Animal Classification With ResNet50

1. Introduction

In this notebook, we will learn how to classify images of Animals by developing ResNet 50.

In [2]:
```
! pip install split-folders
```

```
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
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/venv
```

2. Importing Libraries

In [3]:
```
import matplotlib.pyplot as plt
import numpy as np
import os
import PIL
import tensorflow as tf
import pathlib
import cv2
from keras.preprocessing.image import ImageDataGenerator
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
from keras.models import Sequential, Model,load_model
from keras.callbacks import EarlyStopping,ModelCheckpoint
from keras.layers import Input, Add, Dense, Activation, ZeroPadding2D, BatchNormalization, Flatten, Conv2D, AveragePooling2D, MaxPooling2D, GlobalMaxPooling2D,MaxPool2D
from keras.preprocessing import image
from keras.initializers import glorot_uniform
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, cohen_kappa_score, roc_auc_score, confusion_matrix
from sklearn.metrics import classification_report
from keras.layers import Input, Add, Dense, Activation, ZeroPadding2D, BatchNormalization, Flatten, Conv2D, AveragePooling2D, MaxPooling2D, GlobalMaxPooling2D,MaxPool2D,Dropout
import tensorflow as tf
import splitfolders
import pandas as pd
import glob
from sklearn.metrics import confusion_matrix
import plotly.graph_objects as go
import itertools
import plotly.express as px
#Suppressing Warnings
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'
```

2.1 Setting Path

In [4]:
```
data_dir = "../input/animals10/raw-img"
data_dir = pathlib.Path(data_dir)
```

2.2 Importing Images

In [5]:
```
Total_Images = glob.glob('../input/animals10/raw-img/*/*.jpeg')
print("Total Number of Images", len(Total_Images))
Total_Images = pd.Series(Total_Images)
```

```
Total Number of Images 24209
```

In [6]:
```
Total_Df = pd.DataFrame()

Total_Df['FileName'] = Total_Images.map(lambda ImageName :ImageName.split("H")[-1])

Total_Df['ClassId'] = Total_Images.map(lambda ImageName :ImageName.split("/")[-2])

Total_Df.head()
```

Out[6]:

	FileName	ClassId
0	aJ4.jpeg	cavallo
1	../input/animals10/raw-img/cavallo/OIP-TPYKs3X...	cavallo
2	aE2.jpeg	cavallo
3	aEK.jpeg	cavallo
4	aFj.jpeg	cavallo

In [7]:
```
Class_Id_Dist_Total = Total_Df['ClassId'].value_counts()
Class_Id_Dist_Total.head(10)
```

Out[7]:
```
cane          4863
ragno         4497
gallina       3098
cavallo       2623
mucca         1866
scoiattolo    1862
farfalla      1650
pecora        1444
gatto         1227
elefante      1079
Name: ClassId, dtype: int64
```

3. Total Data Distribution

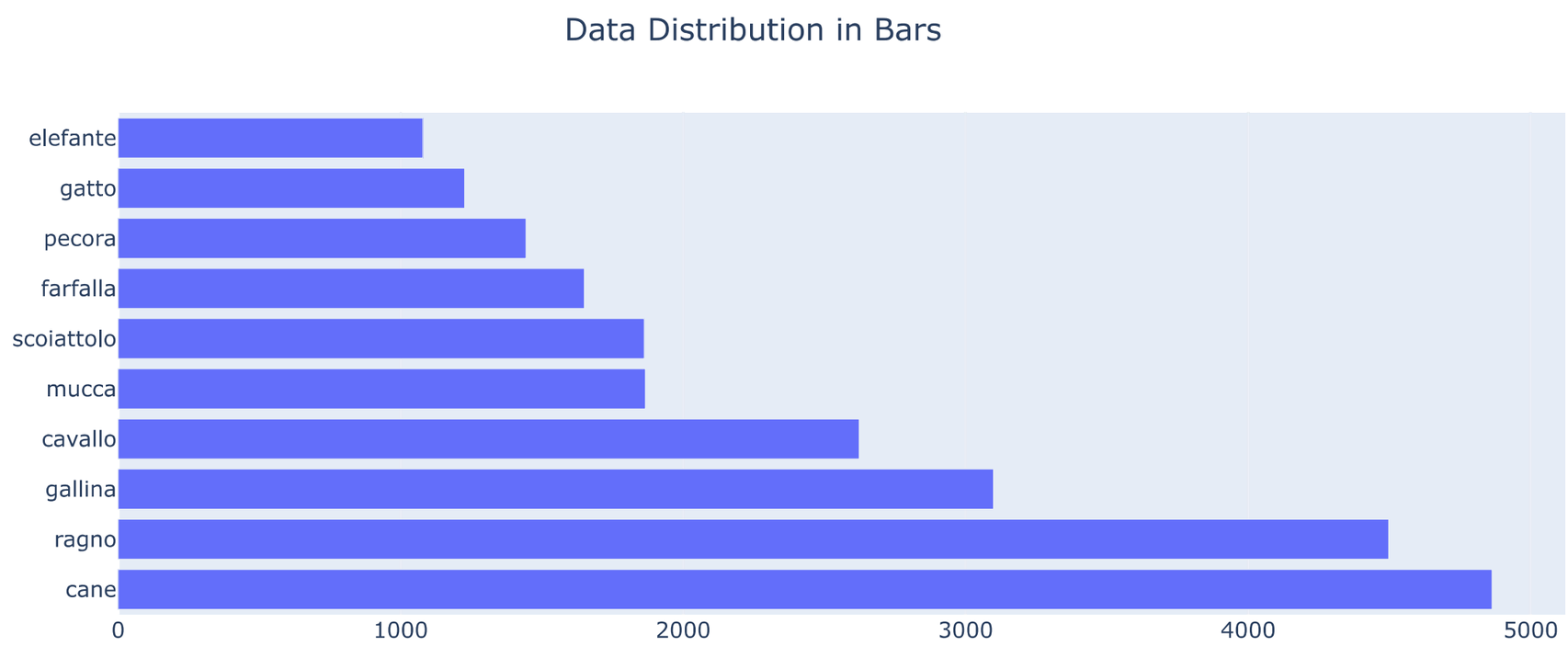
The data is not equally dsitributed.

Cavallo has the Highest Percentage of the data which is 20.1%.

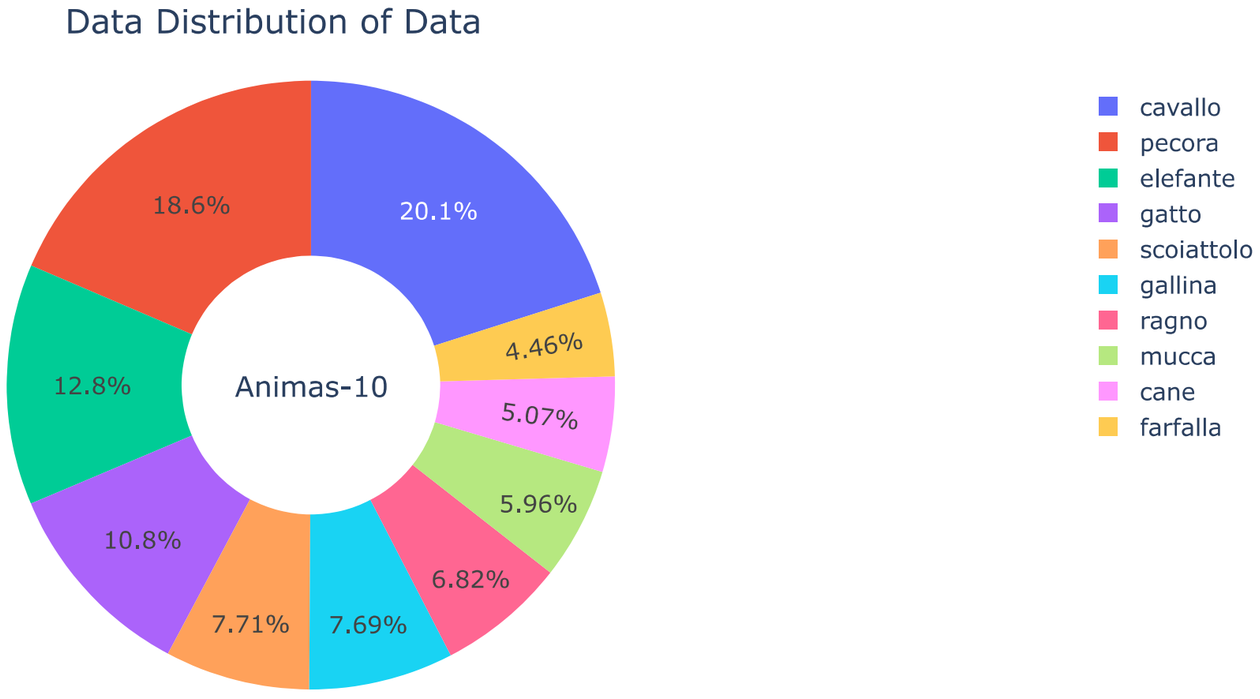
farfalla has the lowest Percentage of the data which is 4.46%.

In [8]:
```
fig = go.Figure(go.Bar(
            x= Class_Id_Dist_Total.values,
            y=Class_Id_Dist_Total.index,
            orientation='h'))

fig.update_layout(title='Data Distribution in Bars',font_size=15,title_x=0.45)
```

```
fig.show()
```



In [9]:
```
fig=px.pie(Class_Id_Dist_Total.head(10),values= 'ClassId', names=Total_Df['ClassId'].unique(),hole=0.425)
fig.update_layout(title='Data Distribution of Data',font_size=15,title_x=0.45,annotations=[dict(text='Animas-10',font_size=18, showarrow=False,height=800,width=700)])
fig.update_traces(textfont_size=15,textinfo='percent')
fig.show()
```



4.Splitting the Data into Train, Test and Val.

In [10]:
```
splitfolders.ratio(data_dir, output="output", seed=101, ratio=(.8, .1, .1))
```

```
Copying files: 26179 files [03:28, 125.78 files/s]
```

Setting Path

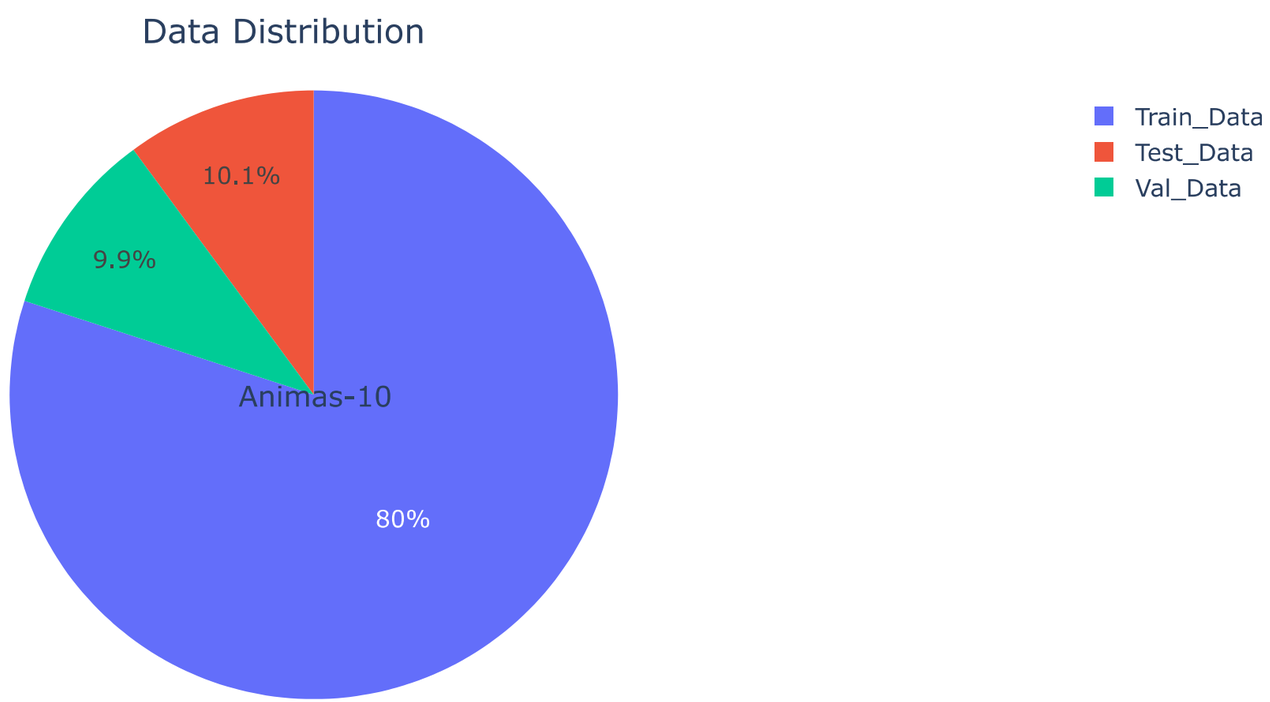
In [11]:
```
train_path='./output/train/'
val_path='./output/val'
test_path='./output/test'
class_names=os.listdir(train_path)
class_names_val=os.listdir(val_path)
class_names_test=os.listdir(test_path)
```

In [12]:
```
train_image1 = glob.glob('./output/train/*/*.jpeg')

Total_TrainImages = train_image1
print("Total number of training images: ", len(Total_TrainImages))


test_image1 = glob.glob('./output/test/*/*.jpeg')

Total_TestImages = test_image1
print("Total number of test images: ", len(Total_TestImages))
```

```
Val_image1 = glob.glob('./output/val/*/*.jpeg')

Total_ValImages = Val_image1
print("Total number of val images: ", len(Total_ValImages))
```

```
Total number of training images:  19366
Total number of test images:  2447
Total number of val images:  2396
```

In [13]:
```
random_x = [len(Total_TrainImages), len(Total_TestImages), len(Total_ValImages)]
names = ['Train_Data', 'Test_Data', 'Val_Data']
fig = px.pie(values=random_x, names=names)
fig.update_layout(title='Data Distribution',font_size=15,title_x=0.45,annotations=[dict(text='Animas-10',font_size=18, showarrow=False,height=800,width=700)])
fig.update_traces(textfont_size=15,textinfo='percent')
fig.show()
```



In [14]:
```
train_image_names = pd.Series(Total_TrainImages)
train_df = pd.DataFrame()

# generate Filename field
train_df['Filename'] = train_image_names.map( lambda img_name: img_name.split("/")[-1])


# generate ClassId field
```

```
train_df['ClassId'] = train_image_names.map(lambda img_name: img_name.split("/")[-2])

train_df.head()
```

Out[14]:

	Filename	ClassId
0	OIP-EvwSg2I2e_CHstLM2w4jwQHaFj.jpeg	pecora
1	OIP-s7t85nbWEXh64qNGmJfydAAAAA.jpeg	pecora
2	OIP-YXmplfa9d5PKoDf3pu0DQwHaE2.jpeg	pecora
3	OIP-AhVWjZy6VerGaBMc6D3j0AHaII.jpeg	pecora
4	OIP-Qe_qREKFRoEErjb1xWVWwwHaE8.jpeg	pecora

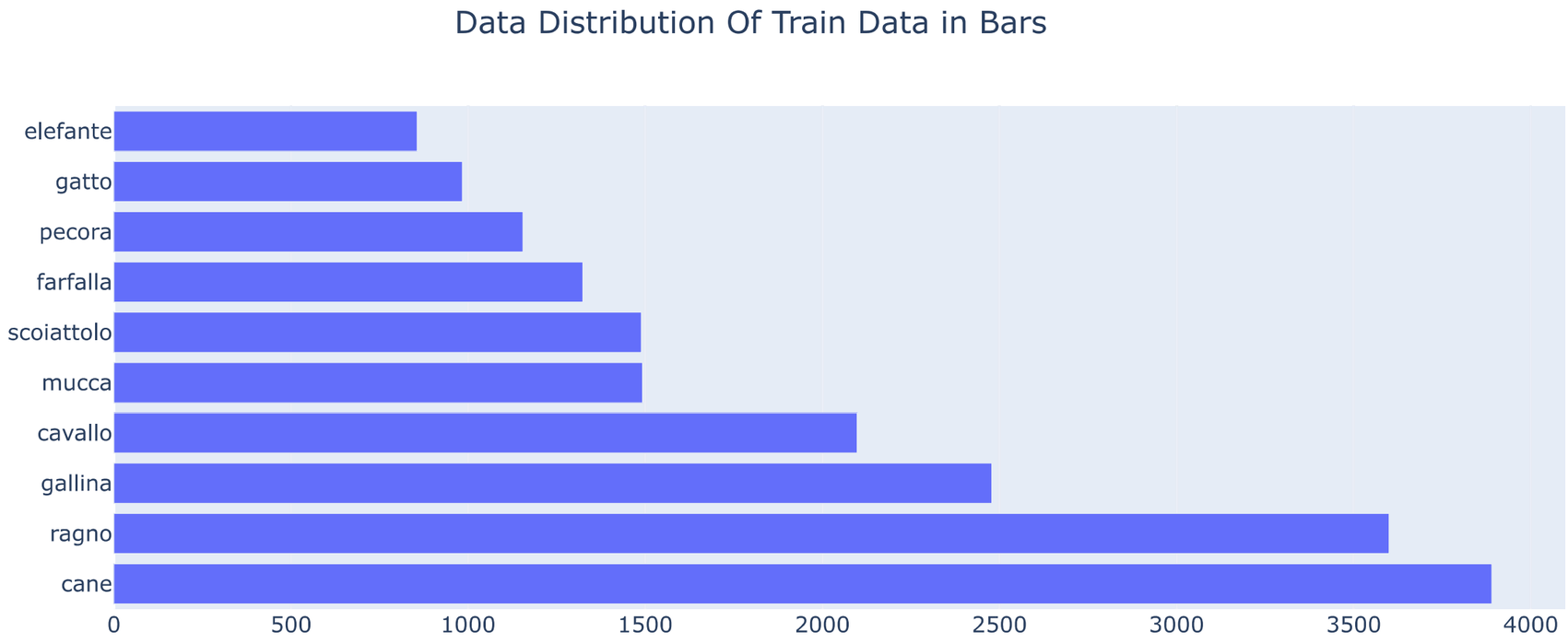
In [15]:
```
class_id_distribution_Train = train_df['ClassId'].value_counts()
class_id_distribution_Train.head(10)
```

Out[15]:
```
cane          3890
ragno         3600
gallina       2478
cavallo       2098
mucca         1492
scoiattolo    1489
farfalla      1324
pecora        1155
gatto          984
elefante       856
Name: ClassId, dtype: int64
```

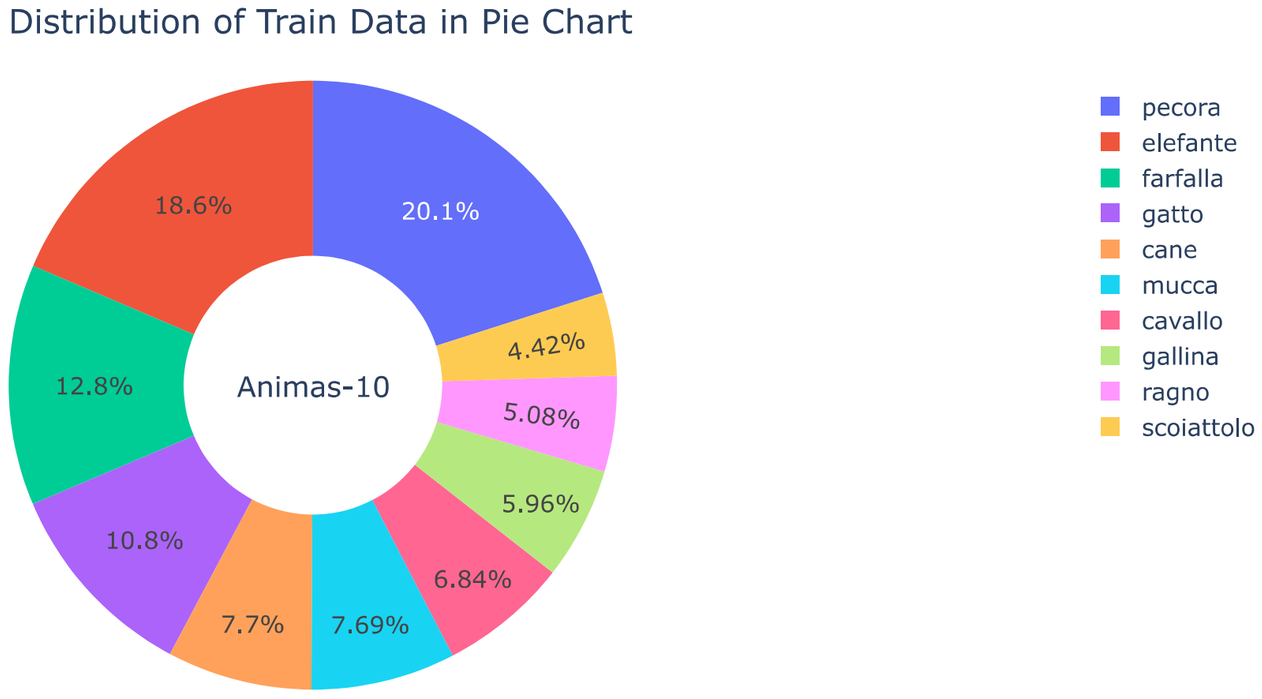
4.1 Distribution of Train Data

The data is not equally dsitributed.

In [16]:
```
fig = go.Figure(go.Bar(
            x=class_id_distribution_Train.values,
            y=class_id_distribution_Train.index,
            orientation='h'))

fig.update_layout(title='Data Distribution Of Train Data in Bars',font_size=15,title_x=0.45)

fig.show()
```



In [17]:
```
fig=px.pie(class_id_distribution_Train.head(10),values= 'ClassId', names=train_df['ClassId'].unique(),hole=0.425)
fig.update_layout(title='Data Distribution of Train Data in Pie Chart',font_size=15,title_x=0.45,annotations=[dict(text='Animas-10',font_size=18, showarrow=False,height=800,width=700)])
fig.update_traces(textfont_size=15,textinfo='percent')
fig.show()
```



4.2 Distribution Of Validation Data.

The data is not equally dsitributed.

In [18]:
```
val_image_names = pd.Series(Total_ValImages)
val_df = pd.DataFrame()

# generate Filename field
val_df['Filename'] = val_image_names.map( lambda img_name: img_name.split("/")[-1])


# generate ClassId field
val_df['ClassId'] = val_image_names.map(lambda img_name: img_name.split("/")[-2])

val_df.head()
```

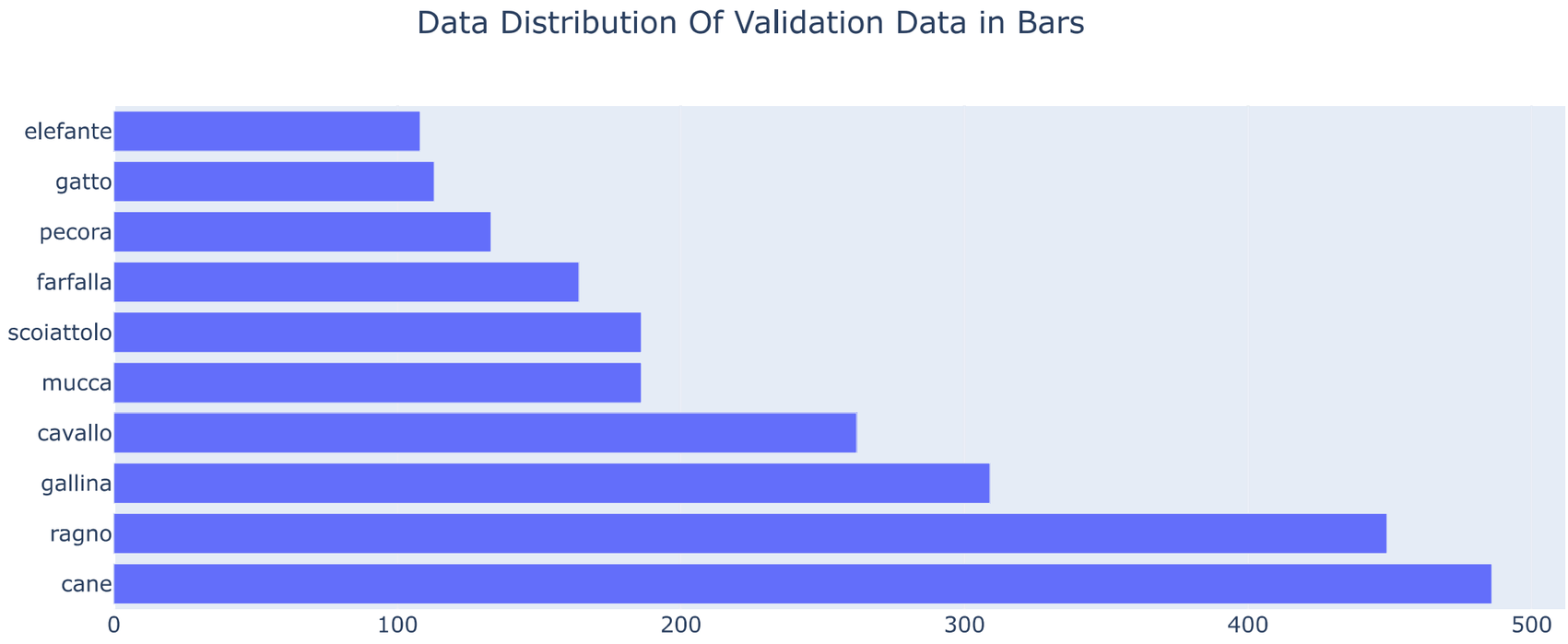
Out[18]:

	Filename	ClassId
0	OIP-7VNcWvoELVN3n57HN6T59wHaDn.jpeg	pecora
1	OIP-BiKOHe6zJu8c9ADiTJqH5gHaDf.jpeg	pecora
2	OIP-EGGcFcF2HE4nMy0WwUI8FQHaID.jpeg	pecora
3	OIP-jF53IAfoHJ_cyx_f0n5uIQHaFs.jpeg	pecora
4	OIP-x8UREud4Xv9ZLKcjGIvdGgHaFc.jpeg	pecora

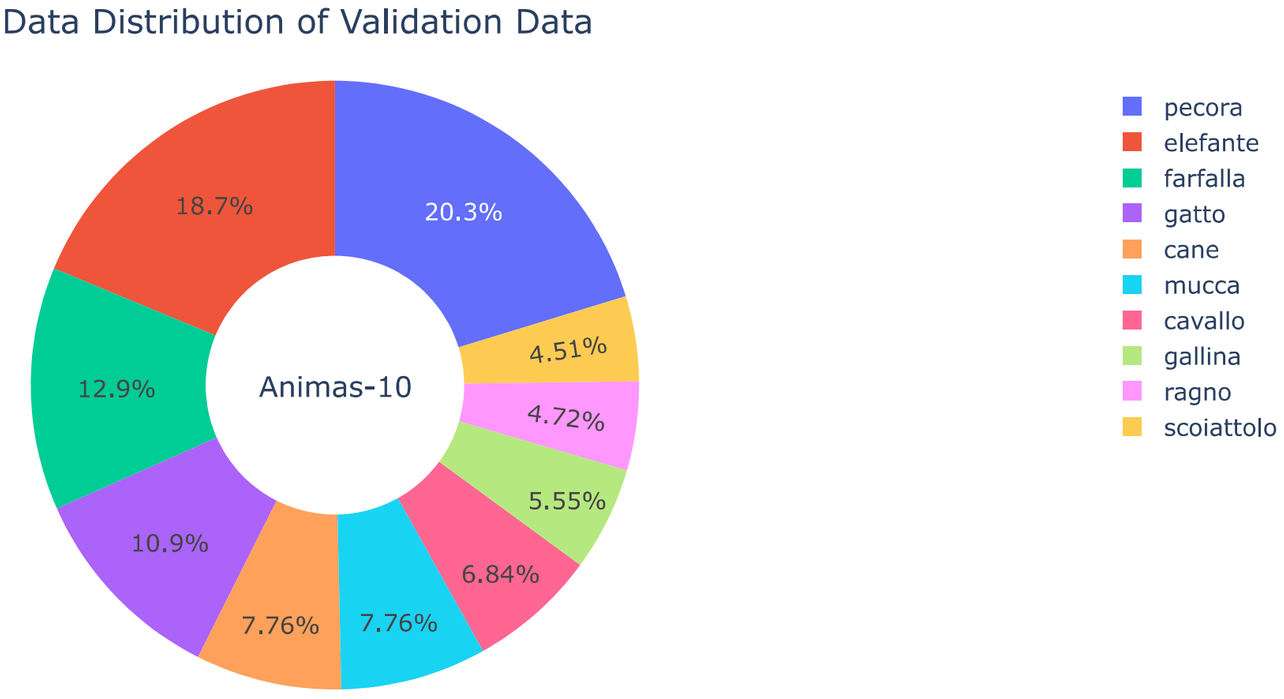
In [19]:
```
class_id_distribution_val = val_df['ClassId'].value_counts()
class_id_distribution_val.head(10)
```

Out[19]:
```
cane          486
ragno         449
gallina       309
cavallo       262
mucca         186
scoiattolo    186
farfalla      164
pecora        133
gatto         113
elefante      108
Name: ClassId, dtype: int64
```

In [20]:
```
fig = go.Figure(go.Bar(
            x=class_id_distribution_val.values,
            y=class_id_distribution_val.index,
            orientation='h'))

fig.update_layout(title='Data Distribution Of Validation Data in Bars',font_size=15,title_x=0.45)

fig.show()
```



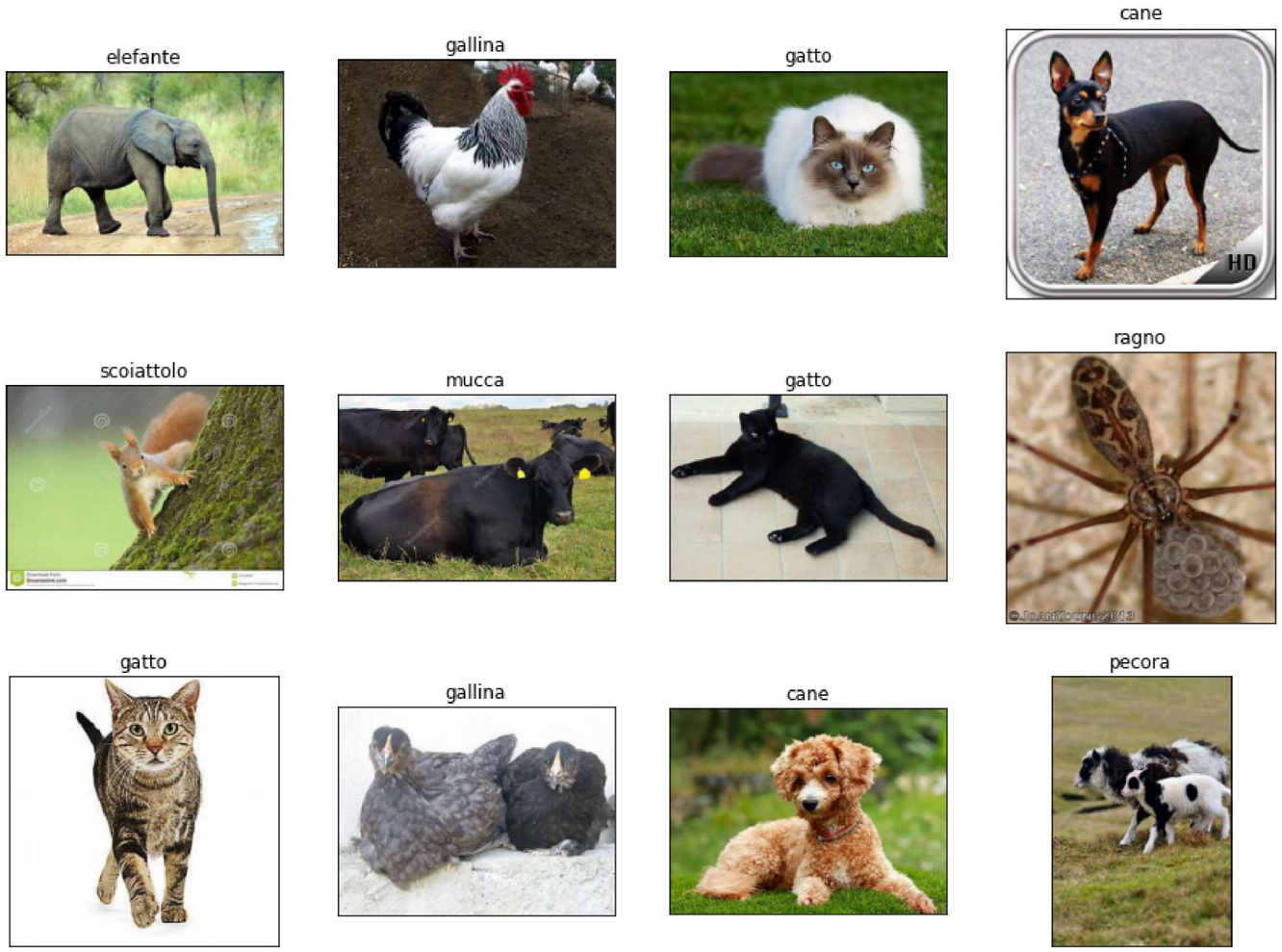
In [21]:
```
fig=px.pie(class_id_distribution_val.head(10),values= 'ClassId', names=train_df['ClassId'].unique(),hole=0.425)
fig.update_layout(title='Data Distribution of Validation Data',font_size=15,title_x=0.45,annotations=[dict(text='Animas-10',font_size=18, showarrow=False,height=800,width=700)])
fig.update_traces(textfont_size=15,textinfo='percent')
fig.show()
```



5 Displaying The Images

We are going to display Some of the Images of different Classes

In [22]:
```
plot_df = train_df.sample(12).reset_index()
plt.figure(figsize=(15, 15))

for i in range(12):
    img_name = plot_df.loc[i, 'Filename']
    label_str = (plot_df.loc[i, 'ClassId'])
    plt.subplot(4,4,i+1)
    plt.imshow(plt.imread(os.path.join(train_path,label_str, img_name)))
    plt.title(label_str)
    plt.xticks([])
    plt.yticks([])
    plt.yticks([])
```



6 Image Data Generator

Takes the path to a directory & generates batches of augmented data

shear_range = 0.1 : shear range 10%

zoom_range = 0.2 : zoom range 20%

width_shift_range = 0.1 : Randomly move the original image horizontally within 10% of the width

height_shift_range=0.1 : Randomly move the original image vertically within 10% of the width

In [23]:
```
from keras.preprocessing.image import ImageDataGenerator
train_datagen = ImageDataGenerator(zoom_range=0.15,width_shift_range=0.2,height_shift_range=0.2,shear_range=0.15)
test_datagen = ImageDataGenerator()
val_datagen = ImageDataGenerator()
train_generator = train_datagen.flow_from_directory(train_path,target_size=(224, 224),batch_size=32,shuffle=True)
```

```
test_generator = test_datagen.flow_from_directory(test_path,target_size=(224,224),batch_size=32,shuffle=False)
val_generator = val_datagen.flow_from_directory(val_path,target_size=(224,224),batch_size=32,shuffle=False)
```

```
Found 20938 images belonging to 10 classes.
Found 2627 images belonging to 10 classes.
Found 2614 images belonging to 10 classes.
```

7 ResNet50

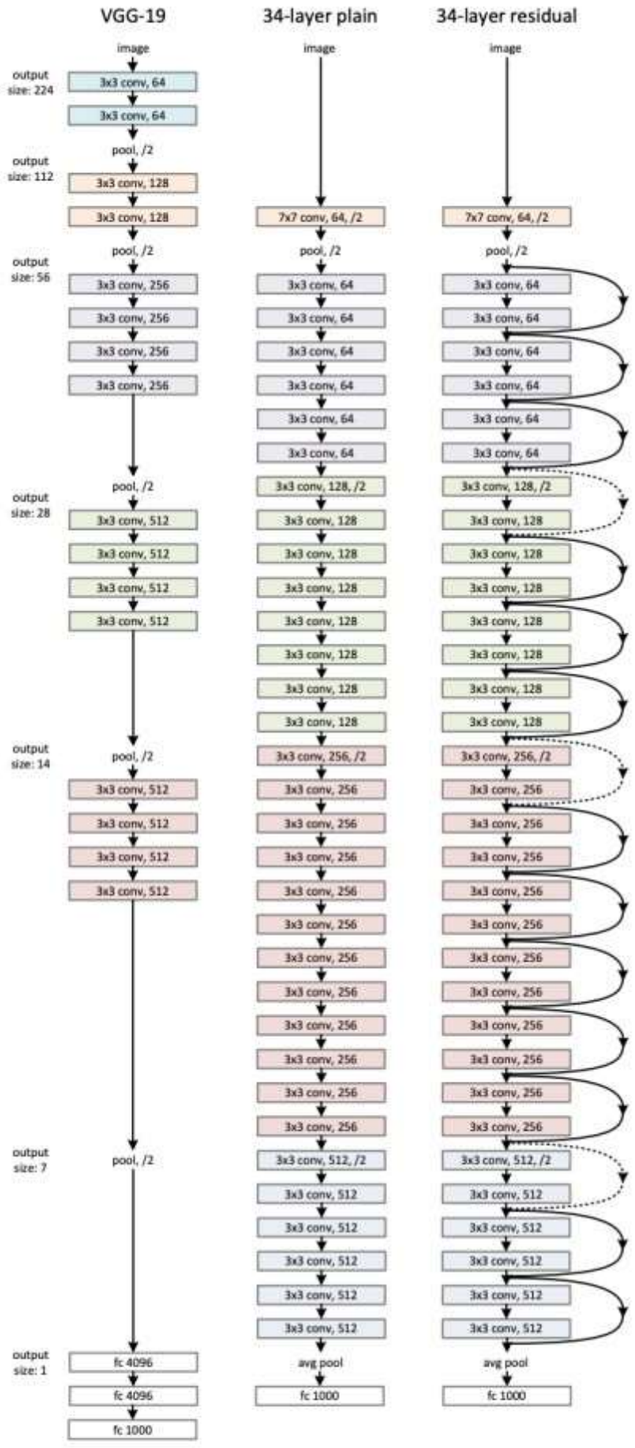


Figure 3. Example network architectures for ImageNet. **Left**: the VGG-19 model [41] (19.6 billion FLOPs) as a reference. **Mid-**

dle: a plain network with 34 parameter layers (3.6 billion FLOPs). **Right**: a residual network with 34 parameter layers (3.6 billion FLOPs). The dotted shortcuts increase dimensions. **Table 1** shows more details and other variants.

It is very important to understand about the ResNet. The advancment in the computer vison task was due to the breakthrough achivement of the ResNet architecture.

The architecture allows you to go deeper in the layers which is 150+ layers.

It is an innovative neural network that was first introduced by Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun in their 2015 computer vision research paper titled 'Deep Residual Learning for Image Recognition'.

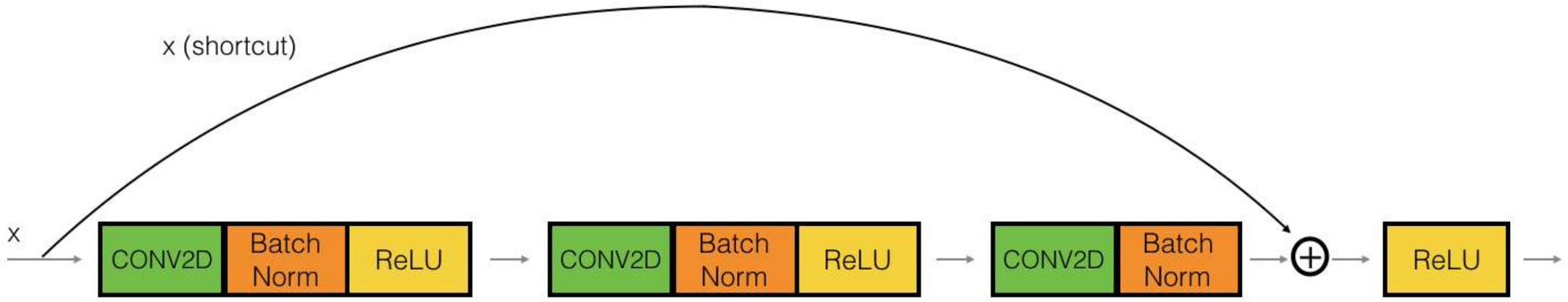
More Details about the paper can be found here Deep Residual Learning for Image Recognition

Before Resnet, In theory the more you have layers the loss value reduces and accuracy increase, but in pratically that did not happen. The more you have layers the accuracy was decreasing.

Convolutional Neural Network has the Problem of the "Vanishing Gradient Problem" During the Backpropagation the value of gradient descent decraeses and there is hardly changes in the weights. To overcome this problem Resnet Comes with the Skip Connections.

Skip Connection — Adding the original input to the output of the convolutional block.

7.1 Identity Block



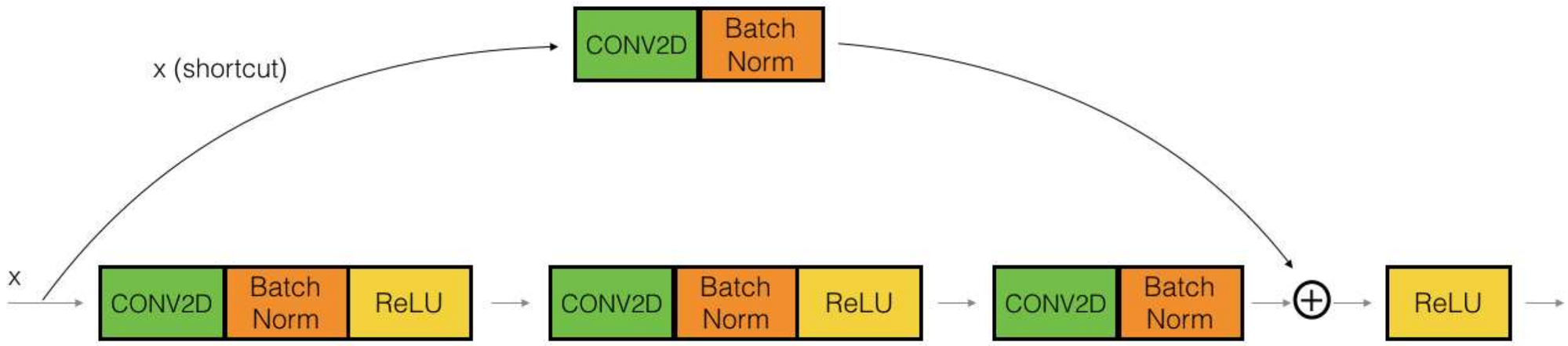
The value of 'x' is added to the output layer if and only if the

Input Size == Output Size.

In [24]:
```
def identity_block(X, f, filters, stage, block):

    conv_name_base = 'res' + str(stage) + block + '_branch'
    bn_name_base = 'bn' + str(stage) + block + '_branch'
```

```
    F1, F2, F3 = filters

    X_shortcut = X

    X = Conv2D(filters=F1, kernel_size=(1, 1), strides=(1, 1), padding='valid', name=conv_name_base + '2a', kernel_initializer=glorot_uniform(seed=0))(X)
    X = BatchNormalization(axis=3, name=bn_name_base + '2a')(X)
    X = Activation('relu')(X)

    X = Conv2D(filters=F2, kernel_size=(f, f), strides=(1, 1), padding='same', name=conv_name_base + '2b', kernel_initializer=glorot_uniform(seed=0))(X)
    X = BatchNormalization(axis=3, name=bn_name_base + '2b')(X)
    X = Activation('relu')(X)

    X = Conv2D(filters=F3, kernel_size=(1, 1), strides=(1, 1), padding='valid', name=conv_name_base + '2c', kernel_initializer=glorot_uniform(seed=0))(X)
    X = BatchNormalization(axis=3, name=bn_name_base + '2c')(X)

    X = Add()([X, X_shortcut])# SKIP Connection
    X = Activation('relu')(X)

    return X
```

7.2 Convolutional Block



if Input Size != Output Size

we add a 'convolutional block' in the shortcut path to make the input size equal to output size.

In [25]:
```
def convolutional_block(X, f, filters, stage, block, s=2):

    conv_name_base = 'res' + str(stage) + block + '_branch'
    bn_name_base = 'bn' + str(stage) + block + '_branch'

    F1, F2, F3 = filters
```

```
    X_shortcut = X

    X = Conv2D(filters=F1, kernel_size=(1, 1), strides=(s, s), padding='valid', name=conv_name_base + '2a', kernel_initializer=glorot_uniform(seed=0))(X)
    X = BatchNormalization(axis=3, name=bn_name_base + '2a')(X)
    X = Activation('relu')(X)

    X = Conv2D(filters=F2, kernel_size=(f, f), strides=(1, 1), padding='same', name=conv_name_base + '2b', kernel_initializer=glorot_uniform(seed=0))(X)
    X = BatchNormalization(axis=3, name=bn_name_base + '2b')(X)
    X = Activation('relu')(X)
    X = Conv2D(filters=F3, kernel_size=(1, 1), strides=(1, 1), padding='valid', name=conv_name_base + '2c', kernel_initializer=glorot_uniform(seed=0))(X)
    X = BatchNormalization(axis=3, name=bn_name_base + '2c')(X)

    X_shortcut = Conv2D(filters=F3, kernel_size=(1, 1), strides=(s, s), padding='valid', name=conv_name_base + '1', kernel_initializer=glorot_uniform(seed=0))(X_shortcut)
    X_shortcut = BatchNormalization(axis=3, name=bn_name_base + '1')(X_shortcut)

    X = Add()([X, X_shortcut])
    X = Activation('relu')(X)

    return X
```

7.3 Final Model

In [26]:

```
def ResNet50(input_shape=(224, 224, 3)):

    X_input = Input(input_shape)

    X = ZeroPadding2D((3, 3))(X_input)

    X = Conv2D(64, (7, 7), strides=(2, 2), name='conv1', kernel_initializer=glorot_uniform(seed=0))(X)
    X = BatchNormalization(axis=3, name='bn_conv1')(X)
    X = Activation('relu')(X)
    X = MaxPooling2D((3, 3), strides=(2, 2))(X)

    X = convolutional_block(X, f=3, filters=[64, 64, 256], stage=2, block='a', s=1)
    X = identity_block(X, 3, [64, 64, 256], stage=2, block='b')
    X = identity_block(X, 3, [64, 64, 256], stage=2, block='c')


    X = convolutional_block(X, f=3, filters=[128, 128, 512], stage=3, block='a', s=2)
    X = identity_block(X, 3, [128, 128, 512], stage=3, block='b')
    X = identity_block(X, 3, [128, 128, 512], stage=3, block='c')
    X = identity_block(X, 3, [128, 128, 512], stage=3, block='d')

    X = convolutional_block(X, f=3, filters=[256, 256, 1024], stage=4, block='a', s=2)
    X = identity_block(X, 3, [256, 256, 1024], stage=4, block='b')
    X = identity_block(X, 3, [256, 256, 1024], stage=4, block='c')
    X = identity_block(X, 3, [256, 256, 1024], stage=4, block='d')
    X = identity_block(X, 3, [256, 256, 1024], stage=4, block='e')
    X = identity_block(X, 3, [256, 256, 1024], stage=4, block='f')

    X = X = convolutional_block(X, f=3, filters=[512, 512, 2048], stage=5, block='a', s=2)
    X = identity_block(X, 3, [512, 512, 2048], stage=5, block='b')
    X = identity_block(X, 3, [512, 512, 2048], stage=5, block='c')

    X = AveragePooling2D(pool_size=(2, 2), padding='same')(X)

    model = Model(inputs=X_input, outputs=X, name='ResNet50')
```

```
    return model
```

In [27]:
```
base_model = ResNet50(input_shape=(224, 224, 3))
```

In [28]:
```
headModel = base_model.output
headModel = Flatten()(headModel)
headModel=Dense(256, activation='relu', name='fc1',kernel_initializer=glorot_uniform(seed=0))(headModel)
headModel=Dense(128, activation='relu', name='fc2',kernel_initializer=glorot_uniform(seed=0))(headModel)
headModel = Dense( 10,activation='softmax', name='fc3',kernel_initializer=glorot_uniform(seed=0))(headModel)
```

In [29]:
```
model = Model(inputs=base_model.input, outputs=headModel)
model.summary()
```

Model: "model"

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 224, 224, 3)	0	
zero_padding2d (ZeroPadding2D)	(None, 230, 230, 3)	0	input_1[0][0]
conv1 (Conv2D)	(None, 112, 112, 64)	9472	zero_padding2d[0][0]
bn_conv1 (BatchNormalization)	(None, 112, 112, 64)	256	conv1[0][0]
activation (Activation)	(None, 112, 112, 64)	0	bn_conv1[0][0]
max_pooling2d (MaxPooling2D)	(None, 55, 55, 64)	0	activation[0][0]
res2a_branch2a (Conv2D)	(None, 55, 55, 64)	4160	max_pooling2d[0][0]
bn2a_branch2a (BatchNormalizati	(None, 55, 55, 64)	256	res2a_branch2a[0][0]
activation_1 (Activation)	(None, 55, 55, 64)	0	bn2a_branch2a[0][0]
res2a_branch2b (Conv2D)	(None, 55, 55, 64)	36928	activation_1[0][0]
bn2a_branch2b (BatchNormalizati	(None, 55, 55, 64)	256	res2a_branch2b[0][0]
activation_2 (Activation)	(None, 55, 55, 64)	0	bn2a_branch2b[0][0]
res2a_branch2c (Conv2D)	(None, 55, 55, 256)	16640	activation_2[0][0]
res2a_branch1 (Conv2D)	(None, 55, 55, 256)	16640	max_pooling2d[0][0]
bn2a_branch2c (BatchNormalizati	(None, 55, 55, 256)	1024	res2a_branch2c[0][0]
bn2a_branch1 (BatchNormalizatio	(None, 55, 55, 256)	1024	res2a_branch1[0][0]
add (Add)	(None, 55, 55, 256)	0	bn2a_branch2c[0][0] bn2a_branch1[0][0]
activation_3 (Activation)	(None, 55, 55, 256)	0	add[0][0]
res2b_branch2a (Conv2D)	(None, 55, 55, 64)	16448	activation_3[0][0]
bn2b_branch2a (BatchNormalizati	(None, 55, 55, 64)	256	res2b_branch2a[0][0]
activation_4 (Activation)	(None, 55, 55, 64)	0	bn2b_branch2a[0][0]
res2b_branch2b (Conv2D)	(None, 55, 55, 64)	36928	activation_4[0][0]
bn2b_branch2b (BatchNormalizati	(None, 55, 55, 64)	256	res2b_branch2b[0][0]
activation_5 (Activation)	(None, 55, 55, 64)	0	bn2b_branch2b[0][0]
res2b_branch2c (Conv2D)	(None, 55, 55, 256)	16640	activation_5[0][0]
bn2b_branch2c (BatchNormalizati	(None, 55, 55, 256)	1024	res2b_branch2c[0][0]
add_1 (Add)	(None, 55, 55, 256)	0	bn2b_branch2c[0][0] activation_3[0][0]

activation_6 (Activation)	(None, 55, 55, 256)	0	add_1[0][0]
res2c_branch2a (Conv2D)	(None, 55, 55, 64)	16448	activation_6[0][0]
bn2c_branch2a (BatchNormalizati	(None, 55, 55, 64)	256	res2c_branch2a[0][0]
activation_7 (Activation)	(None, 55, 55, 64)	0	bn2c_branch2a[0][0]
res2c_branch2b (Conv2D)	(None, 55, 55, 64)	36928	activation_7[0][0]
bn2c_branch2b (BatchNormalizati	(None, 55, 55, 64)	256	res2c_branch2b[0][0]
activation_8 (Activation)	(None, 55, 55, 64)	0	bn2c_branch2b[0][0]
res2c_branch2c (Conv2D)	(None, 55, 55, 256)	16640	activation_8[0][0]
bn2c_branch2c (BatchNormalizati	(None, 55, 55, 256)	1024	res2c_branch2c[0][0]
add_2 (Add)	(None, 55, 55, 256)	0	bn2c_branch2c[0][0] activation_6[0][0]
activation_9 (Activation)	(None, 55, 55, 256)	0	add_2[0][0]
res3a_branch2a (Conv2D)	(None, 28, 28, 128)	32896	activation_9[0][0]
bn3a_branch2a (BatchNormalizati	(None, 28, 28, 128)	512	res3a_branch2a[0][0]
activation_10 (Activation)	(None, 28, 28, 128)	0	bn3a_branch2a[0][0]
res3a_branch2b (Conv2D)	(None, 28, 28, 128)	147584	activation_10[0][0]
bn3a_branch2b (BatchNormalizati	(None, 28, 28, 128)	512	res3a_branch2b[0][0]
activation_11 (Activation)	(None, 28, 28, 128)	0	bn3a_branch2b[0][0]
res3a_branch2c (Conv2D)	(None, 28, 28, 512)	66048	activation_11[0][0]
res3a_branch1 (Conv2D)	(None, 28, 28, 512)	131584	activation_9[0][0]
bn3a_branch2c (BatchNormalizati	(None, 28, 28, 512)	2048	res3a_branch2c[0][0]
bn3a_branch1 (BatchNormalizatio	(None, 28, 28, 512)	2048	res3a_branch1[0][0]
add_3 (Add)	(None, 28, 28, 512)	0	bn3a_branch2c[0][0] bn3a_branch1[0][0]
activation_12 (Activation)	(None, 28, 28, 512)	0	add_3[0][0]
res3b_branch2a (Conv2D)	(None, 28, 28, 128)	65664	activation_12[0][0]
bn3b_branch2a (BatchNormalizati	(None, 28, 28, 128)	512	res3b_branch2a[0][0]
activation_13 (Activation)	(None, 28, 28, 128)	0	bn3b_branch2a[0][0]
res3b_branch2b (Conv2D)	(None, 28, 28, 128)	147584	activation_13[0][0]
bn3b_branch2b (BatchNormalizati	(None, 28, 28, 128)	512	res3b_branch2b[0][0]
activation_14 (Activation)	(None, 28, 28, 128)	0	bn3b_branch2b[0][0]

res3b_branch2c (Conv2D)	(None, 28, 28, 512)	66048	activation_14[0][0]
bn3b_branch2c (BatchNormalizati	(None, 28, 28, 512)	2048	res3b_branch2c[0][0]
add_4 (Add)	(None, 28, 28, 512)	0	bn3b_branch2c[0][0] activation_12[0][0]
activation_15 (Activation)	(None, 28, 28, 512)	0	add_4[0][0]
res3c_branch2a (Conv2D)	(None, 28, 28, 128)	65664	activation_15[0][0]
bn3c_branch2a (BatchNormalizati	(None, 28, 28, 128)	512	res3c_branch2a[0][0]
activation_16 (Activation)	(None, 28, 28, 128)	0	bn3c_branch2a[0][0]
res3c_branch2b (Conv2D)	(None, 28, 28, 128)	147584	activation_16[0][0]
bn3c_branch2b (BatchNormalizati	(None, 28, 28, 128)	512	res3c_branch2b[0][0]
activation_17 (Activation)	(None, 28, 28, 128)	0	bn3c_branch2b[0][0]
res3c_branch2c (Conv2D)	(None, 28, 28, 512)	66048	activation_17[0][0]
bn3c_branch2c (BatchNormalizati	(None, 28, 28, 512)	2048	res3c_branch2c[0][0]
add_5 (Add)	(None, 28, 28, 512)	0	bn3c_branch2c[0][0] activation_15[0][0]
activation_18 (Activation)	(None, 28, 28, 512)	0	add_5[0][0]
res3d_branch2a (Conv2D)	(None, 28, 28, 128)	65664	activation_18[0][0]
bn3d_branch2a (BatchNormalizati	(None, 28, 28, 128)	512	res3d_branch2a[0][0]
activation_19 (Activation)	(None, 28, 28, 128)	0	bn3d_branch2a[0][0]
res3d_branch2b (Conv2D)	(None, 28, 28, 128)	147584	activation_19[0][0]
bn3d_branch2b (BatchNormalizati	(None, 28, 28, 128)	512	res3d_branch2b[0][0]
activation_20 (Activation)	(None, 28, 28, 128)	0	bn3d_branch2b[0][0]
res3d_branch2c (Conv2D)	(None, 28, 28, 512)	66048	activation_20[0][0]
bn3d_branch2c (BatchNormalizati	(None, 28, 28, 512)	2048	res3d_branch2c[0][0]
add_6 (Add)	(None, 28, 28, 512)	0	bn3d_branch2c[0][0] activation_18[0][0]
activation_21 (Activation)	(None, 28, 28, 512)	0	add_6[0][0]
res4a_branch2a (Conv2D)	(None, 14, 14, 256)	131328	activation_21[0][0]
bn4a_branch2a (BatchNormalizati	(None, 14, 14, 256)	1024	res4a_branch2a[0][0]
activation_22 (Activation)	(None, 14, 14, 256)	0	bn4a_branch2a[0][0]
res4a_branch2b (Conv2D)	(None, 14, 14, 256)	590080	activation_22[0][0]
bn4a_branch2b (BatchNormalizati	(None, 14, 14, 256)	1024	res4a_branch2b[0][0]

activation_23 (Activation)	(None, 14, 14, 256)	0	bn4a_branch2b[0][0]
res4a_branch2c (Conv2D)	(None, 14, 14, 1024)	263168	activation_23[0][0]
res4a_branch1 (Conv2D)	(None, 14, 14, 1024)	525312	activation_21[0][0]
bn4a_branch2c (BatchNormalizati	(None, 14, 14, 1024)	4096	res4a_branch2c[0][0]
bn4a_branch1 (BatchNormalizatio	(None, 14, 14, 1024)	4096	res4a_branch1[0][0]
add_7 (Add)	(None, 14, 14, 1024)	0	bn4a_branch2c[0][0] bn4a_branch1[0][0]
activation_24 (Activation)	(None, 14, 14, 1024)	0	add_7[0][0]
res4b_branch2a (Conv2D)	(None, 14, 14, 256)	262400	activation_24[0][0]
bn4b_branch2a (BatchNormalizati	(None, 14, 14, 256)	1024	res4b_branch2a[0][0]
activation_25 (Activation)	(None, 14, 14, 256)	0	bn4b_branch2a[0][0]
res4b_branch2b (Conv2D)	(None, 14, 14, 256)	590080	activation_25[0][0]
bn4b_branch2b (BatchNormalizati	(None, 14, 14, 256)	1024	res4b_branch2b[0][0]
activation_26 (Activation)	(None, 14, 14, 256)	0	bn4b_branch2b[0][0]
res4b_branch2c (Conv2D)	(None, 14, 14, 1024)	263168	activation_26[0][0]
bn4b_branch2c (BatchNormalizati	(None, 14, 14, 1024)	4096	res4b_branch2c[0][0]
add_8 (Add)	(None, 14, 14, 1024)	0	bn4b_branch2c[0][0] activation_24[0][0]
activation_27 (Activation)	(None, 14, 14, 1024)	0	add_8[0][0]
res4c_branch2a (Conv2D)	(None, 14, 14, 256)	262400	activation_27[0][0]
bn4c_branch2a (BatchNormalizati	(None, 14, 14, 256)	1024	res4c_branch2a[0][0]
activation_28 (Activation)	(None, 14, 14, 256)	0	bn4c_branch2a[0][0]
res4c_branch2b (Conv2D)	(None, 14, 14, 256)	590080	activation_28[0][0]
bn4c_branch2b (BatchNormalizati	(None, 14, 14, 256)	1024	res4c_branch2b[0][0]
activation_29 (Activation)	(None, 14, 14, 256)	0	bn4c_branch2b[0][0]
res4c_branch2c (Conv2D)	(None, 14, 14, 1024)	263168	activation_29[0][0]
bn4c_branch2c (BatchNormalizati	(None, 14, 14, 1024)	4096	res4c_branch2c[0][0]
add_9 (Add)	(None, 14, 14, 1024)	0	bn4c_branch2c[0][0] activation_27[0][0]
activation_30 (Activation)	(None, 14, 14, 1024)	0	add_9[0][0]
res4d_branch2a (Conv2D)	(None, 14, 14, 256)	262400	activation_30[0][0]

bn4d_branch2a (BatchNormalizati	(None, 14, 14, 256)	1024	res4d_branch2a[0][0]
activation_31 (Activation)	(None, 14, 14, 256)	0	bn4d_branch2a[0][0]
res4d_branch2b (Conv2D)	(None, 14, 14, 256)	590080	activation_31[0][0]
bn4d_branch2b (BatchNormalizati	(None, 14, 14, 256)	1024	res4d_branch2b[0][0]
activation_32 (Activation)	(None, 14, 14, 256)	0	bn4d_branch2b[0][0]
res4d_branch2c (Conv2D)	(None, 14, 14, 1024)	263168	activation_32[0][0]
bn4d_branch2c (BatchNormalizati	(None, 14, 14, 1024)	4096	res4d_branch2c[0][0]
add_10 (Add)	(None, 14, 14, 1024)	0	bn4d_branch2c[0][0] activation_30[0][0]
activation_33 (Activation)	(None, 14, 14, 1024)	0	add_10[0][0]
res4e_branch2a (Conv2D)	(None, 14, 14, 256)	262400	activation_33[0][0]
bn4e_branch2a (BatchNormalizati	(None, 14, 14, 256)	1024	res4e_branch2a[0][0]
activation_34 (Activation)	(None, 14, 14, 256)	0	bn4e_branch2a[0][0]
res4e_branch2b (Conv2D)	(None, 14, 14, 256)	590080	activation_34[0][0]
bn4e_branch2b (BatchNormalizati	(None, 14, 14, 256)	1024	res4e_branch2b[0][0]
activation_35 (Activation)	(None, 14, 14, 256)	0	bn4e_branch2b[0][0]
res4e_branch2c (Conv2D)	(None, 14, 14, 1024)	263168	activation_35[0][0]
bn4e_branch2c (BatchNormalizati	(None, 14, 14, 1024)	4096	res4e_branch2c[0][0]
add_11 (Add)	(None, 14, 14, 1024)	0	bn4e_branch2c[0][0] activation_33[0][0]
activation_36 (Activation)	(None, 14, 14, 1024)	0	add_11[0][0]
res4f_branch2a (Conv2D)	(None, 14, 14, 256)	262400	activation_36[0][0]
bn4f_branch2a (BatchNormalizati	(None, 14, 14, 256)	1024	res4f_branch2a[0][0]
activation_37 (Activation)	(None, 14, 14, 256)	0	bn4f_branch2a[0][0]
res4f_branch2b (Conv2D)	(None, 14, 14, 256)	590080	activation_37[0][0]
bn4f_branch2b (BatchNormalizati	(None, 14, 14, 256)	1024	res4f_branch2b[0][0]
activation_38 (Activation)	(None, 14, 14, 256)	0	bn4f_branch2b[0][0]
res4f_branch2c (Conv2D)	(None, 14, 14, 1024)	263168	activation_38[0][0]
bn4f_branch2c (BatchNormalizati	(None, 14, 14, 1024)	4096	res4f_branch2c[0][0]
add_12 (Add)	(None, 14, 14, 1024)	0	bn4f_branch2c[0][0] activation_36[0][0]
activation_39 (Activation)	(None, 14, 14, 1024)	0	add_12[0][0]

res5a_branch2a (Conv2D)	(None, 7, 7, 512)	524800	activation_39[0][0]
bn5a_branch2a (BatchNormalizati	(None, 7, 7, 512)	2048	res5a_branch2a[0][0]
activation_40 (Activation)	(None, 7, 7, 512)	0	bn5a_branch2a[0][0]
res5a_branch2b (Conv2D)	(None, 7, 7, 512)	2359808	activation_40[0][0]
bn5a_branch2b (BatchNormalizati	(None, 7, 7, 512)	2048	res5a_branch2b[0][0]
activation_41 (Activation)	(None, 7, 7, 512)	0	bn5a_branch2b[0][0]
res5a_branch2c (Conv2D)	(None, 7, 7, 2048)	1050624	activation_41[0][0]
res5a_branch1 (Conv2D)	(None, 7, 7, 2048)	2099200	activation_39[0][0]
bn5a_branch2c (BatchNormalizati	(None, 7, 7, 2048)	8192	res5a_branch2c[0][0]
bn5a_branch1 (BatchNormalizatio	(None, 7, 7, 2048)	8192	res5a_branch1[0][0]
add_13 (Add)	(None, 7, 7, 2048)	0	bn5a_branch2c[0][0] bn5a_branch1[0][0]
activation_42 (Activation)	(None, 7, 7, 2048)	0	add_13[0][0]
res5b_branch2a (Conv2D)	(None, 7, 7, 512)	1049088	activation_42[0][0]
bn5b_branch2a (BatchNormalizati	(None, 7, 7, 512)	2048	res5b_branch2a[0][0]
activation_43 (Activation)	(None, 7, 7, 512)	0	bn5b_branch2a[0][0]
res5b_branch2b (Conv2D)	(None, 7, 7, 512)	2359808	activation_43[0][0]
bn5b_branch2b (BatchNormalizati	(None, 7, 7, 512)	2048	res5b_branch2b[0][0]
activation_44 (Activation)	(None, 7, 7, 512)	0	bn5b_branch2b[0][0]
res5b_branch2c (Conv2D)	(None, 7, 7, 2048)	1050624	activation_44[0][0]
bn5b_branch2c (BatchNormalizati	(None, 7, 7, 2048)	8192	res5b_branch2c[0][0]
add_14 (Add)	(None, 7, 7, 2048)	0	bn5b_branch2c[0][0] activation_42[0][0]
activation_45 (Activation)	(None, 7, 7, 2048)	0	add_14[0][0]
res5c_branch2a (Conv2D)	(None, 7, 7, 512)	1049088	activation_45[0][0]
bn5c_branch2a (BatchNormalizati	(None, 7, 7, 512)	2048	res5c_branch2a[0][0]
activation_46 (Activation)	(None, 7, 7, 512)	0	bn5c_branch2a[0][0]
res5c_branch2b (Conv2D)	(None, 7, 7, 512)	2359808	activation_46[0][0]
bn5c_branch2b (BatchNormalizati	(None, 7, 7, 512)	2048	res5c_branch2b[0][0]
activation_47 (Activation)	(None, 7, 7, 512)	0	bn5c_branch2b[0][0]
res5c_branch2c (Conv2D)	(None, 7, 7, 2048)	1050624	activation_47[0][0]

bn5c_branch2c (BatchNormalizati	(None, 7, 7, 2048)	8192	res5c_branch2c[0][0]
add_15 (Add)	(None, 7, 7, 2048)	0	bn5c_branch2c[0][0] activation_45[0][0]
activation_48 (Activation)	(None, 7, 7, 2048)	0	add_15[0][0]
average_pooling2d (AveragePooli	(None, 4, 4, 2048)	0	activation_48[0][0]
flatten (Flatten)	(None, 32768)	0	average_pooling2d[0][0]
fc1 (Dense)	(None, 256)	8388864	flatten[0][0]
fc2 (Dense)	(None, 128)	32896	fc1[0][0]
fc3 (Dense)	(None, 10)	1290	fc2[0][0]

```
==================================================================================================
Total params: 32,010,762
Trainable params: 31,957,642
Non-trainable params: 53,120
```

From Here you can compile the model and then you can train your images from scratch or you can load the weights . To do this Do the Follow steps

Step 1 "base_model.load_weights("Path to file")" .

Step 2 "for layer in base_model.layers: layer.trainable = False" .

Step 3 "model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=["accuracy"])" .

Run the Model .

Since the weights file is not uploaded , you can do it on your colab by perfoming all these steps .

I will Import the models so this way you can also have the idea of how to import model .

8 Transfer Learning

A pre-trained model is a saved network that was previously trained on a large dataset, typically on a large-scale image-classification task such as Image Net Data set. The weights of the pre trained model can be ustilse for the classification of other task. You don't have to train your model from scratch.

The intuition behind transfer learning for image classification is that if a model is trained on a large and general enough dataset, this model will effectively serve as a generic model and can be use for other different task of classification. The feature map of Pre Trained Model can be used..

In [30]:
```
from tensorflow.keras.applications import ResNet50

modelT = ResNet50(
      input_shape = (224,224,3),
      include_top = False,
      weights = 'imagenet'
    )
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5
94773248/94765736 [==============================] - 0s 0us/step
94781440/94765736 [==============================] - 0s 0us/step
```

In [31]:
```
for layers in modelT.layers:
    layers.trainable = False
```

In [32]:
```
from keras.layers import  Dropout
y = Flatten()(modelT.output)
y = Dropout(0.5)(y)
y = Dense(10, activation = "softmax")(y)

modelT = keras.Model(modelT.input, y)
modelT.compile(loss = "categorical_crossentropy", optimizer = "adam", metrics = "accuracy")
modelT.summary()
```

Model: "model_1"

Layer (type)	Output Shape	Param #	Connected to
input_2 (InputLayer)	[(None, 224, 224, 3)	0	
conv1_pad (ZeroPadding2D)	(None, 230, 230, 3)	0	input_2[0][0]
conv1_conv (Conv2D)	(None, 112, 112, 64)	9472	conv1_pad[0][0]
conv1_bn (BatchNormalization)	(None, 112, 112, 64)	256	conv1_conv[0][0]
conv1_relu (Activation)	(None, 112, 112, 64)	0	conv1_bn[0][0]
pool1_pad (ZeroPadding2D)	(None, 114, 114, 64)	0	conv1_relu[0][0]
pool1_pool (MaxPooling2D)	(None, 56, 56, 64)	0	pool1_pad[0][0]
conv2_block1_1_conv (Conv2D)	(None, 56, 56, 64)	4160	pool1_pool[0][0]
conv2_block1_1_bn (BatchNormali	(None, 56, 56, 64)	256	conv2_block1_1_conv[0][0]
conv2_block1_1_relu (Activation	(None, 56, 56, 64)	0	conv2_block1_1_bn[0][0]
conv2_block1_2_conv (Conv2D)	(None, 56, 56, 64)	36928	conv2_block1_1_relu[0][0]
conv2_block1_2_bn (BatchNormali	(None, 56, 56, 64)	256	conv2_block1_2_conv[0][0]
conv2_block1_2_relu (Activation	(None, 56, 56, 64)	0	conv2_block1_2_bn[0][0]
conv2_block1_0_conv (Conv2D)	(None, 56, 56, 256)	16640	pool1_pool[0][0]
conv2_block1_3_conv (Conv2D)	(None, 56, 56, 256)	16640	conv2_block1_2_relu[0][0]
conv2_block1_0_bn (BatchNormali	(None, 56, 56, 256)	1024	conv2_block1_0_conv[0][0]
conv2_block1_3_bn (BatchNormali	(None, 56, 56, 256)	1024	conv2_block1_3_conv[0][0]
conv2_block1_add (Add)	(None, 56, 56, 256)	0	conv2_block1_0_bn[0][0] conv2_block1_3_bn[0][0]
conv2_block1_out (Activation)	(None, 56, 56, 256)	0	conv2_block1_add[0][0]
conv2_block2_1_conv (Conv2D)	(None, 56, 56, 64)	16448	conv2_block1_out[0][0]
conv2_block2_1_bn (BatchNormali	(None, 56, 56, 64)	256	conv2_block2_1_conv[0][0]
conv2_block2_1_relu (Activation	(None, 56, 56, 64)	0	conv2_block2_1_bn[0][0]
conv2_block2_2_conv (Conv2D)	(None, 56, 56, 64)	36928	conv2_block2_1_relu[0][0]
conv2_block2_2_bn (BatchNormali	(None, 56, 56, 64)	256	conv2_block2_2_conv[0][0]
conv2_block2_2_relu (Activation	(None, 56, 56, 64)	0	conv2_block2_2_bn[0][0]
conv2_block2_3_conv (Conv2D)	(None, 56, 56, 256)	16640	conv2_block2_2_relu[0][0]
conv2_block2_3_bn (BatchNormali	(None, 56, 56, 256)	1024	conv2_block2_3_conv[0][0]
conv2_block2_add (Add)	(None, 56, 56, 256)	0	conv2_block1_out[0][0]

			conv2_block2_3_bn[0][0]
conv2_block2_out (Activation)	(None, 56, 56, 256)	0	conv2_block2_add[0][0]
conv2_block3_1_conv (Conv2D)	(None, 56, 56, 64)	16448	conv2_block2_out[0][0]
conv2_block3_1_bn (BatchNormali	(None, 56, 56, 64)	256	conv2_block3_1_conv[0][0]
conv2_block3_1_relu (Activation	(None, 56, 56, 64)	0	conv2_block3_1_bn[0][0]
conv2_block3_2_conv (Conv2D)	(None, 56, 56, 64)	36928	conv2_block3_1_relu[0][0]
conv2_block3_2_bn (BatchNormali	(None, 56, 56, 64)	256	conv2_block3_2_conv[0][0]
conv2_block3_2_relu (Activation	(None, 56, 56, 64)	0	conv2_block3_2_bn[0][0]
conv2_block3_3_conv (Conv2D)	(None, 56, 56, 256)	16640	conv2_block3_2_relu[0][0]
conv2_block3_3_bn (BatchNormali	(None, 56, 56, 256)	1024	conv2_block3_3_conv[0][0]
conv2_block3_add (Add)	(None, 56, 56, 256)	0	conv2_block2_out[0][0] conv2_block3_3_bn[0][0]
conv2_block3_out (Activation)	(None, 56, 56, 256)	0	conv2_block3_add[0][0]
conv3_block1_1_conv (Conv2D)	(None, 28, 28, 128)	32896	conv2_block3_out[0][0]
conv3_block1_1_bn (BatchNormali	(None, 28, 28, 128)	512	conv3_block1_1_conv[0][0]
conv3_block1_1_relu (Activation	(None, 28, 28, 128)	0	conv3_block1_1_bn[0][0]
conv3_block1_2_conv (Conv2D)	(None, 28, 28, 128)	147584	conv3_block1_1_relu[0][0]
conv3_block1_2_bn (BatchNormali	(None, 28, 28, 128)	512	conv3_block1_2_conv[0][0]
conv3_block1_2_relu (Activation	(None, 28, 28, 128)	0	conv3_block1_2_bn[0][0]
conv3_block1_0_conv (Conv2D)	(None, 28, 28, 512)	131584	conv2_block3_out[0][0]
conv3_block1_3_conv (Conv2D)	(None, 28, 28, 512)	66048	conv3_block1_2_relu[0][0]
conv3_block1_0_bn (BatchNormali	(None, 28, 28, 512)	2048	conv3_block1_0_conv[0][0]
conv3_block1_3_bn (BatchNormali	(None, 28, 28, 512)	2048	conv3_block1_3_conv[0][0]
conv3_block1_add (Add)	(None, 28, 28, 512)	0	conv3_block1_0_bn[0][0] conv3_block1_3_bn[0][0]
conv3_block1_out (Activation)	(None, 28, 28, 512)	0	conv3_block1_add[0][0]
conv3_block2_1_conv (Conv2D)	(None, 28, 28, 128)	65664	conv3_block1_out[0][0]
conv3_block2_1_bn (BatchNormali	(None, 28, 28, 128)	512	conv3_block2_1_conv[0][0]
conv3_block2_1_relu (Activation	(None, 28, 28, 128)	0	conv3_block2_1_bn[0][0]
conv3_block2_2_conv (Conv2D)	(None, 28, 28, 128)	147584	conv3_block2_1_relu[0][0]
conv3_block2_2_bn (BatchNormali	(None, 28, 28, 128)	512	conv3_block2_2_conv[0][0]

conv3_block2_2_relu (Activation	(None, 28, 28, 128)	0	conv3_block2_2_bn[0][0]
conv3_block2_3_conv (Conv2D)	(None, 28, 28, 512)	66048	conv3_block2_2_relu[0][0]
conv3_block2_3_bn (BatchNormali	(None, 28, 28, 512)	2048	conv3_block2_3_conv[0][0]
conv3_block2_add (Add)	(None, 28, 28, 512)	0	conv3_block1_out[0][0] conv3_block2_3_bn[0][0]
conv3_block2_out (Activation)	(None, 28, 28, 512)	0	conv3_block2_add[0][0]
conv3_block3_1_conv (Conv2D)	(None, 28, 28, 128)	65664	conv3_block2_out[0][0]
conv3_block3_1_bn (BatchNormali	(None, 28, 28, 128)	512	conv3_block3_1_conv[0][0]
conv3_block3_1_relu (Activation	(None, 28, 28, 128)	0	conv3_block3_1_bn[0][0]
conv3_block3_2_conv (Conv2D)	(None, 28, 28, 128)	147584	conv3_block3_1_relu[0][0]
conv3_block3_2_bn (BatchNormali	(None, 28, 28, 128)	512	conv3_block3_2_conv[0][0]
conv3_block3_2_relu (Activation	(None, 28, 28, 128)	0	conv3_block3_2_bn[0][0]
conv3_block3_3_conv (Conv2D)	(None, 28, 28, 512)	66048	conv3_block3_2_relu[0][0]
conv3_block3_3_bn (BatchNormali	(None, 28, 28, 512)	2048	conv3_block3_3_conv[0][0]
conv3_block3_add (Add)	(None, 28, 28, 512)	0	conv3_block2_out[0][0] conv3_block3_3_bn[0][0]
conv3_block3_out (Activation)	(None, 28, 28, 512)	0	conv3_block3_add[0][0]
conv3_block4_1_conv (Conv2D)	(None, 28, 28, 128)	65664	conv3_block3_out[0][0]
conv3_block4_1_bn (BatchNormali	(None, 28, 28, 128)	512	conv3_block4_1_conv[0][0]
conv3_block4_1_relu (Activation	(None, 28, 28, 128)	0	conv3_block4_1_bn[0][0]
conv3_block4_2_conv (Conv2D)	(None, 28, 28, 128)	147584	conv3_block4_1_relu[0][0]
conv3_block4_2_bn (BatchNormali	(None, 28, 28, 128)	512	conv3_block4_2_conv[0][0]
conv3_block4_2_relu (Activation	(None, 28, 28, 128)	0	conv3_block4_2_bn[0][0]
conv3_block4_3_conv (Conv2D)	(None, 28, 28, 512)	66048	conv3_block4_2_relu[0][0]
conv3_block4_3_bn (BatchNormali	(None, 28, 28, 512)	2048	conv3_block4_3_conv[0][0]
conv3_block4_add (Add)	(None, 28, 28, 512)	0	conv3_block3_out[0][0] conv3_block4_3_bn[0][0]
conv3_block4_out (Activation)	(None, 28, 28, 512)	0	conv3_block4_add[0][0]
conv4_block1_1_conv (Conv2D)	(None, 14, 14, 256)	131328	conv3_block4_out[0][0]
conv4_block1_1_bn (BatchNormali	(None, 14, 14, 256)	1024	conv4_block1_1_conv[0][0]
conv4_block1_1_relu (Activation	(None, 14, 14, 256)	0	conv4_block1_1_bn[0][0]
conv4_block1_2_conv (Conv2D)	(None, 14, 14, 256)	590080	conv4_block1_1_relu[0][0]

conv4_block1_2_bn (BatchNormali	(None, 14, 14, 256)	1024	conv4_block1_2_conv[0][0]
conv4_block1_2_relu (Activation	(None, 14, 14, 256)	0	conv4_block1_2_bn[0][0]
conv4_block1_0_conv (Conv2D)	(None, 14, 14, 1024)	525312	conv3_block4_out[0][0]
conv4_block1_3_conv (Conv2D)	(None, 14, 14, 1024)	263168	conv4_block1_2_relu[0][0]
conv4_block1_0_bn (BatchNormali	(None, 14, 14, 1024)	4096	conv4_block1_0_conv[0][0]
conv4_block1_3_bn (BatchNormali	(None, 14, 14, 1024)	4096	conv4_block1_3_conv[0][0]
conv4_block1_add (Add)	(None, 14, 14, 1024)	0	conv4_block1_0_bn[0][0] conv4_block1_3_bn[0][0]
conv4_block1_out (Activation)	(None, 14, 14, 1024)	0	conv4_block1_add[0][0]
conv4_block2_1_conv (Conv2D)	(None, 14, 14, 256)	262400	conv4_block1_out[0][0]
conv4_block2_1_bn (BatchNormali	(None, 14, 14, 256)	1024	conv4_block2_1_conv[0][0]
conv4_block2_1_relu (Activation	(None, 14, 14, 256)	0	conv4_block2_1_bn[0][0]
conv4_block2_2_conv (Conv2D)	(None, 14, 14, 256)	590080	conv4_block2_1_relu[0][0]
conv4_block2_2_bn (BatchNormali	(None, 14, 14, 256)	1024	conv4_block2_2_conv[0][0]
conv4_block2_2_relu (Activation	(None, 14, 14, 256)	0	conv4_block2_2_bn[0][0]
conv4_block2_3_conv (Conv2D)	(None, 14, 14, 1024)	263168	conv4_block2_2_relu[0][0]
conv4_block2_3_bn (BatchNormali	(None, 14, 14, 1024)	4096	conv4_block2_3_conv[0][0]
conv4_block2_add (Add)	(None, 14, 14, 1024)	0	conv4_block1_out[0][0] conv4_block2_3_bn[0][0]
conv4_block2_out (Activation)	(None, 14, 14, 1024)	0	conv4_block2_add[0][0]
conv4_block3_1_conv (Conv2D)	(None, 14, 14, 256)	262400	conv4_block2_out[0][0]
conv4_block3_1_bn (BatchNormali	(None, 14, 14, 256)	1024	conv4_block3_1_conv[0][0]
conv4_block3_1_relu (Activation	(None, 14, 14, 256)	0	conv4_block3_1_bn[0][0]
conv4_block3_2_conv (Conv2D)	(None, 14, 14, 256)	590080	conv4_block3_1_relu[0][0]
conv4_block3_2_bn (BatchNormali	(None, 14, 14, 256)	1024	conv4_block3_2_conv[0][0]
conv4_block3_2_relu (Activation	(None, 14, 14, 256)	0	conv4_block3_2_bn[0][0]
conv4_block3_3_conv (Conv2D)	(None, 14, 14, 1024)	263168	conv4_block3_2_relu[0][0]
conv4_block3_3_bn (BatchNormali	(None, 14, 14, 1024)	4096	conv4_block3_3_conv[0][0]
conv4_block3_add (Add)	(None, 14, 14, 1024)	0	conv4_block2_out[0][0] conv4_block3_3_bn[0][0]
conv4_block3_out (Activation)	(None, 14, 14, 1024)	0	conv4_block3_add[0][0]

conv4_block4_1_conv (Conv2D)	(None, 14, 14, 256)	262400	conv4_block3_out[0][0]
conv4_block4_1_bn (BatchNormali	(None, 14, 14, 256)	1024	conv4_block4_1_conv[0][0]
conv4_block4_1_relu (Activation	(None, 14, 14, 256)	0	conv4_block4_1_bn[0][0]
conv4_block4_2_conv (Conv2D)	(None, 14, 14, 256)	590080	conv4_block4_1_relu[0][0]
conv4_block4_2_bn (BatchNormali	(None, 14, 14, 256)	1024	conv4_block4_2_conv[0][0]
conv4_block4_2_relu (Activation	(None, 14, 14, 256)	0	conv4_block4_2_bn[0][0]
conv4_block4_3_conv (Conv2D)	(None, 14, 14, 1024)	263168	conv4_block4_2_relu[0][0]
conv4_block4_3_bn (BatchNormali	(None, 14, 14, 1024)	4096	conv4_block4_3_conv[0][0]
conv4_block4_add (Add)	(None, 14, 14, 1024)	0	conv4_block3_out[0][0] conv4_block4_3_bn[0][0]
conv4_block4_out (Activation)	(None, 14, 14, 1024)	0	conv4_block4_add[0][0]
conv4_block5_1_conv (Conv2D)	(None, 14, 14, 256)	262400	conv4_block4_out[0][0]
conv4_block5_1_bn (BatchNormali	(None, 14, 14, 256)	1024	conv4_block5_1_conv[0][0]
conv4_block5_1_relu (Activation	(None, 14, 14, 256)	0	conv4_block5_1_bn[0][0]
conv4_block5_2_conv (Conv2D)	(None, 14, 14, 256)	590080	conv4_block5_1_relu[0][0]
conv4_block5_2_bn (BatchNormali	(None, 14, 14, 256)	1024	conv4_block5_2_conv[0][0]
conv4_block5_2_relu (Activation	(None, 14, 14, 256)	0	conv4_block5_2_bn[0][0]
conv4_block5_3_conv (Conv2D)	(None, 14, 14, 1024)	263168	conv4_block5_2_relu[0][0]
conv4_block5_3_bn (BatchNormali	(None, 14, 14, 1024)	4096	conv4_block5_3_conv[0][0]
conv4_block5_add (Add)	(None, 14, 14, 1024)	0	conv4_block4_out[0][0] conv4_block5_3_bn[0][0]
conv4_block5_out (Activation)	(None, 14, 14, 1024)	0	conv4_block5_add[0][0]
conv4_block6_1_conv (Conv2D)	(None, 14, 14, 256)	262400	conv4_block5_out[0][0]
conv4_block6_1_bn (BatchNormali	(None, 14, 14, 256)	1024	conv4_block6_1_conv[0][0]
conv4_block6_1_relu (Activation	(None, 14, 14, 256)	0	conv4_block6_1_bn[0][0]
conv4_block6_2_conv (Conv2D)	(None, 14, 14, 256)	590080	conv4_block6_1_relu[0][0]
conv4_block6_2_bn (BatchNormali	(None, 14, 14, 256)	1024	conv4_block6_2_conv[0][0]
conv4_block6_2_relu (Activation	(None, 14, 14, 256)	0	conv4_block6_2_bn[0][0]
conv4_block6_3_conv (Conv2D)	(None, 14, 14, 1024)	263168	conv4_block6_2_relu[0][0]
conv4_block6_3_bn (BatchNormali	(None, 14, 14, 1024)	4096	conv4_block6_3_conv[0][0]
conv4_block6_add (Add)	(None, 14, 14, 1024)	0	conv4_block5_out[0][0] conv4_block6_3_bn[0][0]

conv4_block6_out (Activation)	(None, 14, 14, 1024)	0	conv4_block6_add[0][0]
conv5_block1_1_conv (Conv2D)	(None, 7, 7, 512)	524800	conv4_block6_out[0][0]
conv5_block1_1_bn (BatchNormali	(None, 7, 7, 512)	2048	conv5_block1_1_conv[0][0]
conv5_block1_1_relu (Activation	(None, 7, 7, 512)	0	conv5_block1_1_bn[0][0]
conv5_block1_2_conv (Conv2D)	(None, 7, 7, 512)	2359808	conv5_block1_1_relu[0][0]
conv5_block1_2_bn (BatchNormali	(None, 7, 7, 512)	2048	conv5_block1_2_conv[0][0]
conv5_block1_2_relu (Activation	(None, 7, 7, 512)	0	conv5_block1_2_bn[0][0]
conv5_block1_0_conv (Conv2D)	(None, 7, 7, 2048)	2099200	conv4_block6_out[0][0]
conv5_block1_3_conv (Conv2D)	(None, 7, 7, 2048)	1050624	conv5_block1_2_relu[0][0]
conv5_block1_0_bn (BatchNormali	(None, 7, 7, 2048)	8192	conv5_block1_0_conv[0][0]
conv5_block1_3_bn (BatchNormali	(None, 7, 7, 2048)	8192	conv5_block1_3_conv[0][0]
conv5_block1_add (Add)	(None, 7, 7, 2048)	0	conv5_block1_0_bn[0][0] conv5_block1_3_bn[0][0]
conv5_block1_out (Activation)	(None, 7, 7, 2048)	0	conv5_block1_add[0][0]
conv5_block2_1_conv (Conv2D)	(None, 7, 7, 512)	1049088	conv5_block1_out[0][0]
conv5_block2_1_bn (BatchNormali	(None, 7, 7, 512)	2048	conv5_block2_1_conv[0][0]
conv5_block2_1_relu (Activation	(None, 7, 7, 512)	0	conv5_block2_1_bn[0][0]
conv5_block2_2_conv (Conv2D)	(None, 7, 7, 512)	2359808	conv5_block2_1_relu[0][0]
conv5_block2_2_bn (BatchNormali	(None, 7, 7, 512)	2048	conv5_block2_2_conv[0][0]
conv5_block2_2_relu (Activation	(None, 7, 7, 512)	0	conv5_block2_2_bn[0][0]
conv5_block2_3_conv (Conv2D)	(None, 7, 7, 2048)	1050624	conv5_block2_2_relu[0][0]
conv5_block2_3_bn (BatchNormali	(None, 7, 7, 2048)	8192	conv5_block2_3_conv[0][0]
conv5_block2_add (Add)	(None, 7, 7, 2048)	0	conv5_block1_out[0][0] conv5_block2_3_bn[0][0]
conv5_block2_out (Activation)	(None, 7, 7, 2048)	0	conv5_block2_add[0][0]
conv5_block3_1_conv (Conv2D)	(None, 7, 7, 512)	1049088	conv5_block2_out[0][0]
conv5_block3_1_bn (BatchNormali	(None, 7, 7, 512)	2048	conv5_block3_1_conv[0][0]
conv5_block3_1_relu (Activation	(None, 7, 7, 512)	0	conv5_block3_1_bn[0][0]
conv5_block3_2_conv (Conv2D)	(None, 7, 7, 512)	2359808	conv5_block3_1_relu[0][0]
conv5_block3_2_bn (BatchNormali	(None, 7, 7, 512)	2048	conv5_block3_2_conv[0][0]
conv5_block3_2_relu (Activation	(None, 7, 7, 512)	0	conv5_block3_2_bn[0][0]

```
conv5_block3_3_conv (Conv2D)    (None, 7, 7, 2048)   1050624     conv5_block3_2_relu[0][0]
conv5_block3_3_bn (BatchNormali (None, 7, 7, 2048)   8192        conv5_block3_3_conv[0][0]
conv5_block3_add (Add)          (None, 7, 7, 2048)   0           conv5_block2_out[0][0]
                                                                 conv5_block3_3_bn[0][0]
conv5_block3_out (Activation)   (None, 7, 7, 2048)   0           conv5_block3_add[0][0]
flatten_1 (Flatten)             (None, 100352)       0           conv5_block3_out[0][0]
dropout (Dropout)               (None, 100352)       0           flatten_1[0][0]
dense (Dense)                   (None, 10)           1003530     dropout[0][0]
==================================================================================================
Total params: 24,591,242
Trainable params: 1,003,530
Non-trainable params: 23,587,712
```

9 Training The Model

A callback is an object that can perform actions at various stages of training (e.g. at the start or end of an epoch, before or after a single batch, etc).

Early Stopping is use to prevent model from overfitting.

In [33]:
```
es=EarlyStopping(monitor='val_accuracy', mode='max', verbose=1, patience=20)
```

In [34]:
```
mc = ModelCheckpoint('./working/model.h5', monitor='val_accuracy', mode='max' )
```

In [35]:
```
History = modelT.fit_generator(train_generator,validation_data=val_generator,epochs=10,verbose=1, callbacks=[mc,es])
```

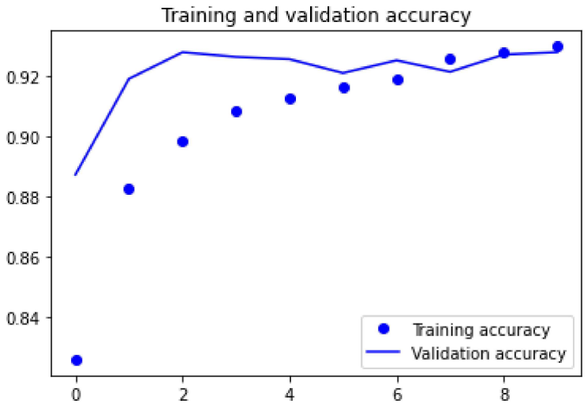
```
/opt/conda/lib/python3.7/site-packages/keras/engine/training.py:1972: UserWarning:

`Model.fit_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators.

Epoch 1/10
655/655 [==============================] - 265s 390ms/step - loss: 5.0193 - accuracy: 0.8259 - val_loss: 4.7282 - val_accuracy: 0.8871
/opt/conda/lib/python3.7/site-packages/keras/utils/generic_utils.py:497: CustomMaskWarning:

Custom mask layers require a config and must override get_config. When loading, the custom mask layer must be passed to the custom_objects argument.
```

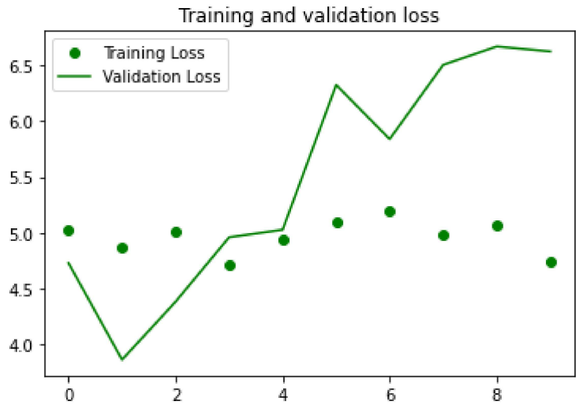
```
Epoch 2/10
655/655 [==============================] - 254s 387ms/step - loss: 4.8719 - accuracy: 0.8823 - val_loss: 3.8639 - val_accuracy: 0.9189
Epoch 3/10
655/655 [==============================] - 254s 388ms/step - loss: 5.0158 - accuracy: 0.8981 - val_loss: 4.3832 - val_accuracy: 0.9277
Epoch 4/10
655/655 [==============================] - 255s 389ms/step - loss: 4.7195 - accuracy: 0.9083 - val_loss: 4.9593 - val_accuracy: 0.9262
Epoch 5/10
655/655 [==============================] - 251s 383ms/step - loss: 4.9447 - accuracy: 0.9127 - val_loss: 5.0267 - val_accuracy: 0.9254
Epoch 6/10
655/655 [==============================] - 254s 388ms/step - loss: 5.0986 - accuracy: 0.9163 - val_loss: 6.3222 - val_accuracy: 0.9208
Epoch 7/10
655/655 [==============================] - 253s 385ms/step - loss: 5.1980 - accuracy: 0.9189 - val_loss: 5.8380 - val_accuracy: 0.9250
Epoch 8/10
655/655 [==============================] - 253s 386ms/step - loss: 4.9863 - accuracy: 0.9257 - val_loss: 6.5021 - val_accuracy: 0.9212
Epoch 9/10
655/655 [==============================] - 253s 386ms/step - loss: 5.0748 - accuracy: 0.9280 - val_loss: 6.6662 - val_accuracy: 0.9269
Epoch 10/10
655/655 [==============================] - 254s 388ms/step - loss: 4.7483 - accuracy: 0.9297 - val_loss: 6.6218 - val_accuracy: 0.9277
```

10 Plotting the Graph and Result

In [36]:

```
acc = History.history['accuracy']
val_acc = History.history['val_accuracy']
loss = History.history['loss']
val_loss = History.history['val_loss']

epochs = range(len(acc))

plt.plot(epochs, acc, 'bo', label='Training accuracy')
plt.plot(epochs, val_acc, 'b', label='Validation accuracy')
plt.title('Training and validation accuracy')
plt.legend()

plt.figure()

plt.plot(epochs, loss, 'go', label='Training Loss')
plt.plot(epochs, val_loss, 'g', label='Validation Loss')
plt.title('Training and validation loss')
plt.legend()

plt.show()
```





11 Prediction

In [37]:
```
test_loss, test_acc = modelT.evaluate(test_generator, steps=len(test_generator), verbose=1)
print('Loss: %.3f' % (test_loss * 100.0))
print('Accuracy: %.3f' % (test_acc * 100.0))
```

```
83/83 [==============================] - 7s 82ms/step - loss: 7.0904 - accuracy: 0.9364
Loss: 709.037
Accuracy: 93.643
```

In [38]:
```
from sklearn.metrics import classification_report
```

In [39]:
```
y_val = test_generator.classes
y_pred = modelT.predict(test_generator)
y_pred = np.argmax(y_pred,axis=1)
```

In [40]:
```
print(classification_report(y_val,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.90      0.97      0.93       487
           1       0.88      0.96      0.92       263
           2       0.98      0.88      0.93       146
           3       0.95      0.97      0.96       212
           4       0.95      0.97      0.96       311
           5       0.92      0.91      0.91       168
           6       0.93      0.74      0.83       188
           7       0.90      0.93      0.92       182
           8       0.99      0.97      0.98       483
           9       0.96      0.92      0.94       187

    accuracy                           0.94      2627
   macro avg       0.94      0.92      0.93      2627
weighted avg       0.94      0.94      0.94      2627
```

In [41]:
```
class_indices = test_generator.class_indices
indices = {v:k for k,v in class_indices.items()}
```

In [42]:
```
filenames = test_generator.filenames
```

In [43]:
```
val_df = pd.DataFrame()
val_df['filename'] = filenames
val_df['actual'] = y_val
val_df['predicted'] = y_pred
val_df['actual'] = val_df['actual'].apply(lambda x: indices[x])
val_df['predicted'] = val_df['predicted'].apply(lambda x: indices[x])
val_df.loc[val_df['actual']==val_df['predicted'],'Same'] = True
val_df.loc[val_df['actual']!=val_df['predicted'],'Same'] = False
val_df.head(10)
```

Out[43]:

	filename	actual	predicted	Same
0	cane/OIP---ZRsOF7zsMqhW30WeF8-AHaFj.jpeg	cane	cane	True
1	cane/OIP--DQsiH-9LgIDNIEvJpZQZQHaFw.jpeg	cane	cane	True
2	cane/OIP--OzHmoOBxcUQs7N3KjButwHaFP.jpeg	cane	cane	True
3	cane/OIP--Z2dg_o5sGo8jua-d6vSagHaEK.jpeg	cane	cane	True
4	cane/OIP--hrVyxBHfXGXIpLKB_Id2AHaEK.jpeg	cane	cane	True
5	cane/OIP--iPPduM2--389E1IozUz_QHaFj.jpeg	cane	cane	True
6	cane/OIP--khXa4p9B3QV8JmsHX29hgHaEK.jpeg	cane	cane	True
7	cane/OIP-00zvckpZ8XDEzk-Fv4HjkQHaGL.jpeg	cane	cane	True
8	cane/OIP-0ERf2k91vjSIq_POpXYImwHaFj.jpeg	cane	cane	True
9	cane/OIP-0G1ImzPnTshwQJ_IEMNHgQHaFR.jpeg	cane	cane	True

In [44]:
```
val_df = val_df.sample(frac=1).reset_index(drop=True)
```

12 Prediction comparison

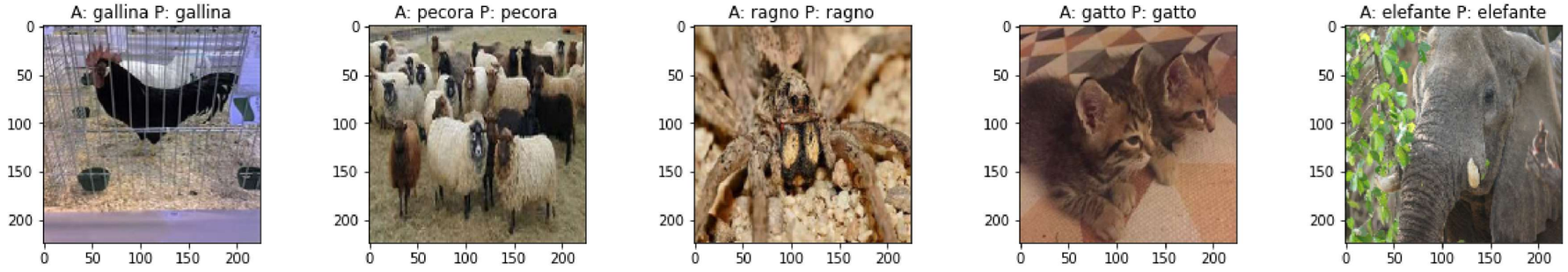
In [45]:
```
from tensorflow.keras.preprocessing.image import ImageDataGenerator, array_to_img, img_to_array, load_img
img_size = 224
def readImage(path):
    img = load_img(path,color_mode='rgb',target_size=(img_size,img_size))
    img = img_to_array(img)
    img = img/255.

    return img

def display_images(temp_df):
    temp_df = temp_df.reset_index(drop=True)
    plt.figure(figsize = (20 , 20))
    n = 0
    for i in range(15):
        n+=1
        plt.subplot(5 , 5, n)
        plt.subplots_adjust(hspace = 0.5 , wspace = 0.3)
        image = readImage(f"../input/animals10/raw-img/{temp_df.filename[i]}")
        plt.imshow(image)
        plt.title(f'A: {temp_df.actual[i]} P: {temp_df.predicted[i]}')
```

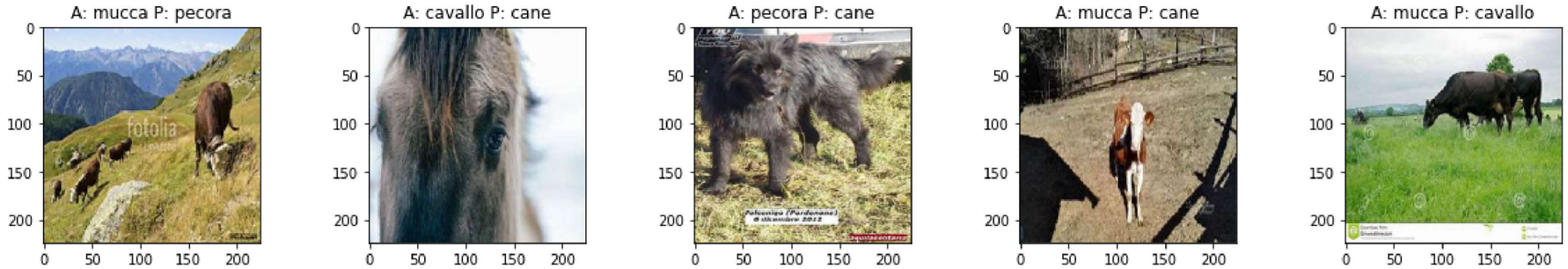
Correctly classified.

In [46]:
```
display_images(val_df[val_df['Same']==True])
```


Missclassified

In [47]:
```
display_images(val_df[val_df['Same']!=True])
```





13 Confusion Matrix

In [48]:
```
cm = confusion_matrix(y_true=y_val, y_pred=y_pred)
```

In [49]:
```
def plot_confusion_matrix(cm, classes,normalize=False,title='Confusion matrix',cmap=plt.cm.Blues):

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    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]
        print("Normalized confusion matrix")
    else:
```

```
        print('Confusion matrix, without normalization')
        print(cm)

    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 label')
        plt.xlabel('Predicted label')
```

In [50]:
```
cm_plot_labels = ['cane', 'ragno', 'mucca', 'scoiattolo', 'elefante', 'pecora', 'gatto', 'farfalla', 'gallina', 'cavallo']

plot_confusion_matrix(cm=cm, classes=cm_plot_labels, title='Confusion Matrix')
```

```
Confusion matrix, without normalization
[[470   5   1   0   5   5   0   1   0   0]
 [  5 252   0   1   0   1   2   2   0   0]
 [  1   4 129   0   0   1   3   6   0   2]
 [  0   0   0 205   2   1   0   0   4   0]
 [  5   2   0   0 302   1   1   0   0   0]
 [ 10   0   0   0   0 153   0   2   0   3]
 [ 17  19   1   0   2   1 140   8   0   0]
 [  4   3   0   0   0   0   5 170   0   0]
 [  3   0   0   8   3   0   0   0 467   2]
 [  5   0   0   2   3   4   0   0   1 172]]
```