 14, 14, 1024)
conv4_block6_1_conv | Conv2D | input: | output: | (None, 14, 14, 1024) | (None, 14, 14, 256)
conv4_block6_1_bn | BatchNormalization | input: | output | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block6_1_relu | Activation | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block6_2_pad | ZeroPadding2D | input | output: | (None, 14, 14, 256) | (None, 16, 16, 256)
conv4_block6_2_conv | Conv2D | input: | output: | (None, 16, 16, 256) | (None, 14, 14, 256)
conv4_block6_2_bn | BatchNormalization | input: | output | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block6_2_relu | Activation | input: | output | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block6_3_conv | Conv2D | input | output: | (None, 14, 14, 256) | (None, 14, 14, 1024)
conv4_block6_out | Add | input | output: | [(None, 14, 14, 1024), (None, 14, 14, 1024)] | (None, 14, 14, 1024)
conv4_block7_preact_bn | BatchNormalization | input: | output | (None, 14, 14, 1024) | (None, 14, 14, 1024)
conv4_block7_preact_relu | Activation | input | output: | (None, 14, 14, 1024) | (None, 14, 14, 1024)
conv4_block7_1_conv | Conv2D | input | output: | (None, 14, 14, 1024) | (None, 14, 14, 256)
conv4_block7_1_bn | BatchNormalization | input | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block7_1_relu | Activation | input: | output | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block7_2_pad | ZeroPadding2D | input: | output: | (None, 14, 14, 256) | (None, 16, 16, 256)
conv4_block7_2_conv | Conv2D | input: | output | (None, 16, 16, 256) | (None, 14, 14, 256)

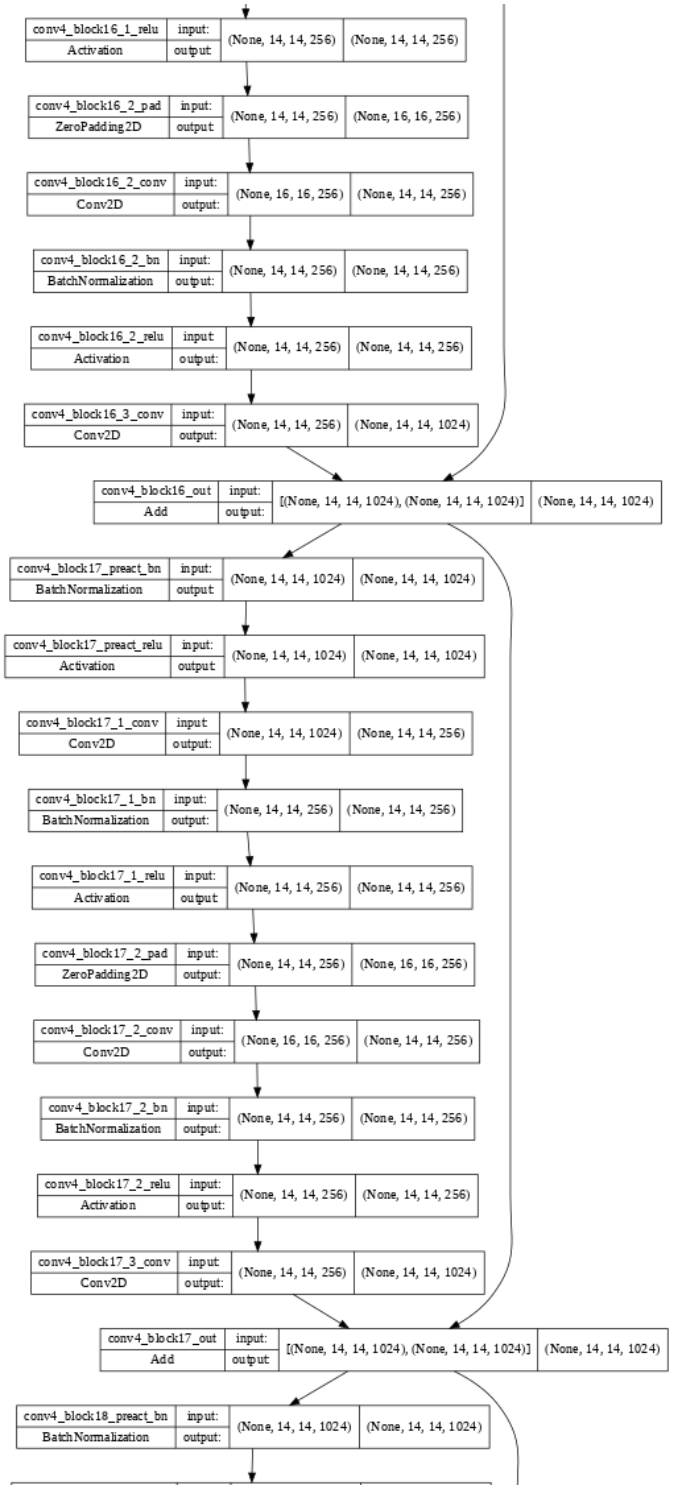

conv4_block7_2_bn | BatchNormalization | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block7_2_relu | Activation | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block7_3_conv | Conv2D | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 1024)
conv4_block7_out | Add | input: | output: | [(None, 14, 14, 1024), (None, 14, 14, 1024)] | (None, 14, 14, 1024)
conv4_block8_preact_bn | BatchNormalization | input: | output: | (None, 14, 14, 1024) | (None, 14, 14, 1024)
conv4_block8_preact_relu | Activation | input: | output: | (None, 14, 14, 1024) | (None, 14, 14, 1024)
conv4_block8_1_conv | Conv2D | input: | output: | (None, 14, 14, 1024) | (None, 14, 14, 256)
conv4_block8_1_bn | BatchNormalization | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block8_1_relu | Activation | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block8_2_pad | ZeroPadding2D | input: | output: | (None, 14, 14, 256) | (None, 16, 16, 256)
conv4_block8_2_conv | Conv2D | input: | output: | (None, 16, 16, 256) | (None, 14, 14, 256)
conv4_block8_2_bn | BatchNormalization | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block8_2_relu | Activation | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block8_3_conv | Conv2D | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 1024)
conv4_block8_out | Add | input: | output: | [(None, 14, 14, 1024), (None, 14, 14, 1024)] | (None, 14, 14, 1024)
conv4_block9_preact_bn | BatchNormalization | input: | output: | (None, 14, 14, 1024) | (None, 14, 14, 1024)
conv4_block9_preact_relu | Activation | input: | output: | (None, 14, 14, 1024) | (None, 14, 14, 1024)
conv4_block9_1_conv | Conv2D | input: | output: | (None, 14, 14, 1024) | (None, 14, 14, 256)
conv4_block9_1_bn | BatchNormalization | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)

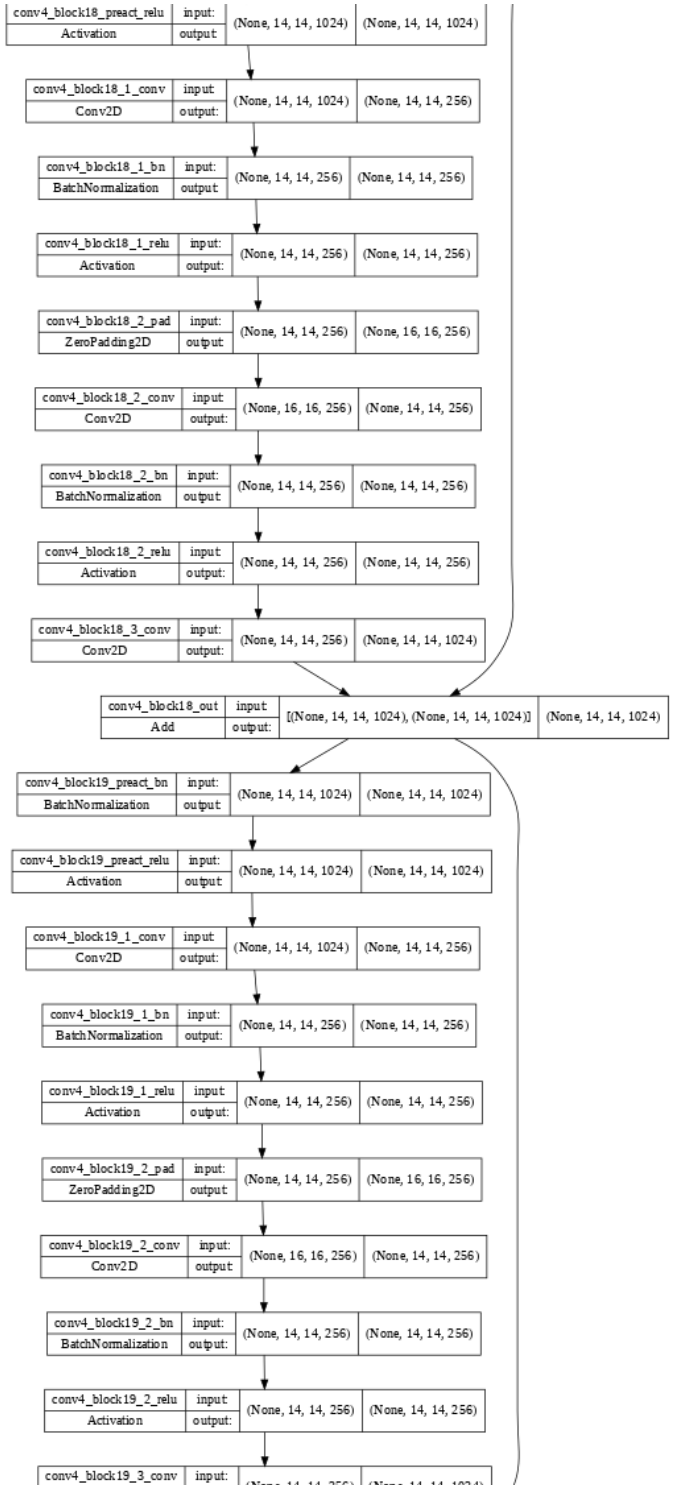

conv4_block9_1_relu | Activation | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block9_2_pad | ZeroPadding2D | input: | output: | (None, 14, 14, 256) | (None, 16, 16, 256)
conv4_block9_2_conv | Conv2D | input: | output: | (None, 16, 16, 256) | (None, 14, 14, 256)
conv4_block9_2_bn | BatchNormalization | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block9_2_relu | Activation | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block9_3_conv | Conv2D | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 1024)
conv4_block9_out | Add | input: | output: | [(None, 14, 14, 1024), (None, 14, 14, 1024)] | (None, 14, 14, 1024)
conv4_block10_preact_bn | BatchNormalization | input: | output: | (None, 14, 14, 1024) | (None, 14, 14, 1024)
conv4_block10_preact_relu | Activation | input: | output: | (None, 14, 14, 1024) | (None, 14, 14, 1024)
conv4_block10_1_conv | Conv2D | input: | output: | (None, 14, 14, 1024) | (None, 14, 14, 256)
conv4_block10_1_bn | BatchNormalization | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block10_1_relu | Activation | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block10_2_pad | ZeroPadding2D | input: | output: | (None, 14, 14, 256) | (None, 16, 16, 256)
conv4_block10_2_conv | Conv2D | input: | output: | (None, 16, 16, 256) | (None, 14, 14, 256)
conv4_block10_2_bn | BatchNormalization | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block10_2_relu | Activation | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block10_3_conv | Conv2D | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 1024)
conv4_block10_out | Add | input: | output: | [(None, 14, 14, 1024), (None, 14, 14, 1024)] | (None, 14, 14, 1024)
conv4_block11_preact_bn | BatchNormalization | input: | output: | (None, 14, 14, 1024) | (None, 14, 14, 1024)

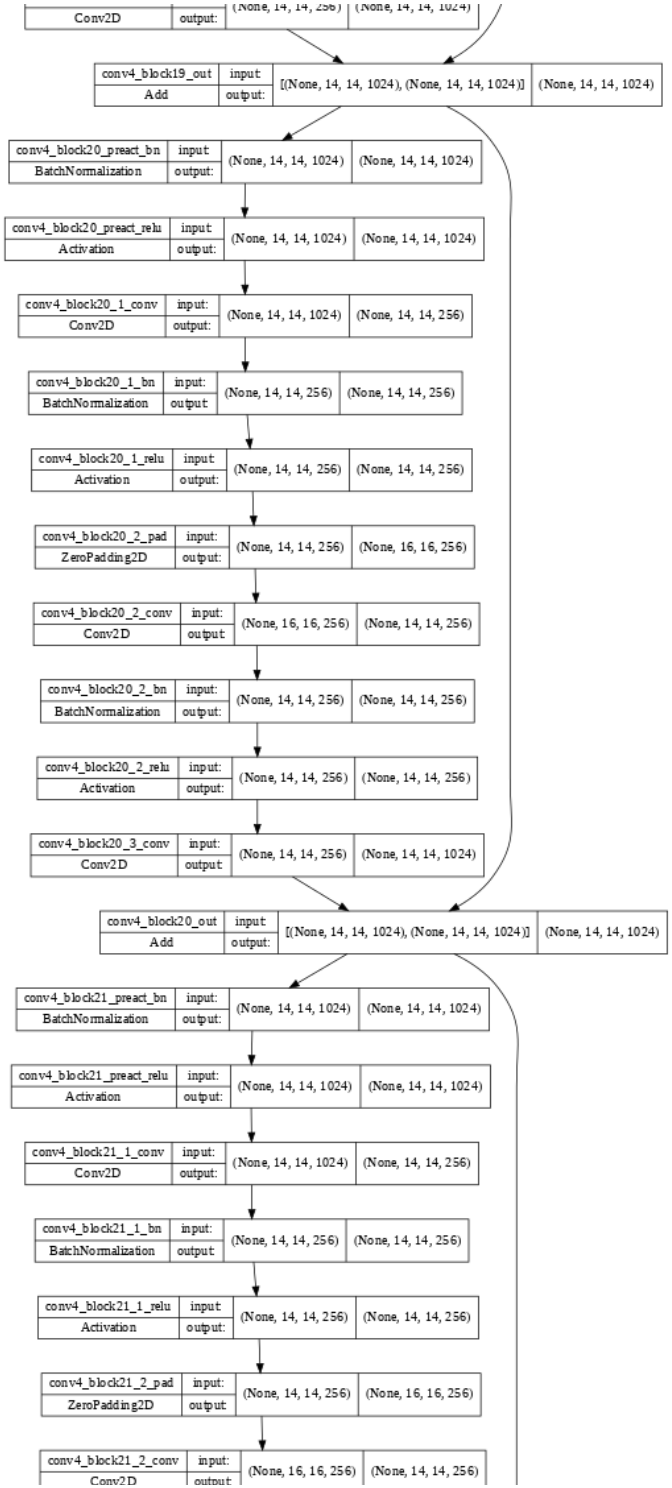

Activation | output: | (None, 14, 14, 1024) | (None, 14, 14, 1024)
conv4_block11_1_conv | Conv2D | input: | output: | (None, 14, 14, 1024) | (None, 14, 14, 256)
conv4_block11_1_bn | BatchNormalization | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block11_1_relu | Activation | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block11_2_pad | ZeroPadding2D | input: | output: | (None, 14, 14, 256) | (None, 16, 16, 256)
conv4_block11_2_conv | Conv2D | input: | output: | (None, 16, 16, 256) | (None, 14, 14, 256)
conv4_block11_2_bn | BatchNormalization | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block11_2_relu | Activation | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block11_3_conv | Conv2D | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 1024)
conv4_block11_out | Add | input: | output: | [(None, 14, 14, 1024), (None, 14, 14, 1024)] | (None, 14, 14, 1024)
conv4_block12_preact_bn | BatchNormalization | input: | output: | (None, 14, 14, 1024) | (None, 14, 14, 1024)
conv4_block12_preact_relu | Activation | input: | output: | (None, 14, 14, 1024) | (None, 14, 14, 1024)
conv4_block12_1_conv | Conv2D | input: | output: | (None, 14, 14, 1024) | (None, 14, 14, 256)
conv4_block12_1_bn | BatchNormalization | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block12_1_relu | Activation | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block12_2_pad | ZeroPadding2D | input: | output: | (None, 14, 14, 256) | (None, 16, 16, 256)
conv4_block12_2_conv | Conv2D | input: | output: | (None, 16, 16, 256) | (None, 14, 14, 256)
conv4_block12_2_bn | BatchNormalization | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block12_2_relu | Activation | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block12_3_conv | input: | (None, 14, 14, 256) | (None, 14, 14, 1024)

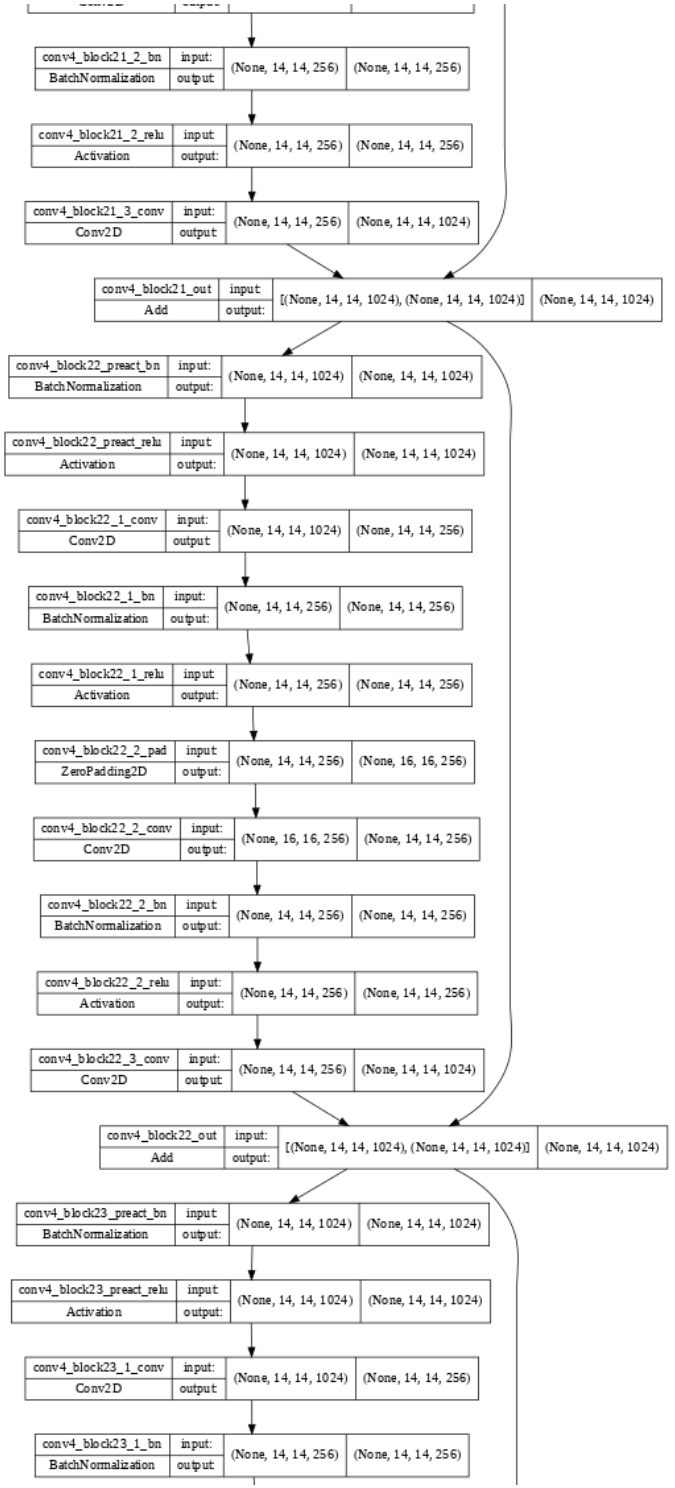

conv4_block12_out | input: | [(None, 14, 14, 1024), (None, 14, 14, 1024)] | (None, 14, 14, 1024)
Add | output:
conv4_block13_preact_bn | input: | (None, 14, 14, 1024) | (None, 14, 14, 1024)
BatchNormalization | output:
conv4_block13_preact_relu | input: | (None, 14, 14, 1024) | (None, 14, 14, 1024)
Activation | output:
conv4_block13_1_conv | input: | (None, 14, 14, 1024) | (None, 14, 14, 256)
Conv2D | output:
conv4_block13_1_bn | input: | (None, 14, 14, 256) | (None, 14, 14, 256)
BatchNormalization | output:
conv4_block13_1_relu | input: | (None, 14, 14, 256) | (None, 14, 14, 256)
Activation | output:
conv4_block13_2_pad | input: | (None, 14, 14, 256) | (None, 16, 16, 256)
ZeroPadding2D | output:
conv4_block13_2_conv | input: | (None, 16, 16, 256) | (None, 14, 14, 256)
Conv2D | output:
conv4_block13_2_bn | input: | (None, 14, 14, 256) | (None, 14, 14, 256)
BatchNormalization | output:
conv4_block13_2_relu | input: | (None, 14, 14, 256) | (None, 14, 14, 256)
Activation | output:
conv4_block13_3_conv | input: | (None, 14, 14, 256) | (None, 14, 14, 1024)
Conv2D | output:
conv4_block13_out | input: | [(None, 14, 14, 1024), (None, 14, 14, 1024)] | (None, 14, 14, 1024)
Add | output:
conv4_block14_preact_bn | input: | (None, 14, 14, 1024) | (None, 14, 14, 1024)
BatchNormalization | output:
conv4_block14_preact_relu | input: | (None, 14, 14, 1024) | (None, 14, 14, 1024)
Activation | output:
conv4_block14_1_conv | input: | (None, 14, 14, 1024) | (None, 14, 14, 256)
Conv2D | output:
conv4_block14_1_bn | input: | (None, 14, 14, 256) | (None, 14, 14, 256)
BatchNormalization | output:
conv4_block14_1_relu | input: | (None, 14, 14, 256) | (None, 14, 14, 256)
Activation | output:
conv4_block14_2_pad | input: | (None, 14, 14, 256) | (None, 16, 16, 256)
ZeroPadding2D | output:
conv4_block14_2_conv | input: | (None, 16, 16, 256) | (None, 14, 14, 256)
Conv2D | output:

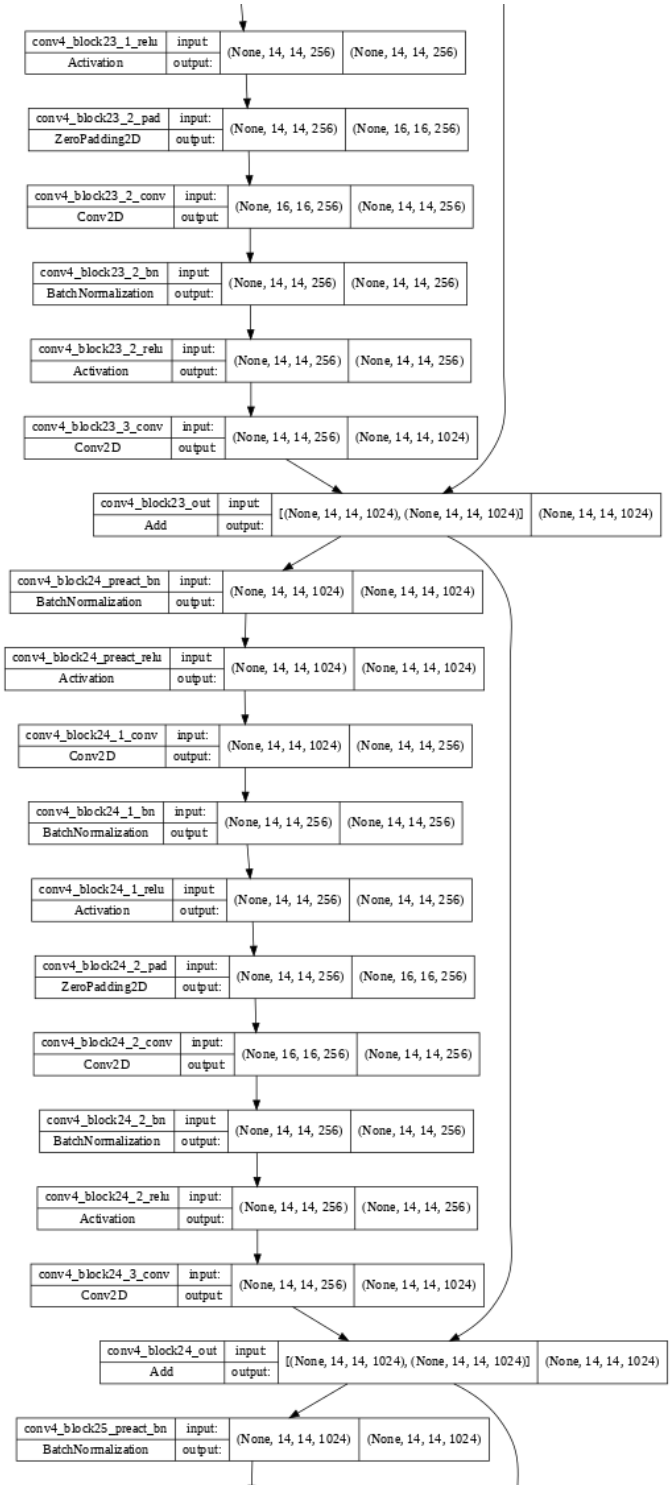

conv4_block14_2_bn | input: | (None, 14, 14, 256) | (None, 14, 14, 256)
BatchNormalization | output:
conv4_block14_2_relu | input: | (None, 14, 14, 256) | (None, 14, 14, 256)
Activation | output:
conv4_block14_3_conv | input: | (None, 14, 14, 256) | (None, 14, 14, 1024)
Conv2D | output:
conv4_block14_out | input: | [(None, 14, 14, 1024), (None, 14, 14, 1024)] | (None, 14, 14, 1024)
Add | output:
conv4_block15_preact_bn | input: | (None, 14, 14, 1024) | (None, 14, 14, 1024)
BatchNormalization | output:
conv4_block15_preact_relu | input: | (None, 14, 14, 1024) | (None, 14, 14, 1024)
Activation | output:
conv4_block15_1_conv | input: | (None, 14, 14, 1024) | (None, 14, 14, 256)
Conv2D | output:
conv4_block15_1_bn | input: | (None, 14, 14, 256) | (None, 14, 14, 256)
BatchNormalization | output:
conv4_block15_1_relu | input: | (None, 14, 14, 256) | (None, 14, 14, 256)
Activation | output:
conv4_block15_2_pad | input: | (None, 14, 14, 256) | (None, 16, 16, 256)
ZeroPadding2D | output:
conv4_block15_2_conv | input: | (None, 16, 16, 256) | (None, 14, 14, 256)
Conv2D | output:
conv4_block15_2_bn | input: | (None, 14, 14, 256) | (None, 14, 14, 256)
BatchNormalization | output:
conv4_block15_2_relu | input: | (None, 14, 14, 256) | (None, 14, 14, 256)
Activation | output:
conv4_block15_3_conv | input: | (None, 14, 14, 256) | (None, 14, 14, 1024)
Conv2D | output:
conv4_block15_out | input: | [(None, 14, 14, 1024), (None, 14, 14, 1024)] | (None, 14, 14, 1024)
Add | output:
conv4_block16_preact_bn | input: | (None, 14, 14, 1024) | (None, 14, 14, 1024)
BatchNormalization | output:
conv4_block16_preact_relu | input: | (None, 14, 14, 1024) | (None, 14, 14, 1024)
Activation | output:
conv4_block16_1_conv | input: | (None, 14, 14, 1024) | (None, 14, 14, 256)
Conv2D | output:
conv4_block16_1_bn | input: | (None, 14, 14, 256) | (None, 14, 14, 256)
BatchNormalization | output:

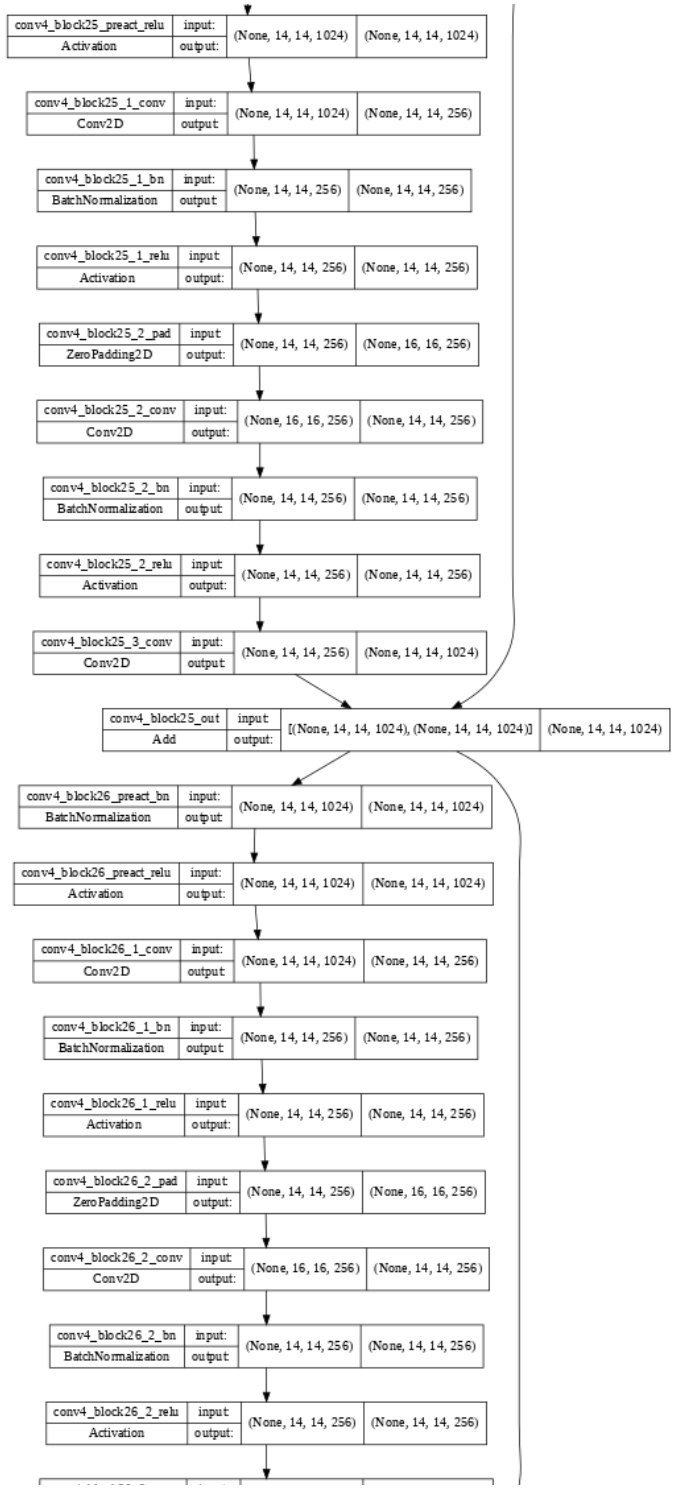

conv4_block16_1_relu | Activation | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block16_2_pad | ZeroPadding2D | input: | output: | (None, 14, 14, 256) | (None, 16, 16, 256)
conv4_block16_2_conv | Conv2D | input: | output: | (None, 16, 16, 256) | (None, 14, 14, 256)
conv4_block16_2_bn | BatchNormalization | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block16_2_relu | Activation | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block16_3_conv | Conv2D | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 1024)
conv4_block16_out | Add | input: | output: | [(None, 14, 14, 1024), (None, 14, 14, 1024)] | (None, 14, 14, 1024)
conv4_block17_preact_bn | BatchNormalization | input: | output | (None, 14, 14, 1024) | (None, 14, 14, 1024)
conv4_block17_preact_relu | Activation | input: | output | (None, 14, 14, 1024) | (None, 14, 14, 1024)
conv4_block17_1_conv | Conv2D | input: | output: | (None, 14, 14, 1024) | (None, 14, 14, 256)
conv4_block17_1_bn | BatchNormalization | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block17_1_relu | Activation | input: | output | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block17_2_pad | ZeroPadding2D | input: | output: | (None, 14, 14, 256) | (None, 16, 16, 256)
conv4_block17_2_conv | Conv2D | input: | output: | (None, 16, 16, 256) | (None, 14, 14, 256)
conv4_block17_2_bn | BatchNormalization | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block17_2_relu | Activation | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block17_3_conv | Conv2D | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 1024)
conv4_block17_out | Add | input: | output: | [(None, 14, 14, 1024), (None, 14, 14, 1024)] | (None, 14, 14, 1024)
conv4_block18_preact_bn | BatchNormalization | input: | output: | (None, 14, 14, 1024) | (None, 14, 14, 1024)

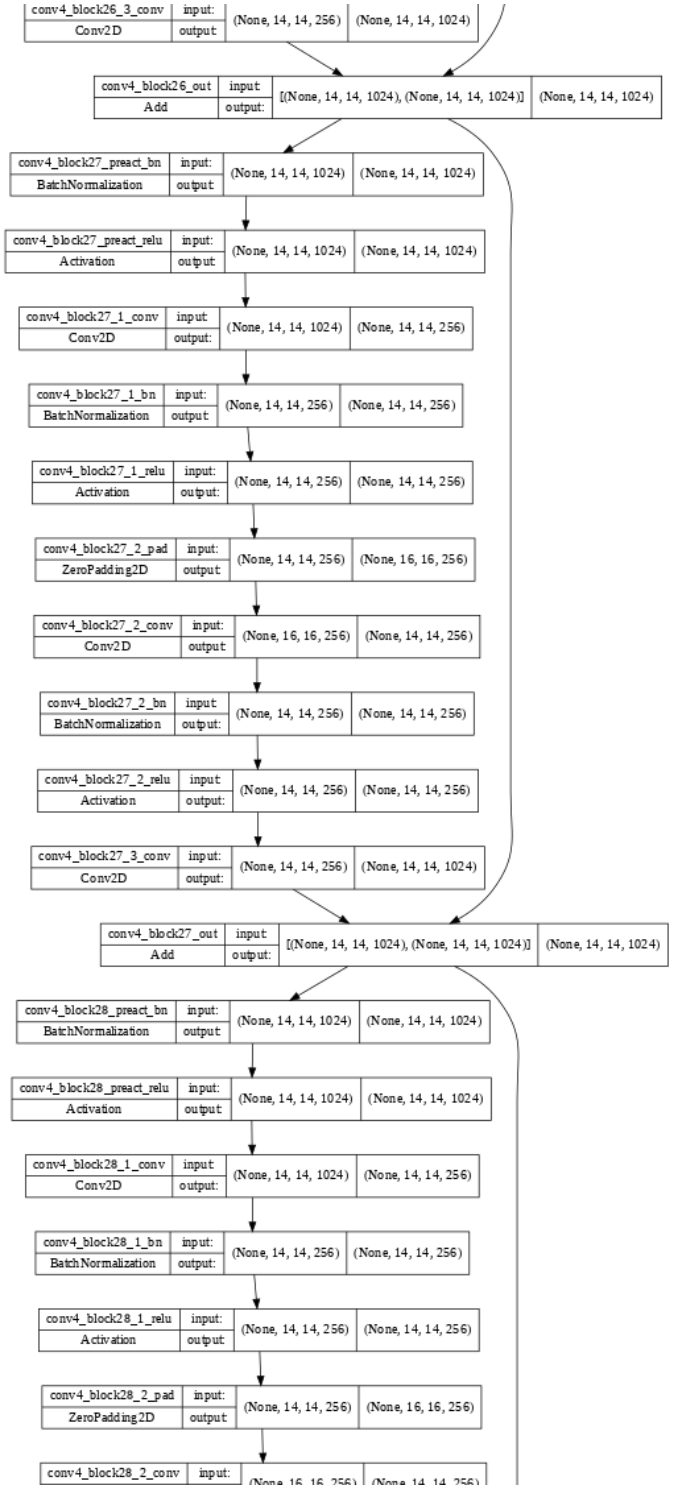

conv4_block18_preact_relu | Activation | input: | output: | (None, 14, 14, 1024) | (None, 14, 14, 1024)
conv4_block18_1_conv | Conv2D | input: | output: | (None, 14, 14, 1024) | (None, 14, 14, 256)
conv4_block18_1_bn | BatchNormalization | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block18_1_relu | Activation | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block18_2_pad | ZeroPadding2D | input: | output: | (None, 14, 14, 256) | (None, 16, 16, 256)
conv4_block18_2_conv | Conv2D | input: | output: | (None, 16, 16, 256) | (None, 14, 14, 256)
conv4_block18_2_bn | BatchNormalization | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block18_2_relu | Activation | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block18_3_conv | Conv2D | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 1024)
conv4_block18_out | Add | input: | output: | [(None, 14, 14, 1024), (None, 14, 14, 1024)] | (None, 14, 14, 1024)
conv4_block19_preact_bn | BatchNormalization | input: | output: | (None, 14, 14, 1024) | (None, 14, 14, 1024)
conv4_block19_preact_relu | Activation | input: | output: | (None, 14, 14, 1024) | (None, 14, 14, 1024)
conv4_block19_1_conv | Conv2D | input: | output: | (None, 14, 14, 1024) | (None, 14, 14, 256)
conv4_block19_1_bn | BatchNormalization | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block19_1_relu | Activation | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block19_2_pad | ZeroPadding2D | input: | output: | (None, 14, 14, 256) | (None, 16, 16, 256)
conv4_block19_2_conv | Conv2D | input: | output: | (None, 16, 16, 256) | (None, 14, 14, 256)
conv4_block19_2_bn | BatchNormalization | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block19_2_relu | Activation | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block19_3_conv | input:

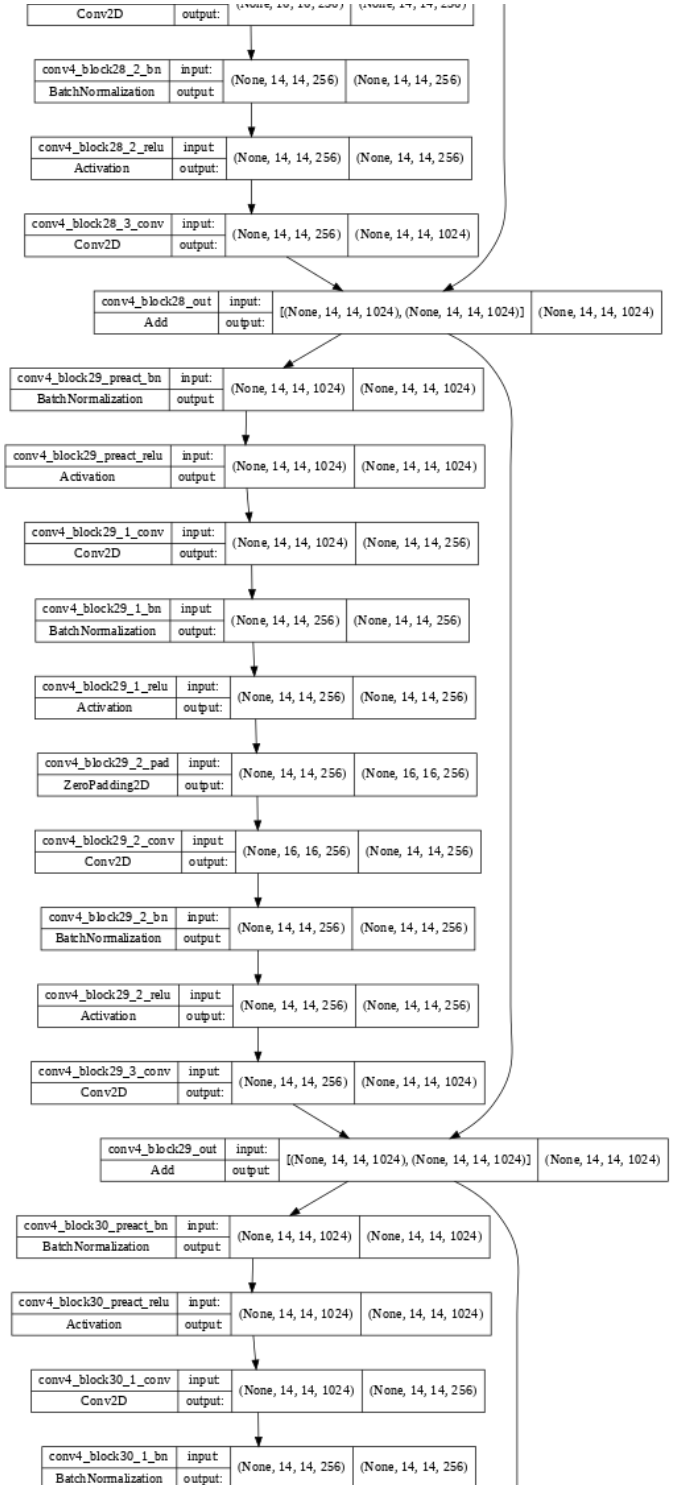

Conv2D output: (None, 14, 14, 256) (None, 14, 14, 1024)
conv4_block19_out Add input: output: [(None, 14, 14, 1024), (None, 14, 14, 1024)] (None, 14, 14, 1024)
conv4_block20_preact_bn BatchNormalization input: output: (None, 14, 14, 1024) (None, 14, 14, 1024)
conv4_block20_preact_relu Activation input: output: (None, 14, 14, 1024) (None, 14, 14, 1024)
conv4_block20_1_conv Conv2D input: output: (None, 14, 14, 1024) (None, 14, 14, 256)
conv4_block20_1_bn BatchNormalization input: output: (None, 14, 14, 256) (None, 14, 14, 256)
conv4_block20_1_relu Activation input: output: (None, 14, 14, 256) (None, 14, 14, 256)
conv4_block20_2_pad ZeroPadding2D input: output: (None, 14, 14, 256) (None, 16, 16, 256)
conv4_block20_2_conv Conv2D input: output: (None, 16, 16, 256) (None, 14, 14, 256)
conv4_block20_2_bn BatchNormalization input: output: (None, 14, 14, 256) (None, 14, 14, 256)
conv4_block20_2_relu Activation input: output: (None, 14, 14, 256) (None, 14, 14, 256)
conv4_block20_3_conv Conv2D input: output: (None, 14, 14, 256) (None, 14, 14, 1024)
conv4_block20_out Add input: output: [(None, 14, 14, 1024), (None, 14, 14, 1024)] (None, 14, 14, 1024)
conv4_block21_preact_bn BatchNormalization input: output: (None, 14, 14, 1024) (None, 14, 14, 1024)
conv4_block21_preact_relu Activation input: output: (None, 14, 14, 1024) (None, 14, 14, 1024)
conv4_block21_1_conv Conv2D input: output: (None, 14, 14, 1024) (None, 14, 14, 256)
conv4_block21_1_bn BatchNormalization input: output: (None, 14, 14, 256) (None, 14, 14, 256)
conv4_block21_1_relu Activation input: output: (None, 14, 14, 256) (None, 14, 14, 256)
conv4_block21_2_pad ZeroPadding2D input: output: (None, 14, 14, 256) (None, 16, 16, 256)
conv4_block21_2_conv Conv2D input: output: (None, 16, 16, 256) (None, 14, 14, 256)

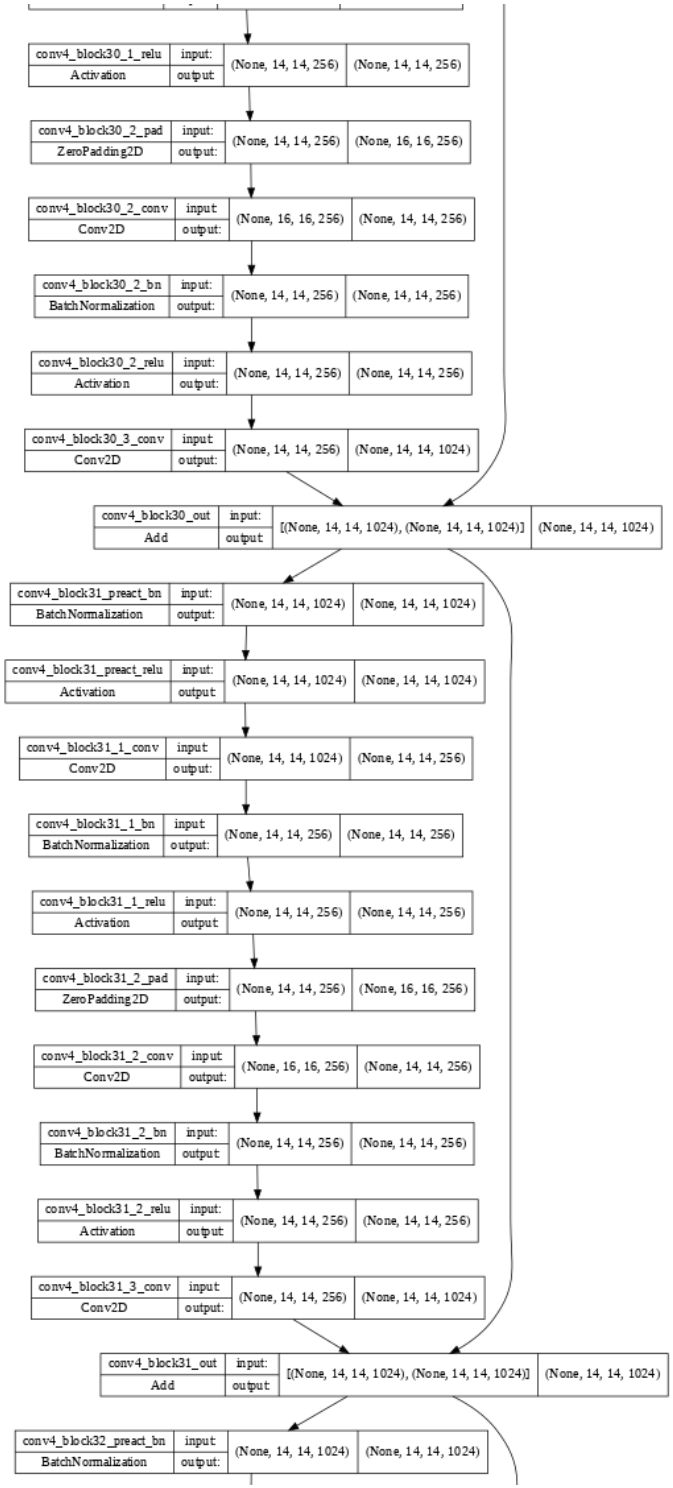

conv4_block21_2_bn | input: | (None, 14, 14, 256) | (None, 14, 14, 256)
BatchNormalization | output:
conv4_block21_2_relu | input: | (None, 14, 14, 256) | (None, 14, 14, 256)
Activation | output:
conv4_block21_3_conv | input: | (None, 14, 14, 256) | (None, 14, 14, 1024)
Conv2D | output:
conv4_block21_out | input: | [(None, 14, 14, 1024), (None, 14, 14, 1024)] | (None, 14, 14, 1024)
Add | output:
conv4_block22_preact_bn | input: | (None, 14, 14, 1024) | (None, 14, 14, 1024)
BatchNormalization | output:
conv4_block22_preact_relu | input: | (None, 14, 14, 1024) | (None, 14, 14, 1024)
Activation | output:
conv4_block22_1_conv | input: | (None, 14, 14, 1024) | (None, 14, 14, 256)
Conv2D | output:
conv4_block22_1_bn | input: | (None, 14, 14, 256) | (None, 14, 14, 256)
BatchNormalization | output:
conv4_block22_1_relu | input: | (None, 14, 14, 256) | (None, 14, 14, 256)
Activation | output:
conv4_block22_2_pad | input: | (None, 14, 14, 256) | (None, 16, 16, 256)
ZeroPadding2D | output:
conv4_block22_2_conv | input: | (None, 16, 16, 256) | (None, 14, 14, 256)
Conv2D | output:
conv4_block22_2_bn | input: | (None, 14, 14, 256) | (None, 14, 14, 256)
BatchNormalization | output:
conv4_block22_2_relu | input: | (None, 14, 14, 256) | (None, 14, 14, 256)
Activation | output:
conv4_block22_3_conv | input: | (None, 14, 14, 256) | (None, 14, 14, 1024)
Conv2D | output:
conv4_block22_out | input: | [(None, 14, 14, 1024), (None, 14, 14, 1024)] | (None, 14, 14, 1024)
Add | output:
conv4_block23_preact_bn | input: | (None, 14, 14, 1024) | (None, 14, 14, 1024)
BatchNormalization | output:
conv4_block23_preact_relu | input: | (None, 14, 14, 1024) | (None, 14, 14, 1024)
Activation | output:
conv4_block23_1_conv | input: | (None, 14, 14, 1024) | (None, 14, 14, 256)
Conv2D | output:
conv4_block23_1_bn | input: | (None, 14, 14, 256) | (None, 14, 14, 256)
BatchNormalization | output:

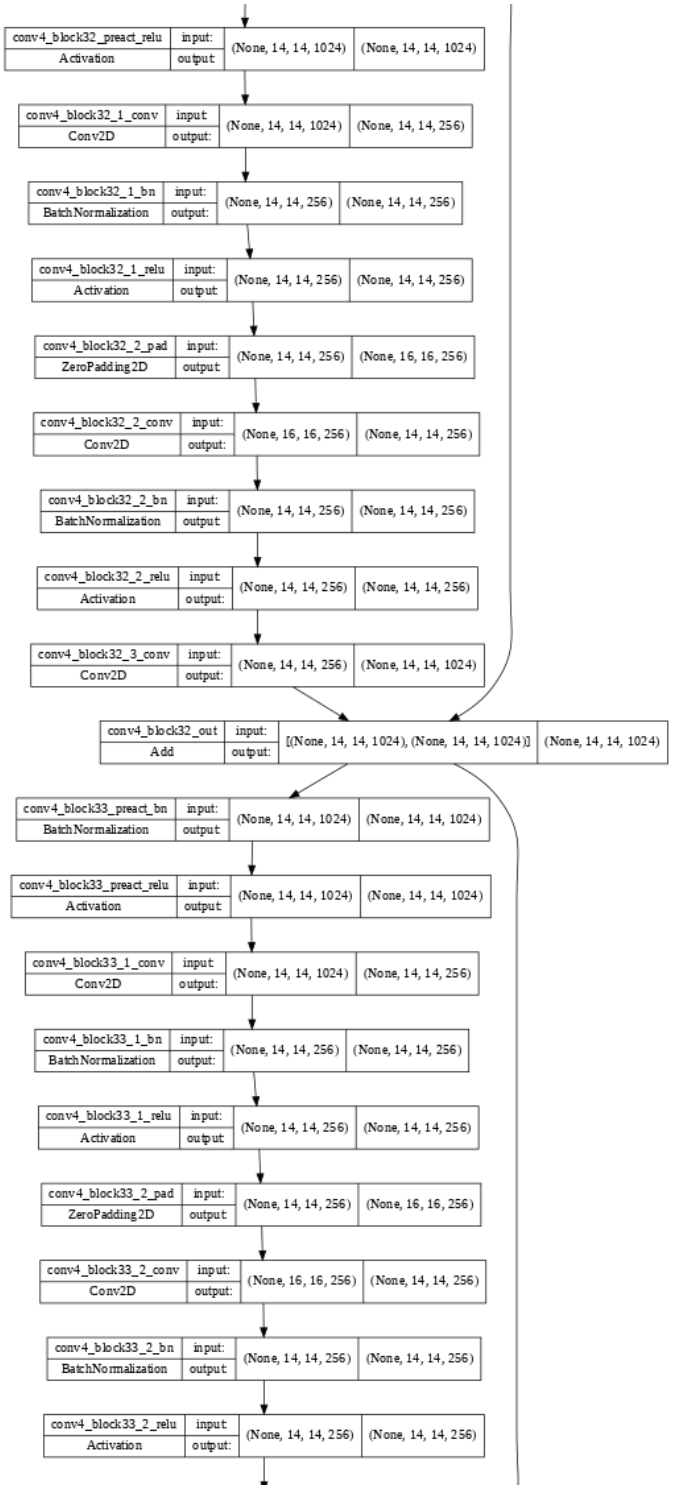

conv4_block23_1_relu | Activation | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block23_2_pad | ZeroPadding2D | input: | output: | (None, 14, 14, 256) | (None, 16, 16, 256)
conv4_block23_2_conv | Conv2D | input: | output: | (None, 16, 16, 256) | (None, 14, 14, 256)
conv4_block23_2_bn | BatchNormalization | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block23_2_relu | Activation | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block23_3_conv | Conv2D | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 1024)
conv4_block23_out | Add | input: | output: | [(None, 14, 14, 1024), (None, 14, 14, 1024)] | (None, 14, 14, 1024)
conv4_block24_preact_bn | BatchNormalization | input: | output: | (None, 14, 14, 1024) | (None, 14, 14, 1024)
conv4_block24_preact_relu | Activation | input: | output: | (None, 14, 14, 1024) | (None, 14, 14, 1024)
conv4_block24_1_conv | Conv2D | input: | output: | (None, 14, 14, 1024) | (None, 14, 14, 256)
conv4_block24_1_bn | BatchNormalization | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block24_1_relu | Activation | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block24_2_pad | ZeroPadding2D | input: | output: | (None, 14, 14, 256) | (None, 16, 16, 256)
conv4_block24_2_conv | Conv2D | input: | output: | (None, 16, 16, 256) | (None, 14, 14, 256)
conv4_block24_2_bn | BatchNormalization | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block24_2_relu | Activation | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block24_3_conv | Conv2D | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 1024)
conv4_block24_out | Add | input: | output: | [(None, 14, 14, 1024), (None, 14, 14, 1024)] | (None, 14, 14, 1024)
conv4_block25_preact_bn | BatchNormalization | input: | output: | (None, 14, 14, 1024) | (None, 14, 14, 1024)

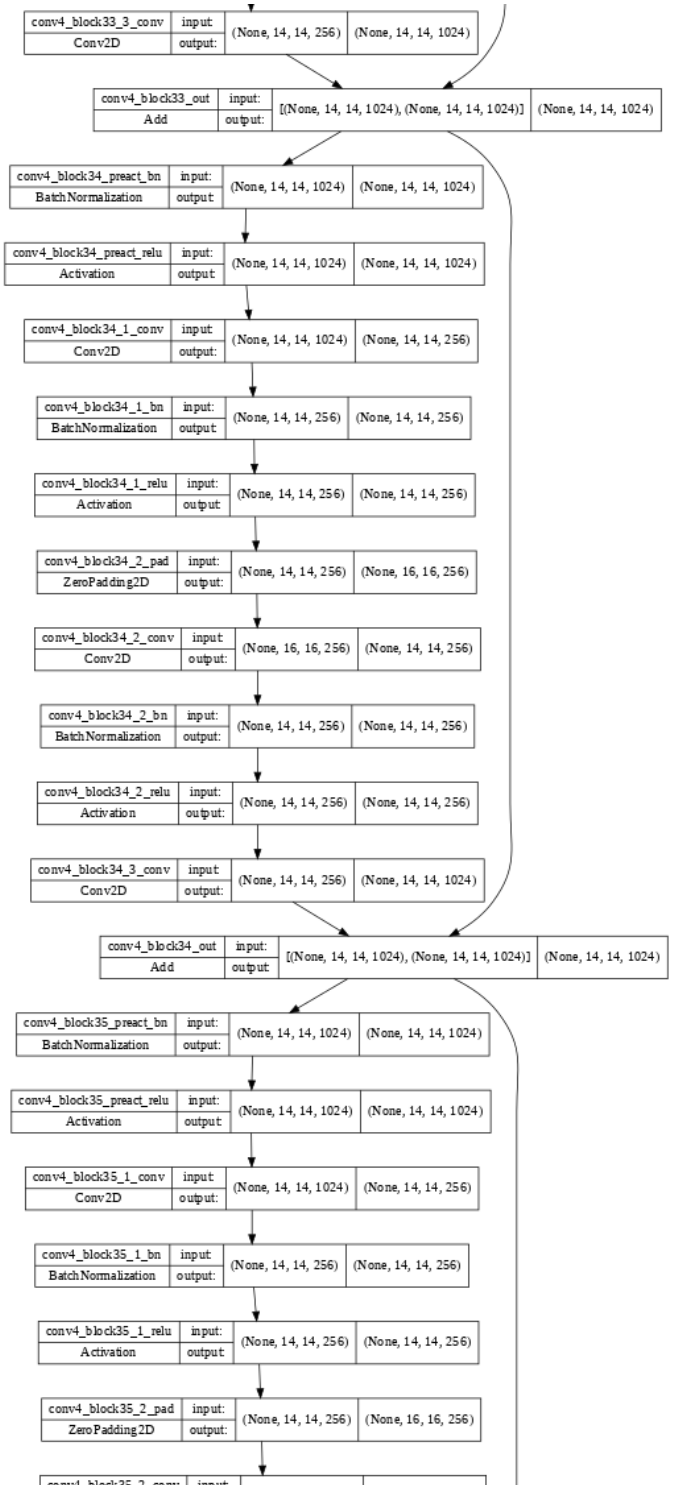

conv4_block25_preact_relu | Activation | input: | output: | (None, 14, 14, 1024) | (None, 14, 14, 1024)
conv4_block25_1_conv | Conv2D | input: | output: | (None, 14, 14, 1024) | (None, 14, 14, 256)
conv4_block25_1_bn | BatchNormalization | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block25_1_relu | Activation | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block25_2_pad | ZeroPadding2D | input: | output: | (None, 14, 14, 256) | (None, 16, 16, 256)
conv4_block25_2_conv | Conv2D | input: | output: | (None, 16, 16, 256) | (None, 14, 14, 256)
conv4_block25_2_bn | BatchNormalization | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block25_2_relu | Activation | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block25_3_conv | Conv2D | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 1024)
conv4_block25_out | Add | input | output: | [(None, 14, 14, 1024), (None, 14, 14, 1024)] | (None, 14, 14, 1024)
conv4_block26_preact_bn | BatchNormalization | input: | output: | (None, 14, 14, 1024) | (None, 14, 14, 1024)
conv4_block26_preact_relu | Activation | input: | output: | (None, 14, 14, 1024) | (None, 14, 14, 1024)
conv4_block26_1_conv | Conv2D | input: | output: | (None, 14, 14, 1024) | (None, 14, 14, 256)
conv4_block26_1_bn | BatchNormalization | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block26_1_relu | Activation | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block26_2_pad | ZeroPadding2D | input: | output: | (None, 14, 14, 256) | (None, 16, 16, 256)
conv4_block26_2_conv | Conv2D | input: | output: | (None, 16, 16, 256) | (None, 14, 14, 256)
conv4_block26_2_bn | BatchNormalization | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block26_2_relu | Activation | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)

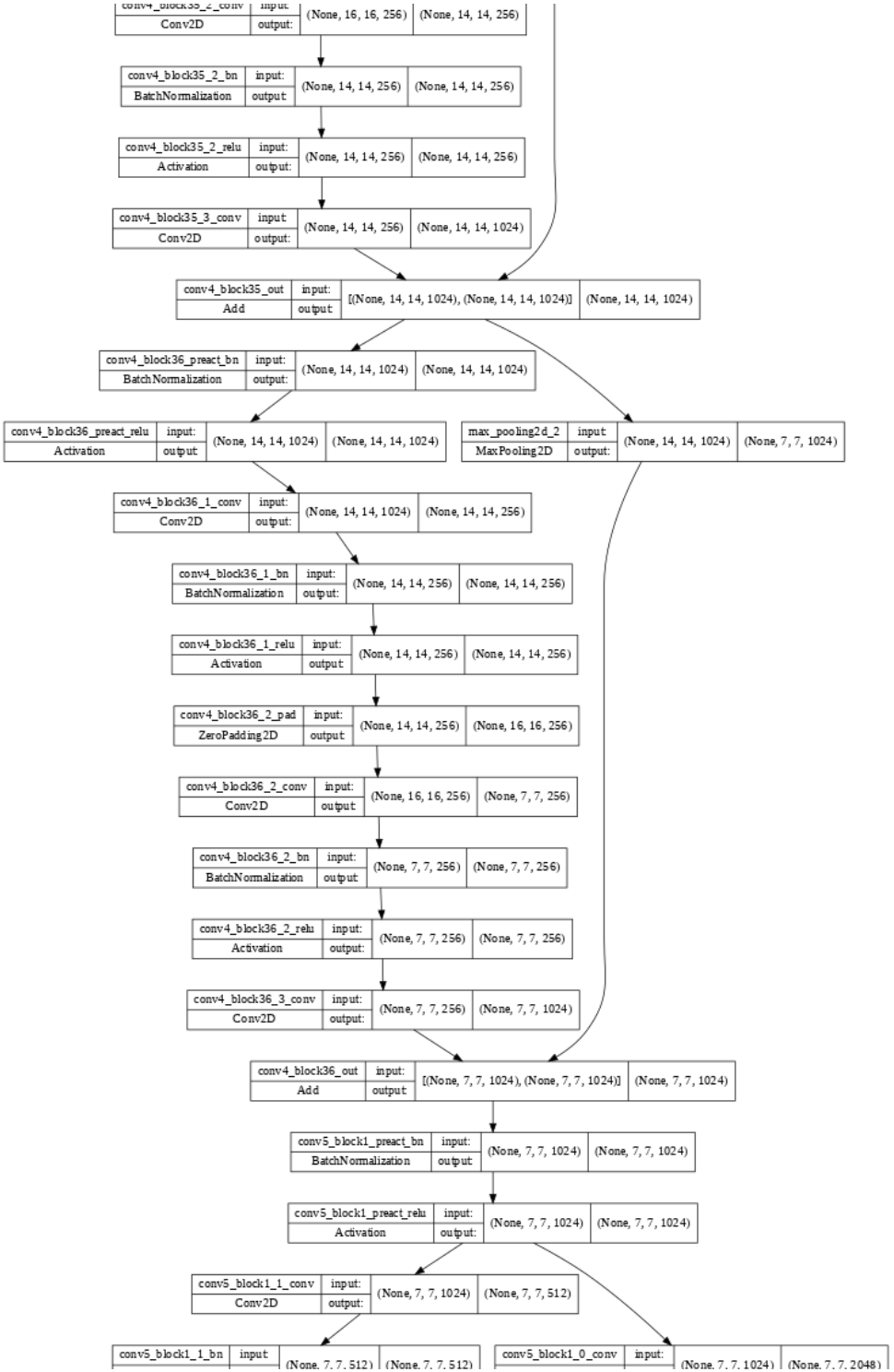

conv4_block26_3_conv | Conv2D | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 1024)
conv4_block26_out | Add | input: | output: | [(None, 14, 14, 1024), (None, 14, 14, 1024)] | (None, 14, 14, 1024)
conv4_block27_preact_bn | BatchNormalization | input: | output: | (None, 14, 14, 1024) | (None, 14, 14, 1024)
conv4_block27_preact_relu | Activation | input: | output: | (None, 14, 14, 1024) | (None, 14, 14, 1024)
conv4_block27_1_conv | Conv2D | input: | output: | (None, 14, 14, 1024) | (None, 14, 14, 256)
conv4_block27_1_bn | BatchNormalization | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block27_1_relu | Activation | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block27_2_pad | ZeroPadding2D | input: | output: | (None, 14, 14, 256) | (None, 16, 16, 256)
conv4_block27_2_conv | Conv2D | input: | output: | (None, 16, 16, 256) | (None, 14, 14, 256)
conv4_block27_2_bn | BatchNormalization | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block27_2_relu | Activation | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block27_3_conv | Conv2D | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 1024)
conv4_block27_out | Add | input: | output: | [(None, 14, 14, 1024), (None, 14, 14, 1024)] | (None, 14, 14, 1024)
conv4_block28_preact_bn | BatchNormalization | input: | output: | (None, 14, 14, 1024) | (None, 14, 14, 1024)
conv4_block28_preact_relu | Activation | input: | output: | (None, 14, 14, 1024) | (None, 14, 14, 1024)
conv4_block28_1_conv | Conv2D | input: | output: | (None, 14, 14, 1024) | (None, 14, 14, 256)
conv4_block28_1_bn | BatchNormalization | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block28_1_relu | Activation | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block28_2_pad | ZeroPadding2D | input: | output: | (None, 14, 14, 256) | (None, 16, 16, 256)
conv4_block28_2_conv | input:

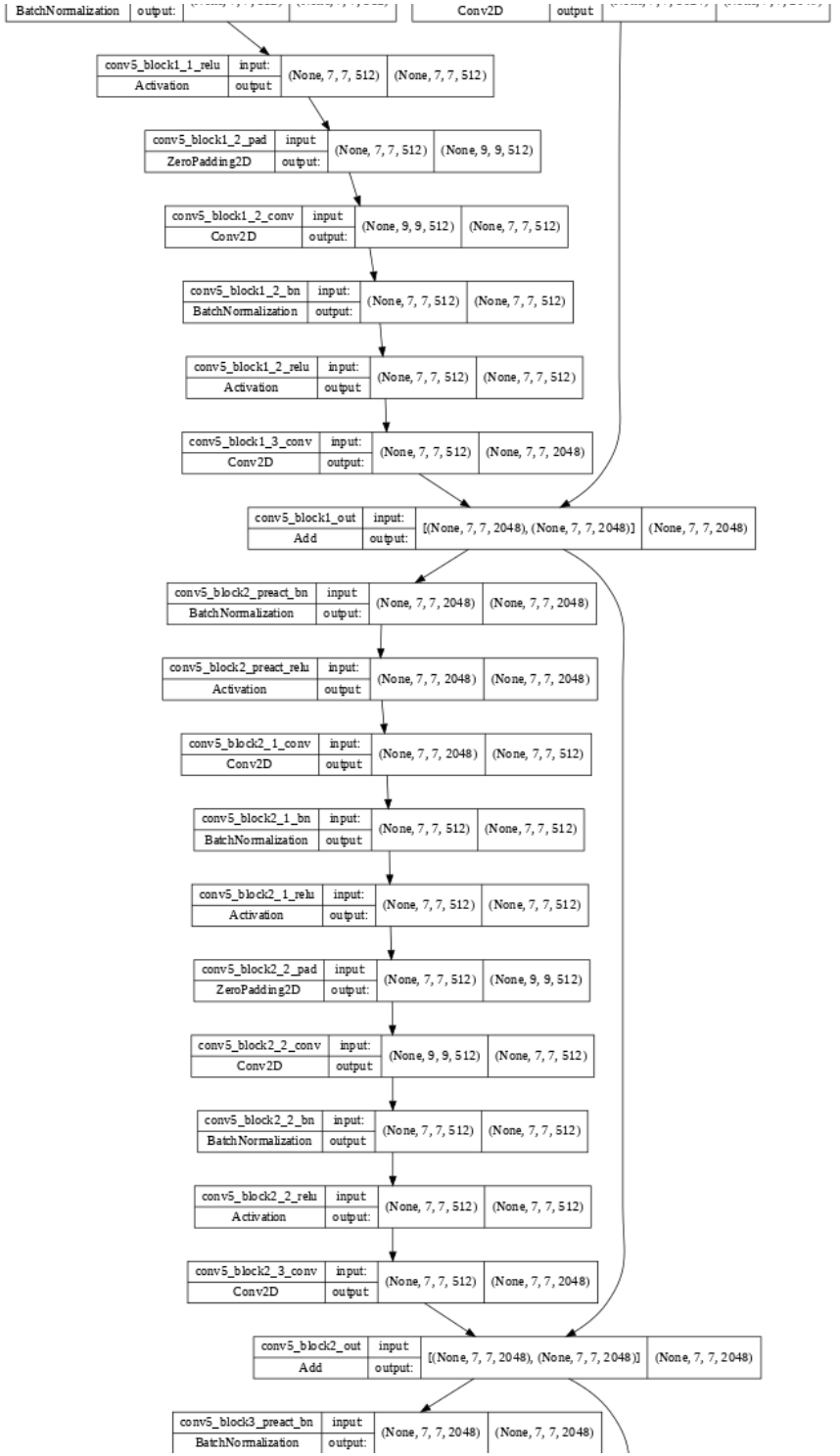

Conv2D output:
conv4_block28_2_bn input: (None, 14, 14, 256) (None, 14, 14, 256)
BatchNormalization output:
conv4_block28_2_relu input: (None, 14, 14, 256) (None, 14, 14, 256)
Activation output:
conv4_block28_3_conv input: (None, 14, 14, 256) (None, 14, 14, 1024)
Conv2D output:
conv4_block28_out input: [(None, 14, 14, 1024), (None, 14, 14, 1024)] (None, 14, 14, 1024)
Add output:
conv4_block29_preact_bn input: (None, 14, 14, 1024) (None, 14, 14, 1024)
BatchNormalization output:
conv4_block29_preact_relu input: (None, 14, 14, 1024) (None, 14, 14, 1024)
Activation output:
conv4_block29_1_conv input: (None, 14, 14, 1024) (None, 14, 14, 256)
Conv2D output:
conv4_block29_1_bn input: (None, 14, 14, 256) (None, 14, 14, 256)
BatchNormalization output:
conv4_block29_1_relu input: (None, 14, 14, 256) (None, 14, 14, 256)
Activation output:
conv4_block29_2_pad input: (None, 14, 14, 256) (None, 16, 16, 256)
ZeroPadding2D output:
conv4_block29_2_conv input: (None, 16, 16, 256) (None, 14, 14, 256)
Conv2D output:
conv4_block29_2_bn input: (None, 14, 14, 256) (None, 14, 14, 256)
BatchNormalization output:
conv4_block29_2_relu input: (None, 14, 14, 256) (None, 14, 14, 256)
Activation output:
conv4_block29_3_conv input: (None, 14, 14, 256) (None, 14, 14, 1024)
Conv2D output:
conv4_block29_out input: [(None, 14, 14, 1024), (None, 14, 14, 1024)] (None, 14, 14, 1024)
Add output:
conv4_block30_preact_bn input: (None, 14, 14, 1024) (None, 14, 14, 1024)
BatchNormalization output:
conv4_block30_preact_relu input: (None, 14, 14, 1024) (None, 14, 14, 1024)
Activation output:
conv4_block30_1_conv input: (None, 14, 14, 1024) (None, 14, 14, 256)
Conv2D output:
conv4_block30_1_bn input (None, 14, 14, 256) (None, 14, 14, 256)
BatchNormalization output:

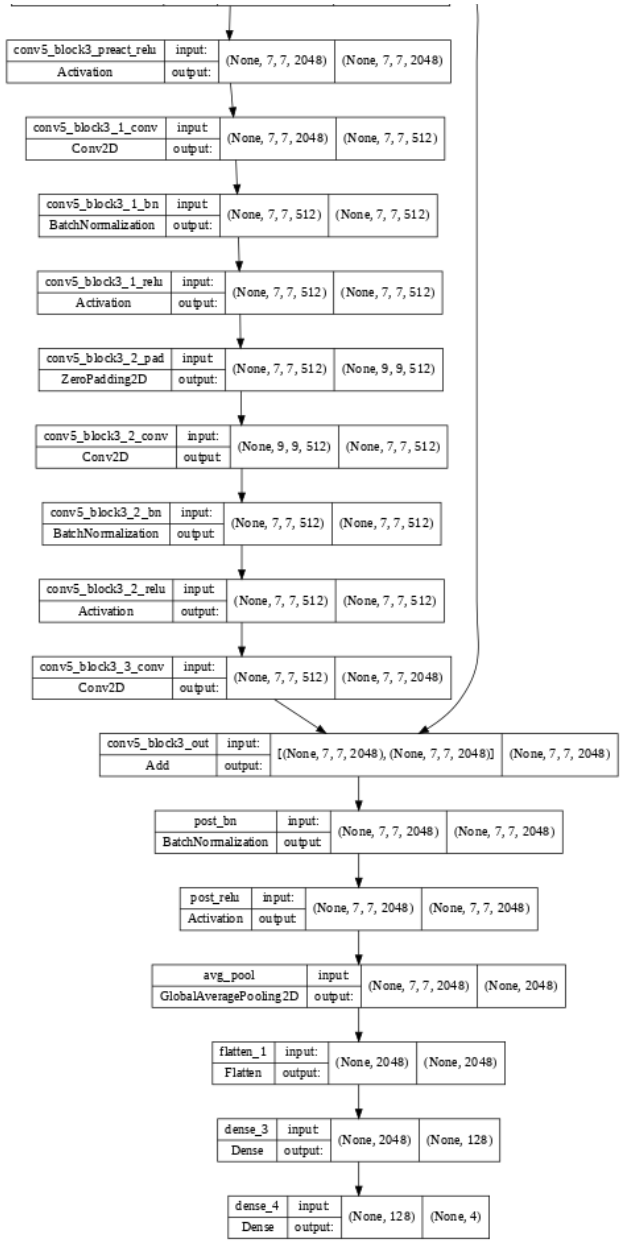

conv4_block30_1_relu | Activation | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block30_2_pad | ZeroPadding2D | input: | output: | (None, 14, 14, 256) | (None, 16, 16, 256)
conv4_block30_2_conv | Conv2D | input: | output: | (None, 16, 16, 256) | (None, 14, 14, 256)
conv4_block30_2_bn | BatchNormalization | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block30_2_relu | Activation | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block30_3_conv | Conv2D | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 1024)
conv4_block30_out | Add | input: | output: | [(None, 14, 14, 1024), (None, 14, 14, 1024)] | (None, 14, 14, 1024)
conv4_block31_preact_bn | BatchNormalization | input: | output: | (None, 14, 14, 1024) | (None, 14, 14, 1024)
conv4_block31_preact_relu | Activation | input: | output: | (None, 14, 14, 1024) | (None, 14, 14, 1024)
conv4_block31_1_conv | Conv2D | input: | output: | (None, 14, 14, 1024) | (None, 14, 14, 256)
conv4_block31_1_bn | BatchNormalization | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block31_1_relu | Activation | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block31_2_pad | ZeroPadding2D | input: | output: | (None, 14, 14, 256) | (None, 16, 16, 256)
conv4_block31_2_conv | Conv2D | input: | output: | (None, 16, 16, 256) | (None, 14, 14, 256)
conv4_block31_2_bn | BatchNormalization | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block31_2_relu | Activation | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block31_3_conv | Conv2D | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 1024)
conv4_block31_out | Add | input: | output: | [(None, 14, 14, 1024), (None, 14, 14, 1024)] | (None, 14, 14, 1024)
conv4_block32_preact_bn | BatchNormalization | input: | output: | (None, 14, 14, 1024) | (None, 14, 14, 1024)


conv4_block32_preact_relu | Activation | input: output: | (None, 14, 14, 1024) | (None, 14, 14, 1024)
conv4_block32_1_conv | Conv2D | input: output: | (None, 14, 14, 1024) | (None, 14, 14, 256)
conv4_block32_1_bn | BatchNormalization | input: output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block32_1_relu | Activation | input: output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block32_2_pad | ZeroPadding2D | input: output: | (None, 14, 14, 256) | (None, 16, 16, 256)
conv4_block32_2_conv | Conv2D | input: output: | (None, 16, 16, 256) | (None, 14, 14, 256)
conv4_block32_2_bn | BatchNormalization | input: output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block32_2_relu | Activation | input: output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block32_3_conv | Conv2D | input: output: | (None, 14, 14, 256) | (None, 14, 14, 1024)
conv4_block32_out | Add | input: output: | [(None, 14, 14, 1024), (None, 14, 14, 1024)] | (None, 14, 14, 1024)
conv4_block33_preact_bn | BatchNormalization | input: output: | (None, 14, 14, 1024) | (None, 14, 14, 1024)
conv4_block33_preact_relu | Activation | input: output: | (None, 14, 14, 1024) | (None, 14, 14, 1024)
conv4_block33_1_conv | Conv2D | input: output: | (None, 14, 14, 1024) | (None, 14, 14, 256)
conv4_block33_1_bn | BatchNormalization | input: output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block33_1_relu | Activation | input: output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block33_2_pad | ZeroPadding2D | input: output: | (None, 14, 14, 256) | (None, 16, 16, 256)
conv4_block33_2_conv | Conv2D | input: output: | (None, 16, 16, 256) | (None, 14, 14, 256)
conv4_block33_2_bn | BatchNormalization | input: output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block33_2_relu | Activation | input: output: | (None, 14, 14, 256) | (None, 14, 14, 256)


conv4_block33_3_conv | Conv2D | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 1024)
conv4_block33_out | Add | input: | output: | [(None, 14, 14, 1024), (None, 14, 14, 1024)] | (None, 14, 14, 1024)
conv4_block34_preact_bn | BatchNormalization | input: | output: | (None, 14, 14, 1024) | (None, 14, 14, 1024)
conv4_block34_preact_relu | Activation | input: | output: | (None, 14, 14, 1024) | (None, 14, 14, 1024)
conv4_block34_1_conv | Conv2D | input: | output: | (None, 14, 14, 1024) | (None, 14, 14, 256)
conv4_block34_1_bn | BatchNormalization | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block34_1_relu | Activation | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block34_2_pad | ZeroPadding2D | input: | output: | (None, 14, 14, 256) | (None, 16, 16, 256)
conv4_block34_2_conv | Conv2D | input: | output: | (None, 16, 16, 256) | (None, 14, 14, 256)
conv4_block34_2_bn | BatchNormalization | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block34_2_relu | Activation | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block34_3_conv | Conv2D | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 1024)
conv4_block34_out | Add | input: | output: | [(None, 14, 14, 1024), (None, 14, 14, 1024)] | (None, 14, 14, 1024)
conv4_block35_preact_bn | BatchNormalization | input: | output: | (None, 14, 14, 1024) | (None, 14, 14, 1024)
conv4_block35_preact_relu | Activation | input: | output: | (None, 14, 14, 1024) | (None, 14, 14, 1024)
conv4_block35_1_conv | Conv2D | input: | output: | (None, 14, 14, 1024) | (None, 14, 14, 256)
conv4_block35_1_bn | BatchNormalization | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block35_1_relu | Activation | input: | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block35_2_pad | ZeroPadding2D | input: | output: | (None, 14, 14, 256) | (None, 16, 16, 256)


Conv2D | output: | (None, 16, 16, 256) | (None, 14, 14, 256)
conv4_block35_2_bn | input: | BatchNormalization | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block35_2_relu | input: | Activation | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block35_3_conv | input: | Conv2D | output: | (None, 14, 14, 256) | (None, 14, 14, 1024)
conv4_block35_out | input: | Add | output: | [(None, 14, 14, 1024), (None, 14, 14, 1024)] | (None, 14, 14, 1024)
conv4_block36_preact_bn | input: | BatchNormalization | output: | (None, 14, 14, 1024) | (None, 14, 14, 1024)
conv4_block36_preact_relu | input: | Activation | output: | (None, 14, 14, 1024) | (None, 14, 14, 1024)
max_pooling2d_2 | input | MaxPooling2D | output: | (None, 14, 14, 1024) | (None, 7, 7, 1024)
conv4_block36_1_conv | input: | Conv2D | output: | (None, 14, 14, 1024) | (None, 14, 14, 256)
conv4_block36_1_bn | input: | BatchNormalization | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block36_1_relu | input: | Activation | output: | (None, 14, 14, 256) | (None, 14, 14, 256)
conv4_block36_2_pad | input: | ZeroPadding2D | output: | (None, 14, 14, 256) | (None, 16, 16, 256)
conv4_block36_2_conv | input: | Conv2D | output: | (None, 16, 16, 256) | (None, 7, 7, 256)
conv4_block36_2_bn | input: | BatchNormalization | output: | (None, 7, 7, 256) | (None, 7, 7, 256)
conv4_block36_2_relu | input: | Activation | output: | (None, 7, 7, 256) | (None, 7, 7, 256)
conv4_block36_3_conv | input: | Conv2D | output: | (None, 7, 7, 256) | (None, 7, 7, 1024)
conv4_block36_out | input: | Add | output: | [(None, 7, 7, 1024), (None, 7, 7, 1024)] | (None, 7, 7, 1024)
conv5_block1_preact_bn | input: | BatchNormalization | output: | (None, 7, 7, 1024) | (None, 7, 7, 1024)
conv5_block1_preact_relu | input: | Activation | output: | (None, 7, 7, 1024) | (None, 7, 7, 1024)
conv5_block1_1_conv | input: | Conv2D | output: | (None, 7, 7, 1024) | (None, 7, 7, 512)
conv5_block1_1_bn | input
(None, 7, 7, 512) | (None, 7, 7, 512)
conv5_block1_0_conv | input:
(None, 7, 7, 1024) | (None, 7, 7, 2048)


BatchNormalization output:
Conv2D output
conv5_block1_1_relu Activation input: output: (None, 7, 7, 512) (None, 7, 7, 512)
conv5_block1_2_pad ZeroPadding2D input: output: (None, 7, 7, 512) (None, 9, 9, 512)
conv5_block1_2_conv Conv2D input: output: (None, 9, 9, 512) (None, 7, 7, 512)
conv5_block1_2_bn BatchNormalization input: output: (None, 7, 7, 512) (None, 7, 7, 512)
conv5_block1_2_relu Activation input: output: (None, 7, 7, 512) (None, 7, 7, 512)
conv5_block1_3_conv Conv2D input: output: (None, 7, 7, 512) (None, 7, 7, 2048)
conv5_block1_out Add input: output: [(None, 7, 7, 2048), (None, 7, 7, 2048)] (None, 7, 7, 2048)
conv5_block2_preact_bn BatchNormalization input: output: (None, 7, 7, 2048) (None, 7, 7, 2048)
conv5_block2_preact_relu Activation input: output: (None, 7, 7, 2048) (None, 7, 7, 2048)
conv5_block2_1_conv Conv2D input: output: (None, 7, 7, 2048) (None, 7, 7, 512)
conv5_block2_1_bn BatchNormalization input: output: (None, 7, 7, 512) (None, 7, 7, 512)
conv5_block2_1_relu Activation input: output: (None, 7, 7, 512) (None, 7, 7, 512)
conv5_block2_2_pad ZeroPadding2D input: output: (None, 7, 7, 512) (None, 9, 9, 512)
conv5_block2_2_conv Conv2D input: output: (None, 9, 9, 512) (None, 7, 7, 512)
conv5_block2_2_bn BatchNormalization input: output: (None, 7, 7, 512) (None, 7, 7, 512)
conv5_block2_2_relu Activation input: output: (None, 7, 7, 512) (None, 7, 7, 512)
conv5_block2_3_conv Conv2D input: output: (None, 7, 7, 512) (None, 7, 7, 2048)
conv5_block2_out Add input: output: [(None, 7, 7, 2048), (None, 7, 7, 2048)] (None, 7, 7, 2048)
conv5_block3_preact_bn BatchNormalization input: output: (None, 7, 7, 2048) (None, 7, 7, 2048)



#5. Transfer Learning with *InceptionV* and *Data Augmentation*

This model was announced by the Google team in December 2015. V2 has two 3x3 convolution kernels instead of a 5x5, while V3 will decompose more thoroughly. The core idea of V3 is to first use two 3x3 convolution kernels instead of 5x5 convolution kernels, and three 3x3 convolution kernels instead of 7x7 convolution kernels to reduce the amount of parameters and speed up

calculations. V3 further decomposes the nxn convolution kernel into 1xn and nx1 convolution kernels, while reducing the size of the feature map and increasing the number of channels.

In [ ]:
```
# Set InceptionV3 layers to frozen to keep the weights already made
base_incv3 = tf.keras.applications.InceptionV3(input_shape=(224,224,3),include_top=False,weights='imagenet')

# Add layers to model
inputs_incv3 = base_incv3.input
incv3 = tf.keras.layers.Flatten()(base_incv3.output)
incv3 = tf.keras.layers.Dense(124, activation='relu')(incv3)
incv3 = tf.keras.layers.BatchNormalization()(incv3)
incv3 = tf.keras.layers.Dropout(0.25)(incv3)
outputs_incv3 = tf.keras.layers.Dense(4, activation='softmax')(incv3)

# Set the correct inputs and outputs to the model
model_incv3 = tf.keras.Model(inputs=inputs_incv3, outputs=outputs_incv3)

# Compile the model
model_incv3.compile(optimizer=keras.optimizers.Adam(learning_rate = 0.0001),loss='categorical_crossentropy',metri
```

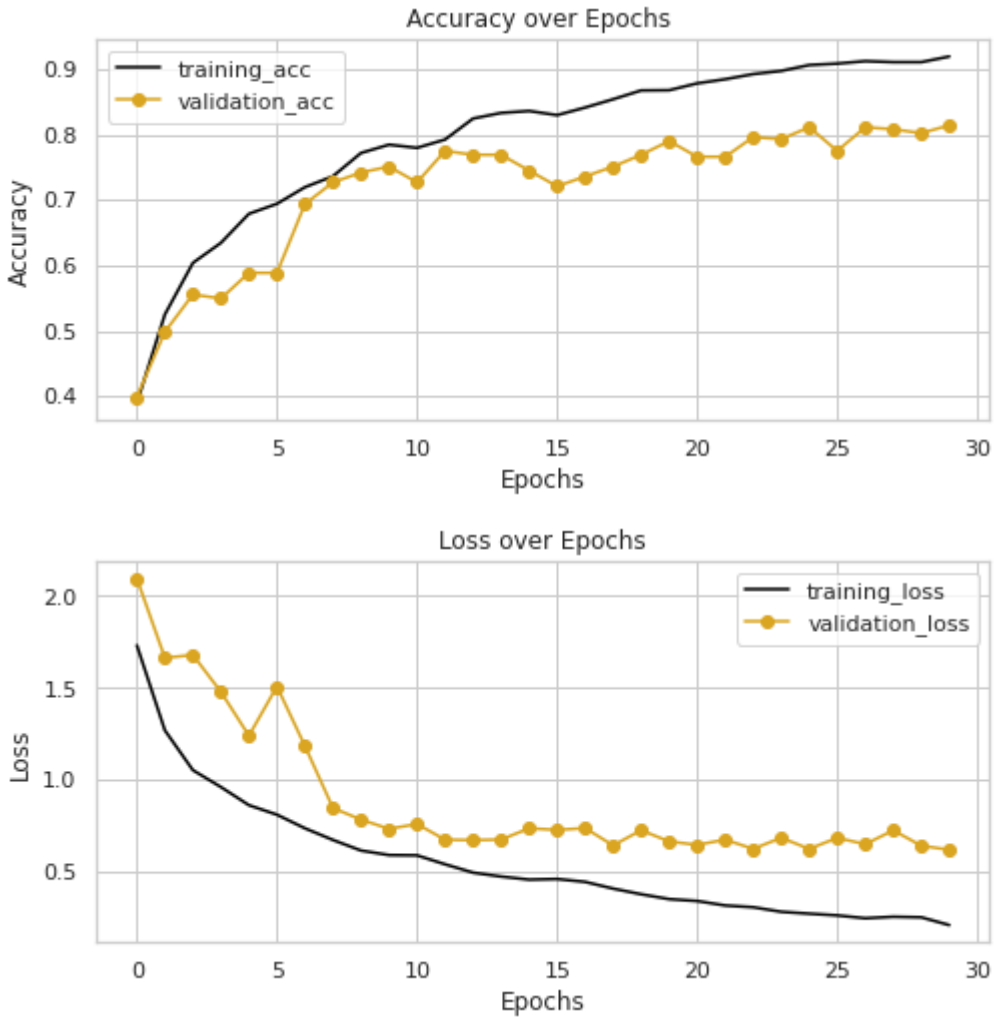
In [ ]:
```
# Fit the model
history_incv3=model_incv3.fit(train_set_augmented,
              epochs=30,
              validation_data=val_set,
              verbose=2)
```

```
Epoch 1/30
21/21 - 47s - loss: 1.7327 - accuracy: 0.3899 - val_loss: 2.0960 - val_accuracy: 0.3994 - 47s/epoch - 2s/step
Epoch 2/30
21/21 - 37s - loss: 1.2703 - accuracy: 0.5252 - val_loss: 1.6644 - val_accuracy: 0.4985 - 37s/epoch - 2s/step
Epoch 3/30
21/21 - 37s - loss: 1.0538 - accuracy: 0.6038 - val_loss: 1.6803 - val_accuracy: 0.5556 - 37s/epoch - 2s/step
Epoch 4/30
21/21 - 37s - loss: 0.9621 - accuracy: 0.6344 - val_loss: 1.4797 - val_accuracy: 0.5495 - 37s/epoch - 2s/step
Epoch 5/30
21/21 - 37s - loss: 0.8633 - accuracy: 0.6787 - val_loss: 1.2413 - val_accuracy: 0.5886 - 37s/epoch - 2s/step
Epoch 6/30
21/21 - 37s - loss: 0.8117 - accuracy: 0.6940 - val_loss: 1.5092 - val_accuracy: 0.5886 - 37s/epoch - 2s/step
Epoch 7/30
21/21 - 37s - loss: 0.7377 - accuracy: 0.7193 - val_loss: 1.1849 - val_accuracy: 0.6937 - 37s/epoch - 2s/step
Epoch 8/30
21/21 - 37s - loss: 0.6742 - accuracy: 0.7361 - val_loss: 0.8478 - val_accuracy: 0.7267 - 37s/epoch - 2s/step
```

```
Epoch 9/30
21/21 - 37s - loss: 0.6167 - accuracy: 0.7715 - val_loss: 0.7850 - val_accuracy: 0.7417 - 37s/epoch - 2s/step
Epoch 10/30
21/21 - 37s - loss: 0.5920 - accuracy: 0.7842 - val_loss: 0.7347 - val_accuracy: 0.7508 - 37s/epoch - 2s/step
Epoch 11/30
21/21 - 37s - loss: 0.5905 - accuracy: 0.7794 - val_loss: 0.7610 - val_accuracy: 0.7267 - 37s/epoch - 2s/step
Epoch 12/30
21/21 - 38s - loss: 0.5420 - accuracy: 0.7920 - val_loss: 0.6758 - val_accuracy: 0.7748 - 38s/epoch - 2s/step
Epoch 13/30
21/21 - 37s - loss: 0.4966 - accuracy: 0.8241 - val_loss: 0.6736 - val_accuracy: 0.7688 - 37s/epoch - 2s/step
Epoch 14/30
21/21 - 38s - loss: 0.4753 - accuracy: 0.8327 - val_loss: 0.6762 - val_accuracy: 0.7688 - 38s/epoch - 2s/step
Epoch 15/30
21/21 - 37s - loss: 0.4586 - accuracy: 0.8356 - val_loss: 0.7376 - val_accuracy: 0.7447 - 37s/epoch - 2s/step
Epoch 16/30
21/21 - 37s - loss: 0.4619 - accuracy: 0.8293 - val_loss: 0.7288 - val_accuracy: 0.7207 - 37s/epoch - 2s/step
Epoch 17/30
21/21 - 38s - loss: 0.4463 - accuracy: 0.8408 - val_loss: 0.7386 - val_accuracy: 0.7357 - 38s/epoch - 2s/step
Epoch 18/30
21/21 - 37s - loss: 0.4098 - accuracy: 0.8535 - val_loss: 0.6447 - val_accuracy: 0.7508 - 37s/epoch - 2s/step
Epoch 19/30
21/21 - 37s - loss: 0.3790 - accuracy: 0.8669 - val_loss: 0.7271 - val_accuracy: 0.7688 - 37s/epoch - 2s/step
Epoch 20/30
21/21 - 37s - loss: 0.3525 - accuracy: 0.8673 - val_loss: 0.6662 - val_accuracy: 0.7898 - 37s/epoch - 2s/step
Epoch 21/30
21/21 - 37s - loss: 0.3423 - accuracy: 0.8777 - val_loss: 0.6475 - val_accuracy: 0.7658 - 37s/epoch - 2s/step
Epoch 22/30
21/21 - 37s - loss: 0.3180 - accuracy: 0.8841 - val_loss: 0.6759 - val_accuracy: 0.7658 - 37s/epoch - 2s/step
Epoch 23/30
21/21 - 37s - loss: 0.3084 - accuracy: 0.8919 - val_loss: 0.6259 - val_accuracy: 0.7958 - 37s/epoch - 2s/step
Epoch 24/30
21/21 - 37s - loss: 0.2837 - accuracy: 0.8968 - val_loss: 0.6861 - val_accuracy: 0.7928 - 37s/epoch - 2s/step
Epoch 25/30
21/21 - 37s - loss: 0.2734 - accuracy: 0.9057 - val_loss: 0.6257 - val_accuracy: 0.8108 - 37s/epoch - 2s/step
Epoch 26/30
21/21 - 37s - loss: 0.2640 - accuracy: 0.9079 - val_loss: 0.6857 - val_accuracy: 0.7748 - 37s/epoch - 2s/step
Epoch 27/30
21/21 - 37s - loss: 0.2485 - accuracy: 0.9117 - val_loss: 0.6513 - val_accuracy: 0.8108 - 37s/epoch - 2s/step
Epoch 28/30
21/21 - 37s - loss: 0.2566 - accuracy: 0.9102 - val_loss: 0.7265 - val_accuracy: 0.8078 - 37s/epoch - 2s/step
Epoch 29/30
21/21 - 37s - loss: 0.2536 - accuracy: 0.9102 - val_loss: 0.6429 - val_accuracy: 0.8018 - 37s/epoch - 2s/step
Epoch 30/30
21/21 - 37s - loss: 0.2114 - accuracy: 0.9187 - val_loss: 0.6208 - val_accuracy: 0.8138 - 37s/epoch - 2s/step
```

In [ ]:

```
plot_learning_curves(history_incv3)
```



Bringing in Holdout Set

In [ ]:

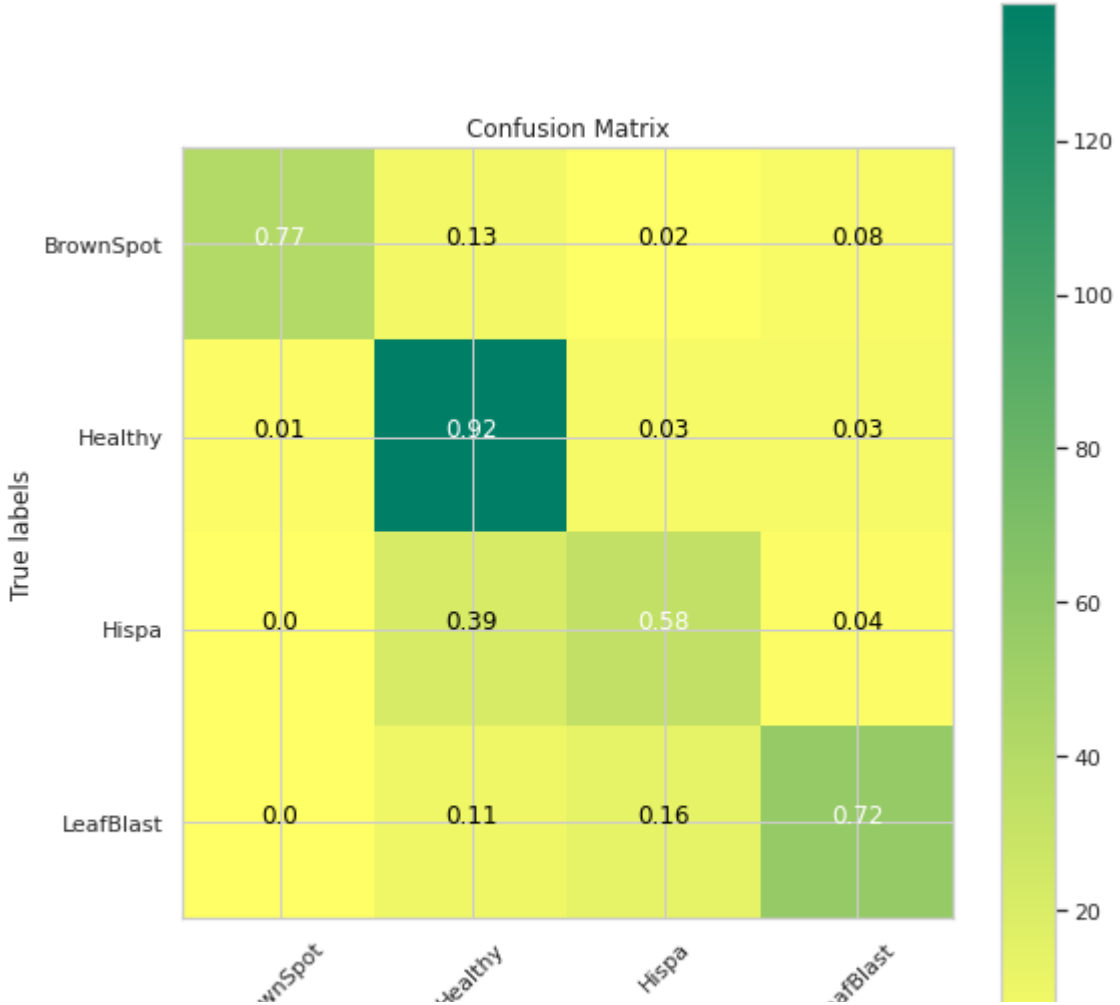
```
# Make predictions with model
test_loss, test_acc = model_incv3.evaluate(test_set)
print('Testing accuracy:', test_acc)
```

```
11/11 [==============================] - 1s 69ms/step - loss: 0.7570 - accuracy: 0.7935
Testing accuracy: 0.7935103178024292
```

In [ ]:

```
#Plot the confusion matrix. Set Normalize = True

def plot_confusion_matrix(cm, classes, normalize=True, title='Confusion Matrix'):

    plt.figure(figsize=(8,8))
    plt.imshow(cm, interpolation='nearest', cmap='summer_r')
    plt.title(title)
    plt.colorbar()

    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        cm = np.around(cm, decimals=2)
        cm[np.isnan(cm)] = 0.0
        print("Normalized Confusion Matrix")
    else:
        print('Confusion Matrix, without normalization')
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, cm[i, j],
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")
    plt.tight_layout()
    plt.ylabel('True labels')
    plt.xlabel('Predicted labels')

#Print the Target names
target_names = []
for key in train_set.class_indices:
    target_names.append(key)

#Confution Matrix
Y_pred = model_incv3.predict_generator(test_set)
y_pred = np.argmax(Y_pred, axis=1)
print('Confusion Matrix')
cm = confusion_matrix(test_set.classes, y_pred)
plot_confusion_matrix(cm, target_names, title='Confusion Matrix')

#Print Classification Report
print('Classification Report')
print(classification_report(test_set.classes, y_pred, target_names=target_names))
```

```
Confusion Matrix
Normalized Confusion Matrix
Classification Report
              precision    recall  f1-score   support

   BrownSpot       0.95      0.77      0.85        53
     Healthy       0.78      0.92      0.85       150
       Hispa       0.63      0.58      0.61        57
   LeafBlast       0.84      0.72      0.78        79

    accuracy                           0.79       339
   macro avg       0.80      0.75      0.77       339
weighted avg       0.80      0.79      0.79       339
```

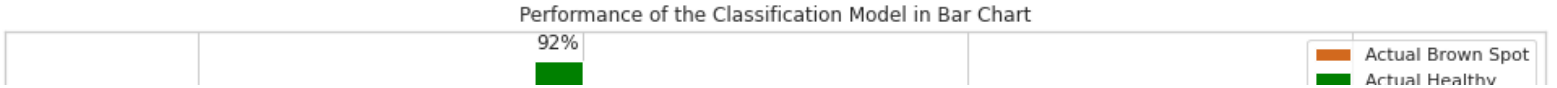


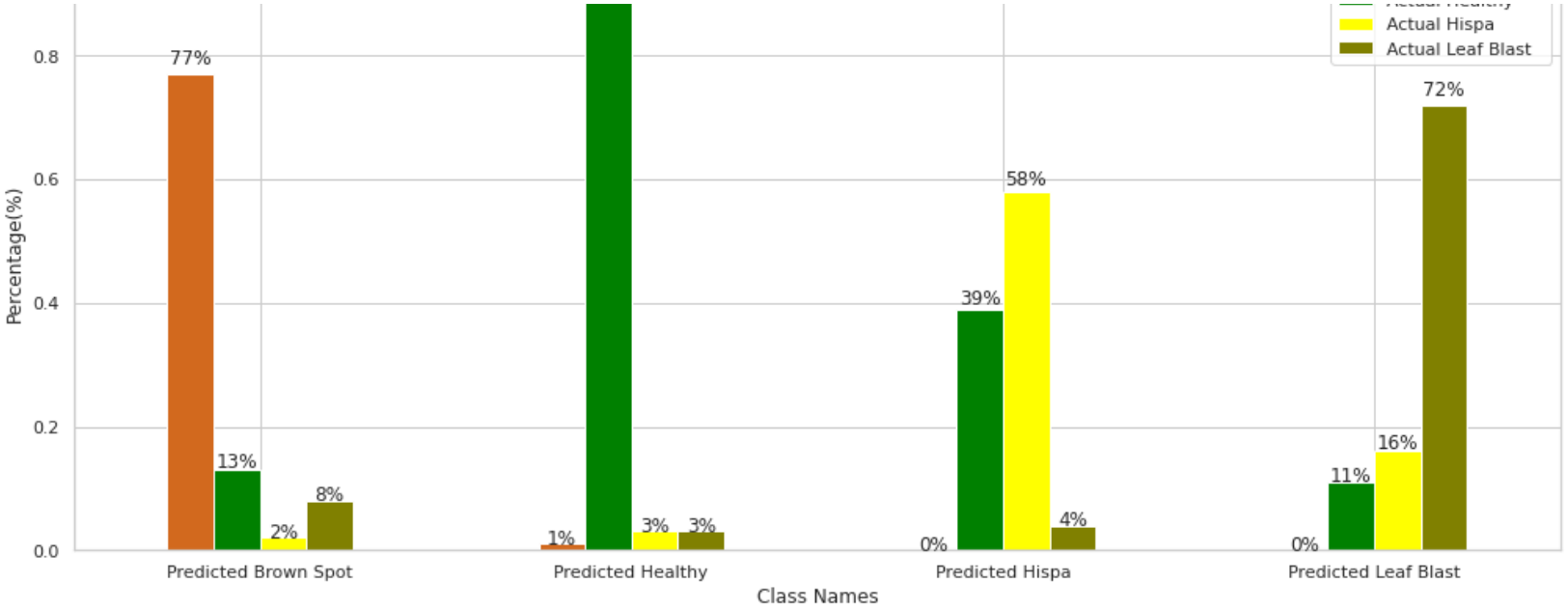
Confusion matrix of N = 339 images in test set. The confusion matrix provides an accurate view of how correctly the model predicts the classes or how the classes are misclassified. The values of the diagonal represented in green correspond to the number of correctly predicted image values for each class. The values of the matrix outside the diagonal represented in yellow correspond to incorrect predictions, where each cell relates the true class to the class predicted by the algorithm.

In [2]:
```
# Create data from the confusion matrix
df_conf_matrix = pd.DataFrame([['Predicted Brown Spot', 0.77, 0.13, 0.02, 0.08], ['Predicted Healthy', 0.01, 0.92
                               ['Predicted Hispa', 0.0, 0.39, 0.58, 0.04],['Predicted Leaf Blast', 0.0, 0.11, 0.1
                     columns=['Class Names', 'Actual Brown Spot', 'Actual Healthy', 'Actual Hispa', 'Actual Leaf Bla
# Preview data
print(df_conf_matrix)
```

```
            Class Names  Actual Brown Spot  Actual Healthy  Actual Hispa  \
0  Predicted Brown Spot               0.77            0.13          0.02
1     Predicted Healthy               0.01            0.92          0.03
2       Predicted Hispa               0.00            0.39          0.58
3  Predicted Leaf Blast               0.00            0.11          0.16

   Actual Leaf Blast
0               0.08
1               0.03
2               0.04
3               0.72
```

In [66]:
```
# set the colors
colors = ['chocolate','green', 'yellow','olive']
# plot with annotations is probably easier
ax = df_conf_matrix.plot(kind='bar', x='Class Names', color=colors, figsize=(17, 7), rot=0, ylabel='Percentage(%)
                         title="Performance of the Classification Model in Bar Chart")
# Add  loop to add the annotations
for p in ax.patches:
    width = p.get_width()
    height = p.get_height()
    x, y = p.get_xy()
    ax.annotate(f'{height:.0%}', (x + width/2, y + height*1.02), ha='center')
```





Conclusion

The advantages of an automated rice disease detection system can prove of much value to agricultural organizations and cultivators. The performances of shallow networks trained from scratch and deep models fine-tuned by transfer learning are evaluated systemically . The best model is the deep *InceptionV3* model trained with transfer learning, which yields an overall accuracy of 79% on the hold-out test set.

- *Brown Spot* - performed predictions of true values (TP) at 77%. However, it has false alarms at 23% with other three classes.
- *Healthy* - performed almost perfect job at 92% with the true values (TP). It has 6% false alarms on *hispa* and *leaf blast*. Missclassification rate is 1% with *brown spot*.
- *Leaf Blast* - Performed correct prediction with 72% which i nearly same as *brown spot*(less 5%). It missclassified 26% with *healthy* and *hispa* leaves.
- *Hispa* - performing one of the worst according to our normalized confusion matrix. It did classify 39% of the leaves as a *healthy* ones which is bad for disease prevention.

Limitations

- Lack of enough samples and unbalanced datasets. The under-representation of a particular category of rice disease results in a lack of necessary samples for the model to learn from, utilizing these sample in association with other mismatched

categories result in unbalanced datasets that can bias the classification results.

- Low quality images, and color similarity make it difficult for models to identify features properly, and thus not learn the necessary points for accurate classification.

Recommendation

Leaf Green can be used to classify two common diseases*(brown spot, leaf blast)* along with *healthy* rice leaves. We recommend to use this product as early as possible to catch disease before it spreads to the rest of the rice plants.

In terms of *hispa* disase model did strugle to classify it correctly. It may be due to the image quality. They may have a lot in common with regular healthy leaves so model classified them as a healthy ones. This class needs to be further improved with more and diverse image datasets.

Future Consideration

The performance of proposed model can be further improved with large dataset of rice diseased images along with other common disease.

Experimenting with a different algorithm and adding some context to the data may also lead to some improvments.

Based on the achieved results a mobile solution (application) can be developed for farmers and agricultural organizations to detect rice leaf diseases at hand.

Adding location data to the model would be helpful for users as some diseases are more common in certain climates.