DOG BREED PREDICTION

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

RGUKT

Rajiv Gandhi University of Knowledge Technologies

RK VALLEY

Submitted by

MADDINENI ANIL KUMAR (R180864)

DUGGI NIVEDHITHA (R180854)

BUKKE ABHILASH NAIK (R180886)

Under the esteemed Guidance of

Mr. SANTOSH KUMAR

RGUKT RK VALLEY

DECLARATION

We hereby declare that the work presented in this report entitled **"Dog Breed Prediction"** in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering** submitted in the department of Computer Science & Engineering, Rajiv Gandhi University of Knowledge Technologies, RK Valley is an authentic record of our own work carried out over a period from July 2023 to December 2023 under the supervision of Mr. **Santosh Kumar**(Assistant Professor, Computer Science &Engineering Department). The matter embodied in the report has not been submitted for the award of any other degree or diploma.

MADDINENI ANIL KUMAR (R180864)

DUGGI NIVEDHITHA (R180854)

BUKKE ABHILASH NAIK (R180886)

Dept. of Computer Science and Engineering

ACKNOWLEDGEMENT

We owe our profound gratitude to our project supervisor Mr. **Santosh Kumar**, who took keen interest and guided us all along in our project work titled - **"Dog Breed Prediction"** till the completion of our project by providing all the necessary information for developing the project. The project development helped us in research and we got to know a lot of new things in our domain. We are really thankful to him.

RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES

RGUKT

(A. P. Government Act 18 of 2008)

RGUKT, RK VALLEY

Department of Computer Science and Engineering

CERTIFICATE FOR PROJECT COMPLETION

This is to certify that the project entitled "**DOG BREED PREDICTION**" submitted by **MADDINENI ANIL KUMAR(R180864), DUGGI NIVEDHITHA(R180854), BUKKE ABHILASH NAIK(R180886),** under our guidance and supervision for the partial fulfilment for the degree Bachelor of Technology in Computer Science and Engineering during the academic semester -2023-2024 at RGUKT, RK VALLEY. To the best of my knowledge, the results embodied in this dissertation work have not been submitted to any University or Institute for the award of any degree or diploma.

Project Internal Guide

Mr. Santosh Kumar

Assistant Professor

RGUKT, RK Valley

Head of the Department

Mr. Satyanandaram N

HOD of CSE

RGUKT, RK Valley

ABSTRACT

Pattern recognition(PR) is realized as a human recognition process which can be completed by computer technology. We should first enter useful information of identifying the object into the computer. For this reason, we must abstract the recognition object and establish its mathematical model to describe it and replace the recognition object for what the machine can process. The description of this object is the pattern. Simply speaking, the pattern recognition is to identify the category to which the object belongs, such as the face in face recognition. Our project is based on PR which is to identify the dog's breed. In our project, based on 10,000+ images of 120 breeds of dogs, we use 4 methods to do the identification. Each method has a different training model. The four models are ResNet18, VGG16, DenseNet161, and Alex Net. Based on our models, we also make some improvements on the optimization methods to increase our identification accuracy. After our comparisons, we find that the ResNet model & VGG16 together is the best, and we take it as our prime model. Our best accuracy can be up to 75.14%. This project uses computer vision and machine learning techniques to predict dog breeds from images. First, we identify dog facial key points for each image using a convolutional neural network. These key points are then used to extract features via SIFT descriptors and color histograms. We then compare a variety of classification algorithms, which use these features to predict the breed of the dog shown in the image. Our best classifier is an SVM with a linear kernel and it predicts the correct dog breed on its first guess 52% of the time; 90% of the time the correct dog breed is in the top 10 predictions.

CHAPTER 1

DOG BREDD PREDICTION USING CNN AND IMAGE PROCESSING

INTRODUCTION:

IMAGE PROCESSING:

Image Processing is a very powerful technique used today to convert an image into a form which is either digital or analog so that it can be used to extract some important and useful data. This process takes a raw image as an input and processes it and gives an improved modified image or characteristics associated with that image as an output. Image processing when used in ML algorithms such as CNN can be used to get very interesting results such as image recognition or creating a model to predict some feature from image. Mainly these three processes are involved in image processing.

- Fetching the required image by using any available tool.
- The fetched image is then analysed and some necessary manipulation is done on it to find some significant patterns in it which are not visible to a naked human eye.
- The last stage is the output stage where the output is either an image or a report based.

What is an Image?

An image is a digital representation of a picture i.e, electronic form. When an image is stored in rastere form it is called bitmap.
JPEG, PNG, GIF89a, GIF, SVG.

Types of Images

1. Binary (0/1) Image: This is a digital image in which every pixel is capable of taking only two colors i.e. black and white. The black color is denoted by 1 and while by 0, that's why it is also called a 0/1 image.

2. Gray scale images: This is a digital image in which each pixel instead of containing color, contains the intensity information of light. This image is also called black and white image or monochrome image.

3. Colored images: This is a digital image in which every pixel contains the intensity of RGB (Red, Green, Blue) color. RGB is chosen because all other colors can be formed by mixing the intensity of these three colors. A colored image every pixel stores 3 values.

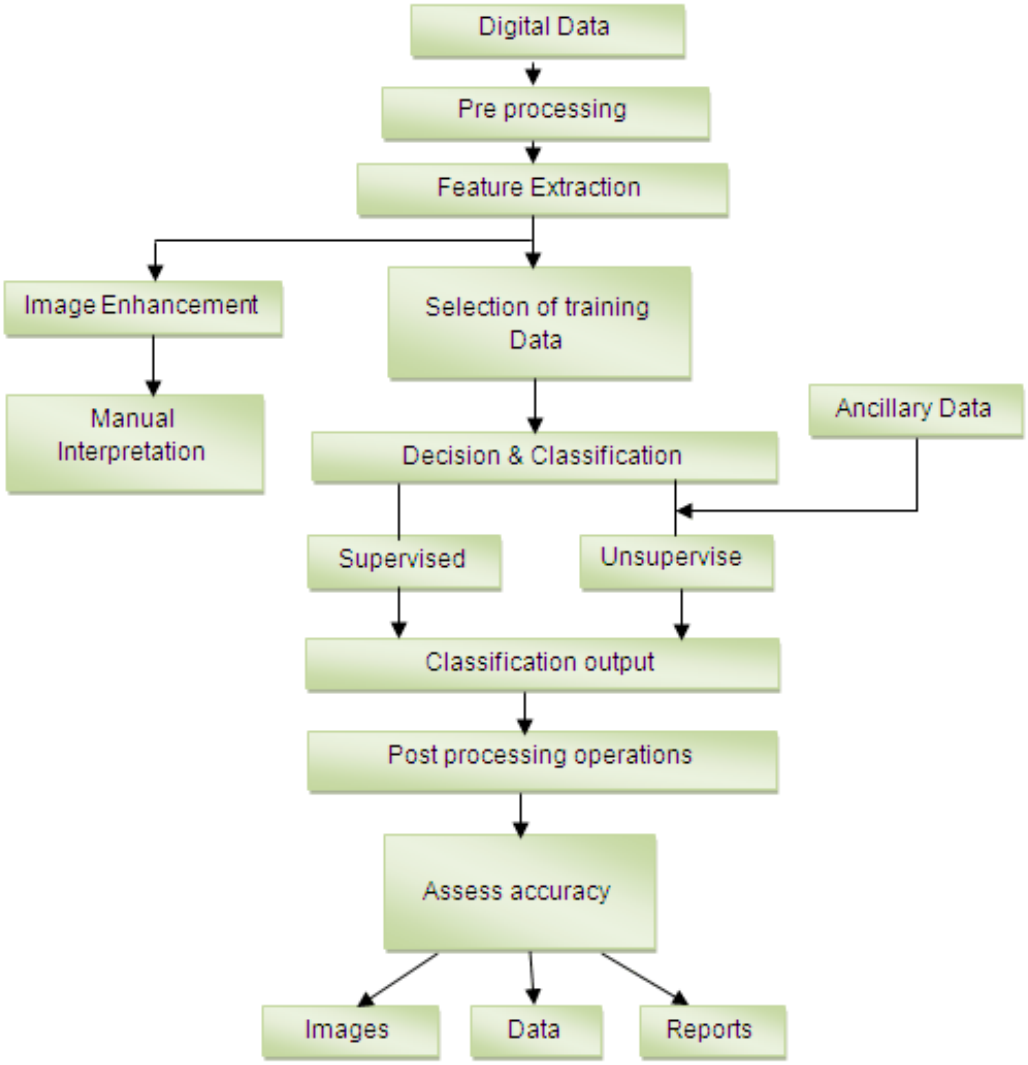


Fig. 1: Flow Chart Showing Diffrent Phases in Digital Image Processing

Types of Digital Images Processing Techniques:

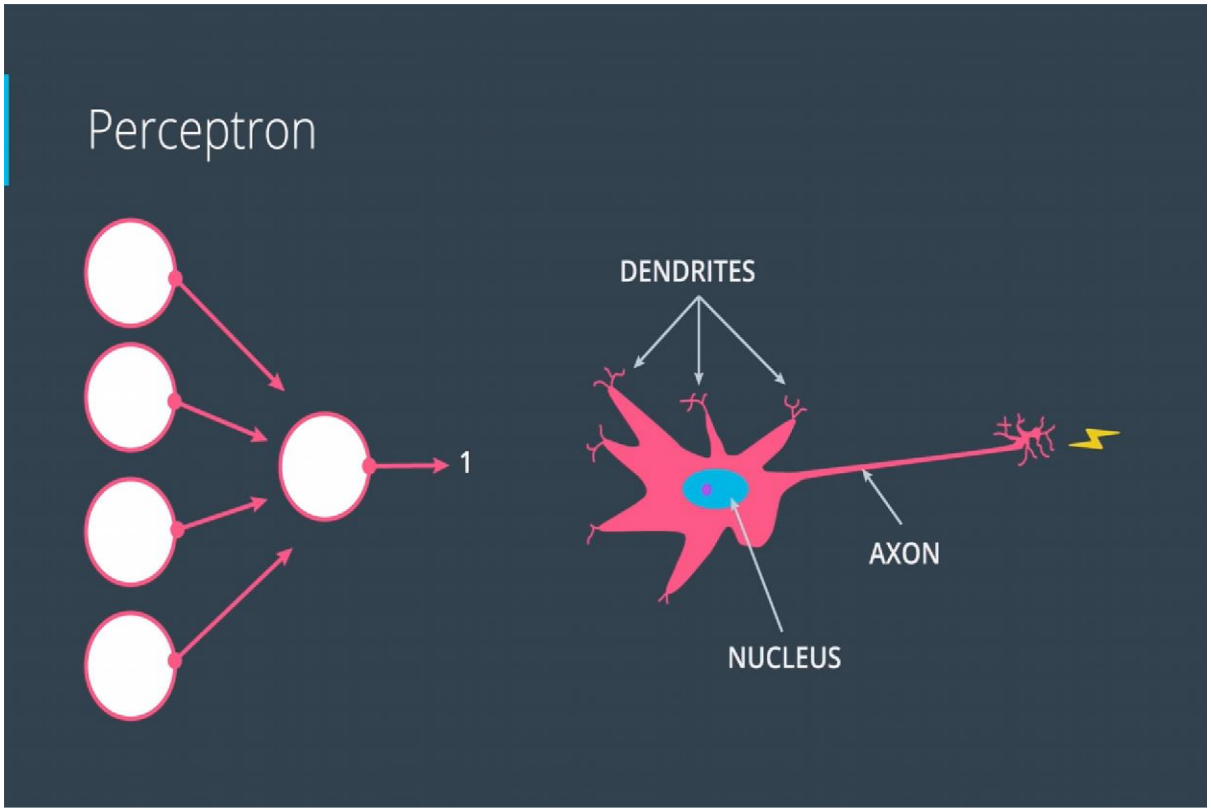
- Editing of the image
- Anisotropic Diffusion
- Linear Filtering of image
- Neural Networks (CNN)
- Image analysis
- Image segmentation
- Image data compression.
- Component Analysis

Convolutional Neural Networks and images(CNN):

Convolution Neural Network is a word taken from neural network present in human body because of their analogy. The working of Convolution Neural Network is exactly same as the working of human neural network.

The human Neural Network is a collection of collection of neurons, dendrites that generates the output signal which is sent to different parts of body. Neural Network is present in brain which performs various computations. The dendrites present at the ends of nerve cells act as sensors and receive the signals, it then converts it to electrical signal which is then passes to the nucleus of the cell through axon where the decision is taken and message is again sent back via similar process.

CNN also work similar to human neural network. This network consists of neurons, weights and biases. This network receives the input and uses the biases and weights to calculate the weighted sum of the input and passes this sum through an activation function and finally comes up with a prediction or an output.



ARTIFICIAL CONVOLUTION NEURAL NETWORKS VS REAL NEURAL NETWORKS.

Activation Function in CNN:

As we know neural network consists of a large number of neurons, the activation function decides whether the neuron need to be activated or not and it is done by calculating the sum of the weights and bias together. This function is mainly required to bring non linearity to our model. Without an activation function our model will just be a linear model and will not be able to train and predict accurately for data which forms a complex boundary. It will only work as linear model which will be of not much use to us.

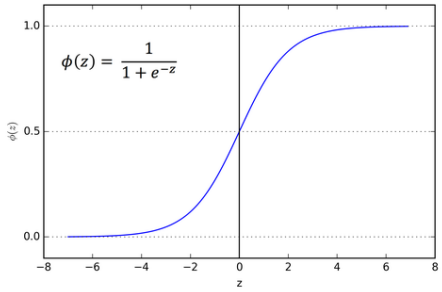
Different types of activation functions:

1. **Sigmoid Function**: The mathematical formula it uses is **f**(a) = 1 / 1 + expp(-a).
   The values of the function lies between 0 and 1.

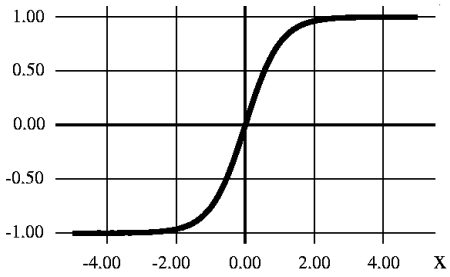
2. **Tanh (Hyperbolic Function):** The mathematical formula used for this function is **f(a) = 1- expp(-2a) / 1 + expp(-2a).** The value of this function is between -1 to 1.

3. **ReLu- Rectified Linear units :** The mathematical formula used for this function is R(a) = maximum (0, a) i.e., if a < 0, R(a) = 0 and if a >= 0, R(a) = a.

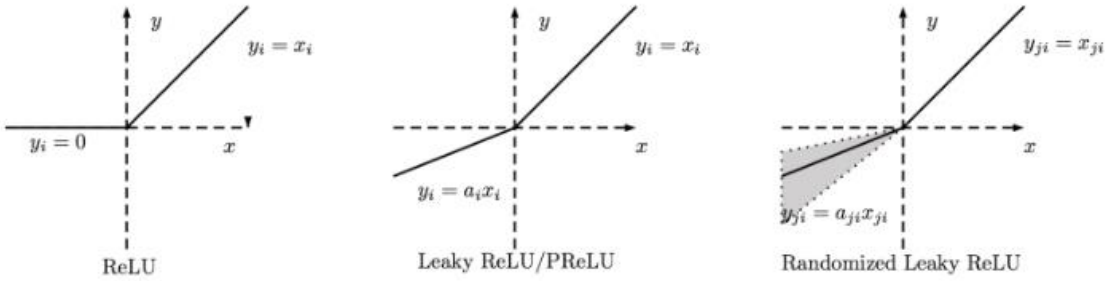
Sigmoid Function:



TanH Function:



ReLu Function:



1.1) PROBLEM STATEMENT

Build a model using deep learning algorithms and image processing to predict the breed of 120 breeds of dogs by taking their image as input. Use transfer learning to build a model that recognizes up to 120 different dog breeds.

1.2) OBJECTIVE

The main objective of this project is:

- To use deep learning algorithms and predict the breed of the dog.
- To get best possible testing data accuracy.
- To train the model on 120 breeds of dogs on over 10,000 images.
- To use CNN to recognize the patterns in the train data and make changes to the model.

- To predict the breed irrespective of the child or adult member of that breed.
- Use Python, Keras and image processing tools to filter our data and cleaning of our input data.

1.3) METHODOLOGY

So, initially, the data is converted into blacks and white images. The data is broken into three sets which are basically training set, testing set, and validation set. The training dataset is used for calculating the pattern in images and adjusting the images. This training dataset is passed through a deep neural network. Feedforward is done and backpropagation is done to calculate the error and adjustment of weights. Then we plug in the image to know the breed of a dog. This breed is then revealed by feedforward. The probabilities of dog breeds are predicted and thus output is calculated.

CHAPTER 2

LITERATURE REVIEW

What is Machine Learning?

Machine learning is a field of Artificial Intelligence that uses data analysis and machine learning algorithms to create a model and trains itself according to the training data and learns some parameters according to it and uses them to predict some output according to those parameters. The data provided is divided to training and testing data and the accuracy of the algorithm is calculated on the basis of test score it obtains.

Machine Learning is mainly divided into 4 categories:

1. **Supervised Machine Learning:**

   In this type of algorithm, the training data is used to analyse the data given, learns some parameters and creates a function which takes the data entries as input and passes them through that function to calculate the output. In input data the features and output columns are clearly separated. Almost 70% of the machine learning is Supervised Machine Learning.

   Algorithms which use supervised machine learning are:

   - Linear Regression
   - Logistic Regression
   - Decision Trees
   - Naive Bays
   - KNN
   - K-Means
   - Random Forest

2. **Unsupervised Machine Learning:**

   This kind of algorithm is used when the input data is not classified as well as not labelled. These algorithms identify the patterns in the provided data and draw inferences from the data sets to identify the data which is unstructured and unlabelled. However, these algorithms are not capable of predicting very accurate results but still it provides quite efficient results base on the kind of unstructured data provided. These algorithms are more complex and difficult to understand than the supervised algorithms.

   Algorithms which use unsupervised learning are:

   - Hierarchical clustering
   - K-means clustering
   - Expectation minimization

3. **Semi-Supervised Machine Learning:**

These are the algorithms which use the combination of both supervised and unsupervised learning techniques. They use both labelled as well as unlabelled data to compute the patterns available and create function to predict the output for the rest data. For its working it can use a bit of labelled data and most part of unlabelled data or its vice versa. Classification, Regression and recognition are some of the methods it uses.

Algorithms which use unsupervised learning are:

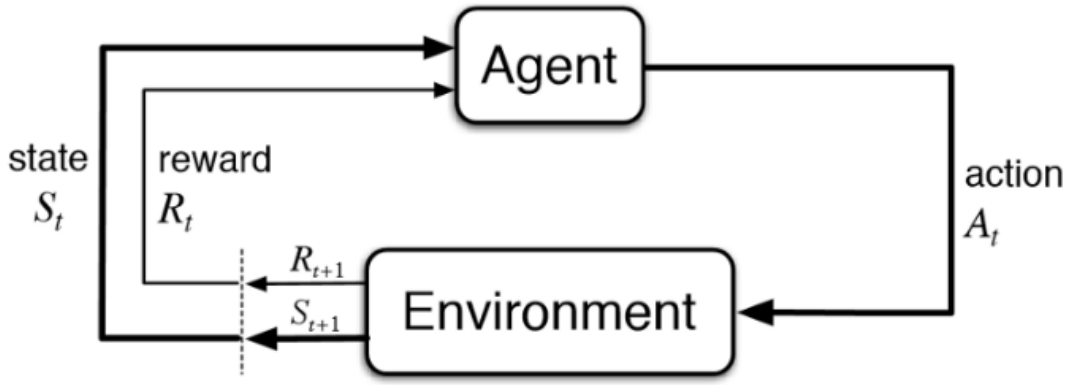
- Face recognition techniques
- Voice recognition techniques

4. **Reinforcement Learning:**

It is a kind of learning where the model interacts with the real world environment for training itself and discover patterns. Here the model learns from its own outputs. Trial and error method is applied where a model performs a lot of trials on the data and tries to predict the output based on previous learning, and for every correct prediction the model is given some positive reward and for a wrong prediction it is given some negative points. Hence by making more and more trials the accuracy of the model increases significantly. Reward and feedback are required by the agent to learn new features and increase accuracy of the system.

Algorithms which use unsupervised learning are:

- Q Learning Algorithm
- State-Action-Reward-State-Action (SARSA)
- Deep Q Network (DQN)
- Deep Deterministic Policy Gradient (DDPG)



Reinforcement Learning Illustration (https://i.stack.imgur.com/eoeSq.png)

How does a machine look at an image?

The brain possessed by human beings is very powerful. It capable of doing multiple, very different and very difficult things together. For an example we can say our eyes see a lot of things every millisecond and the brain process the image to figure out what that image is within no time and we never pay attention to it. But machines are not like human brains. We have to make them learn the images so that they come to know about it, and in doing so the first step is to use image processing to convert an image to a digital image so that our machine is able to read that.

An image in straightforward words is a collection of pixels which are very small in size and the intensity of colour in a unique and a spatial order. This pixel formation is different for different images, changing which the whole image is changed.

To understand this let us take an example where we want to look an image with 5 written in it, The read this the image will be changed to a lattice of pixels where each pixel will contain a value ranging from 0 to 255 where 0 represents the whitest shade of colour and 255 represent the darkest shade of colour.

The neural network then identifies the patterns in that matrix to remember that image so that it can later itself recognize that image. This neural network will take the value of the pixels as a feature on which the ML algorithm will work to predict the output. It is almost impossible for a human eye to recognize the pattern which this network learns to identify that image. We can provide different weights and bias to the model to alter our results.

Representation of Matrix in Pixels:

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Defining a Convolutional Neural Network

A convolutional neural network is a spatial neural network which is mostly used to recognize patterns in an image an input. CNN basically is defined by the following 3 steps:

1. The Convolutional Layer of CNN
2. The Pooling Layer of CNN
3. The output layer of CNN

These layers are explained as:

The Convolutional Layer of CNN:

As the image we have, is stored in the form of a large matrix where each entry contains pixel intensity ranging from 0 to 255. So to calculate or recognize the features we create a 2D weight matrix who's each entry contains a weight. Mostly we take it to be a square matrix of any size like 3*3 or 4*4 etc. now this weight window is moved all over the pixel matrix and the overlapping pixel value is multiplied with the corresponding weight value, and they are summed up to get a we weight of that window. Now this pixel value is moved to right or down so that the whole image is covered and the calculated numbers for each window are added at the end. These processed values help to recognize patters of the image. The weight matrix behaves like a filter which moves over the whole image and extract the important and useful features of that image.

INPUT IMAGE

18	54	51	239	244	188
55	121	75	78	95	88
35	24	204	113	109	221
3	154	104	235	25	130
15	253	225	159	78	233
68	85	180	214	245	0

WEIGHT

1	0	1
0	1	0
1	0	1

429

Original Image

Convoluted image

Multiple filters are used for analysing and recognizing the depth effect in the input. A single layer filter is not able to get the output with depth as a dimension. Multiple layers will result in a 3D weighted matrix and will take care of depth as a dimension. The output from each filter is summed and compiled together to finally form a convoluted image.

The Pooling Layer:

Pooling is technique which is mainly used when the size of the image is to lagre. If the size of the image is too large we need to reduce the number of learning parameters present in that image recognition. And while doing so it is required to insert pooling layers in between them. It is mainly done so that the size of image does not become too large and while doing so the image depth dimensions are not altered hence there is no point that the depth of the image will be affected in any way. Max pooling is one of the most used pooling which is generally applied. The output image looks almost very much similar to the original image and only the dimensions of image are altered.

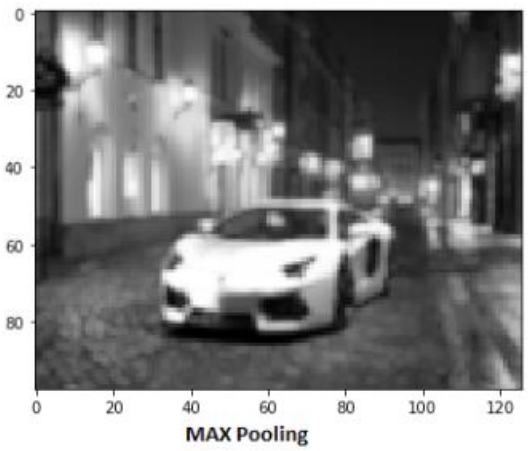
Other types of pooling are average pooling or L2 norm pooling. The size of the output image is controlled by 3 parameters:

- **The number of filters:**
  The number of filters in a network determine the width of the image. Thus the output volume of the image will depend on the number of filters used. Also while pooling the width parameters in filter are not changed at all.

- **Stride:**
  The value of stride decides that by how much amount does the filter shifts to right or down in the pixel matrix. If its value is set to 1 it can move up, down or right by only one-pixel value hence the size of output image will be large. So we use a higher value of stride so that our output size gets reduced.

- **Zero Padding:**
  This feature helps us to maintain and retain the size of our original input image. If this value is set to 0 our original input image will remain as it is but if it is set to a higher value, we can lose our original image.

Convoluted image



MAX Pooling

The Output Layer:

After passing through the convolutional layer and pooling layer we get a compressed image with a lot of features being extracted from it. But after applying the multiple layers to the image we finally have to extract the output class so we have to apply the full connected layers so that we are able to come up with the required output.

For doing so we need to create a 3D activation map to predict whether an image belongs to a particular class or not. The error can also be predicted in the output class by using the loss functions for eg. Cross entropy can be used.

The size of the output image can be calculated using formula: (**(W-F+2P)/S) +1**

- W is input volume size.
- F is filter size.
- P is number of padding applied.
- S is number of strides used.

CHAPTER 3

SYSTEM DEVELOPMENT

Architecture of model:

```
import tensorflow as tf

input_shape=(375, 375)
input=tf.keras.layers.Input((375, 375, 3), name="Input")
'''2 VGG Model'''
#2.1 Adding VGG_PreProcess
vgg_pre_process=tf.keras.layers.Lambda(tf.keras.applications.vgg16.preprocess_input, name='VGG_PreProcess') (input)
#2.2 Downloading VGG Model
vgg_model=tf.keras.applications.VGG16(include_top=False, input_shape=(375, 375, 3))
vgg_model.trainable=False
vgg=vgg_model (vgg_pre_process)

#2.3 #Adding VGG GAP Layer

'''3 REsNet Model'''
vgg_avg=tf.keras.layers.GlobalAveragePooling2D(name="VGG_Average") (vgg)
#3.1 ResNet PreProcess
resnet_pre_process=tf.keras.layers.Lambda(tf.keras.applications.resnet50.preprocess_input, name="Resnet_PreProcess") (input)
#3.2 #Downloading ResNet Model
resnet_model=tf.keras.applications.ResNet50(include_top=False, input_shape=(375, 375, 3))
resnet_model.trainable=False
resnet= resnet_model (resnet_pre_process)
#3.3 #Adding ResNet GAP Layer
resnet_avg=tf.keras.layers.GlobalAveragePooling2D(name="Resent_Average") (resnet)

'''4 Concatinating VGG and ResNet'''
concat=tf.keras.layers.Concatenate( name="Concat") ([vgg_avg, resnet_avg])

#drop1=tf.keras.layers.Dropout(0.5, name="DropOut_1") (concat)
dense1=tf.keras.layers.Dense(512, activation="relu", name="Hidden_1") (concat) #Dense Layer
drop2=tf.keras.layers.Dropout(0.25, name="DropOut_2") (dense1) #DropOut Layer
dense2=tf.keras.layers.Dense(256, activation="relu", name="Hidden_2") (drop2) #Dense Layer
drop3=tf.keras.layers.Dropout(0.25, name="DropOut_3") (dense2) #DropOut Layer
output=tf.keras.layers.Dense(120, activation="softmax", name="Output") (drop3) #Dense Layer(Output)

final_model=tf.keras.models.Model(inputs=[input],
                                  outputs=[output])
```

So this is architecture of our model in which we have added VGG16 & ResNet bottleneck features initially and added our dense layer model with the soft max Activation function. Softmax 16 activation will convert the data into some probabilities. The probabilities of each breed is predicted and based on the maximum value the output is thus calculated.

This architecture is bit complex because it contains bottleneck features of exception model and this model has a great power in the determination of different types of objects.

So the output shape of the VGG16 & ResNet model is 512 & 2048 respectively which will be the input to our model. The output from the dense layer is 120 which is due to 120 breeds present in our output.

Maximum probable breed will be picked up from our model and result will be displayed based on this.

```
Model: "model"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 Input (InputLayer)             [(None, 375, 375, 3  0           []
                                )]

 VGG_PreProcess (Lambda)        (None, 375, 375, 3)  0           ['Input[0][0]']

 Resnet_PreProcess (Lambda)     (None, 375, 375, 3)  0           ['Input[0][0]']

 vgg16 (Functional)             (None, 11, 11, 512)  14714688    ['VGG_PreProcess[0][0]']

 resnet50 (Functional)          (None, 12, 12, 2048  23587712    ['Resnet_PreProcess[0][0]']
                                )

 VGG_Average (GlobalAveragePool  (None, 512)         0           ['vgg16[0][0]']
 ing2D)

 Resent_Average (GlobalAverageP  (None, 2048)        0           ['resnet50[0][0]']
 ooling2D)

 Concat (Concatenate)           (None, 2560)         0           ['VGG_Average[0][0]',
                                                                  'Resent_Average[0][0]']

 Hidden_1 (Dense)               (None, 512)          1311232     ['Concat[0][0]']

 DropOut_2 (Dropout)            (None, 512)          0           ['Hidden_1[0][0]']

 Hidden_2 (Dense)               (None, 256)          131328      ['DropOut_2[0][0]']

 DropOut_3 (Dropout)            (None, 256)          0           ['Hidden_2[0][0]']

 Output (Dense)                 (None, 120)          30840       ['DropOut_3[0][0]']

==================================================================================================
Total params: 39,775,800
Trainable params: 1,473,400
Non-trainable params: 38,302,400
__________________________________________________________________________________________________
```

ALGORITHM

Basic Algorithm Structure

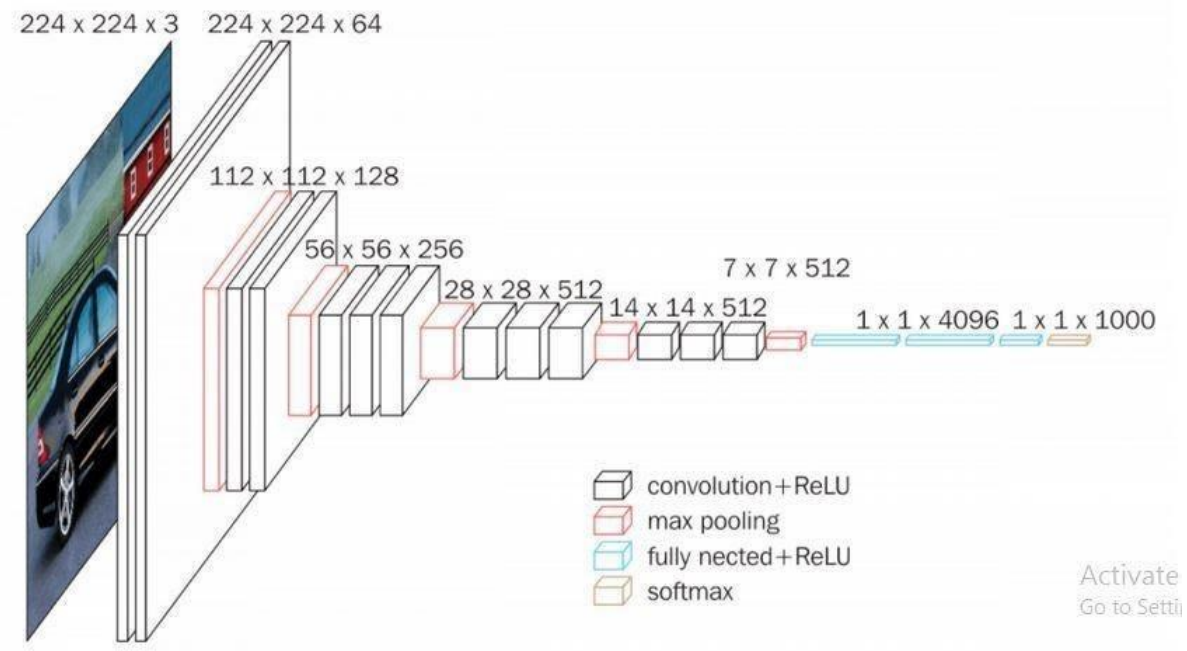
Dog Breed Classifier consists of advanced convolution neural network. Algorithm design of convolution network uses an insight knowledge of machine learning as well as neural networks algorithms. Deep convolution neural networks use advanced filters to detect the patterns that are present within the image. We will make a filter, that will be randomized using various distributions like Gaussian, normal, uniform distribution. So, we place a randomized filter on an image and figure out the pattern. Multiple filters, max-pooling layers, fully connected pooling layers, and dense layers are used to detect the pattern in an image. Activation functions and dropout also played a role in the development of our deep convolution network system that is quite robust. We have constructed a model that contains several layered architectures and it is combined with the and passed our dataset through the model to adjust the weights by attaching some of the data in front of the VGG16 and ResNet50 models.

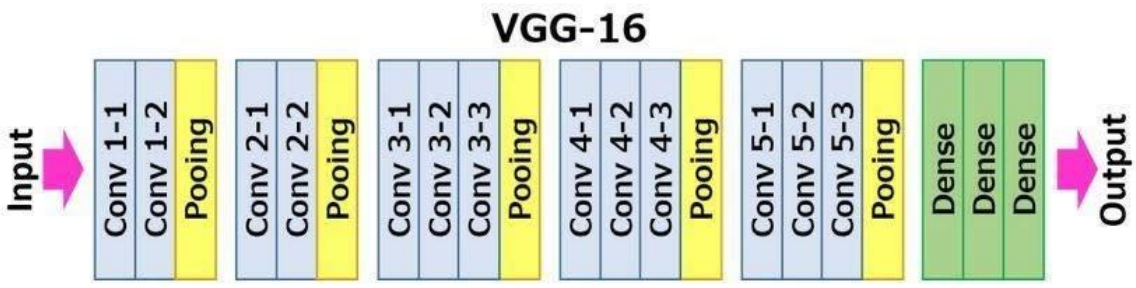
VGG16:

VGG16 is a convolution neural network (CNN) architecture that's considered to be one of the best vision model architectures to date. Instead of having a large number of hyper-parameters, VGG16 uses convolution layers with a 3x3 filter and a stride 1 that are in the same padding and maxpool layer of 2x2 filter of stride 2. It follows this arrangement of convolution and max pool layers consistently throughout the whole architecture. In the end it has two fully connected layers, followed by a softmax for output. The 16 in VGG16 refers to it has 16 layers that have weights. This network is a pretty large network, and it has about 138 million (approx) parameters.

VGG16 is a type of CNN (Convolutional Neural Network) that is considered to be one of the best computer vision models to date. The creators of this model evaluated the networks and increased the depth using an architecture with very small (3 × 3) convolution filters, which showed a significant improvement on the prior-art configurations. They pushed the depth to 16–19 weight layers making it approx — 138 trainable parameters.

VGG16 Architecture





ResNet50

The original ResNet architecture was ResNet-34, which comprised 34 weighted layers. It provided a novel way to add more convolutional layers to a CNN, without running into the vanishing gradient problem, using the concept of shortcut connections. A shortcut connection "skips over" some layers, converting a regular network to a residual network.

The regular network was based on the VGG neural networks (VGG-16 and VGG-19)—each convolutional network had a 3×3 filter. However, a ResNet has fewer filters and is less complex than a VGGNet. A 34-layer ResNet can achieve a performance of 3.6 billion FLOPs, and a smaller 18-layer ResNet can achieve 1.8 billion FLOPs, which is significantly faster than a VGG-19 Network with 19.6 billion FLOPs (read more in the ResNet paper, He et, al, 2015).

The ResNet architecture follows two basic design rules. First, the number of filters in each layer is the same depending on the size of the output feature map. Second, if the feature map's size is halved, it has double the number of filters to maintain the time complexity of each layer.

Special Characteristics of ResNet50

ResNet-50 has an architecture based on the model depicted above, but with one important difference. The 50-layer ResNet uses a bottleneck design for the building block. A bottleneck residual block uses 1×1 convolutions, known as a "bottleneck", which reduces the number of parameters and matrix multiplications. This enables much faster training of each layer. It uses a stack of three layers rather than two layers.

The 50-layer ResNet architecture includes the following elements, as shown in the table below:

- **A 7×7 kernel convolution** alongside 64 other kernels with a 2-sized stride.
- **A max pooling layer** with a 2-sized stride.
- **9 more layers**—3×3,64 kernel convolution, another with 1×1,64 kernels, and a third with 1×1,256 kernels. These 3 layers are repeated 3 times.
- **12 more layers** with 1×1,128 kernels, 3×3,128 kernels, and 1×1,512 kernels, iterated 4 times.
- **18 more layers** with 1×1,256 cores, and 2 cores 3×3,256 and 1×1,1024, iterated 6 times.
- **9 more layers** with 1×1,512 cores, 3×3,512 cores, and 1×1,2048 cores iterated 3 times. (up to this point the network has 50 layers)

- **Average pooling**, followed by a fully connected layer with 1000 nodes, using the softmax activation function.

ResNet50 Architecture

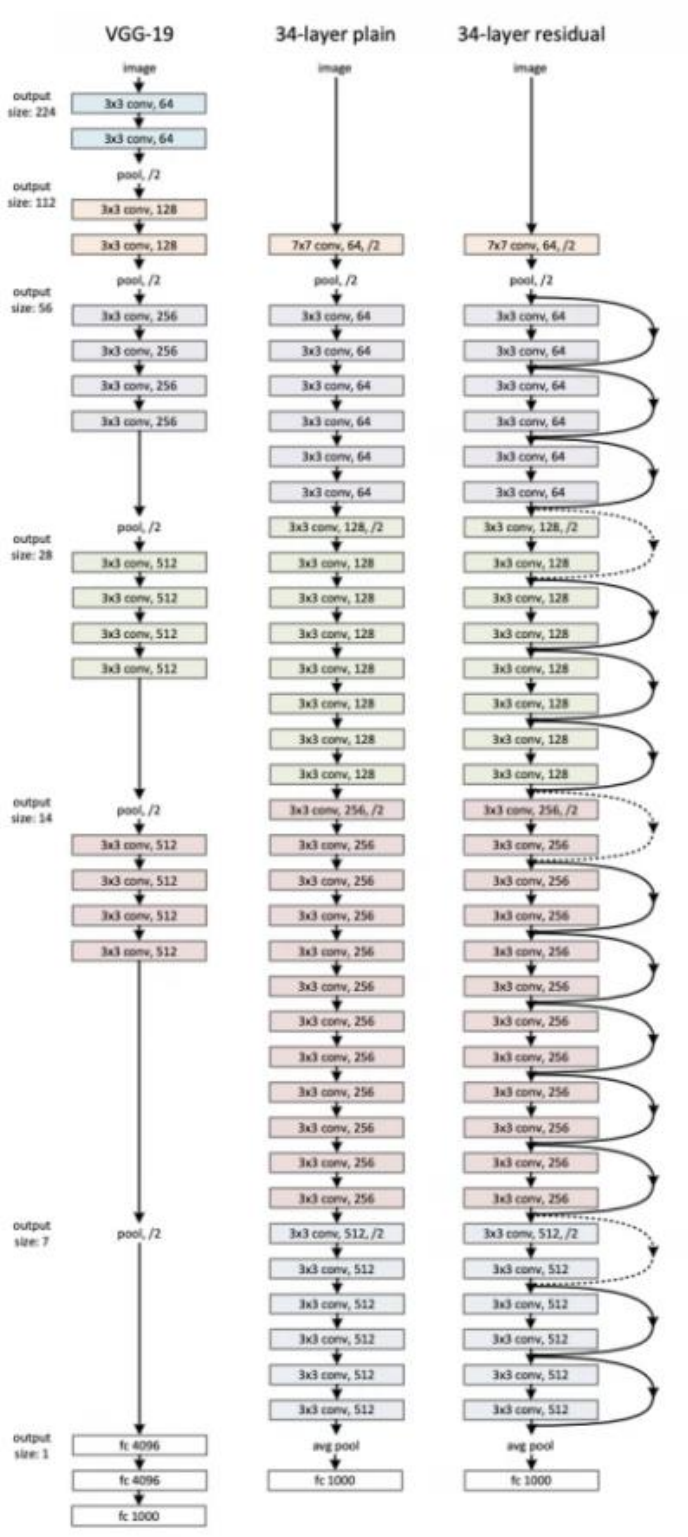
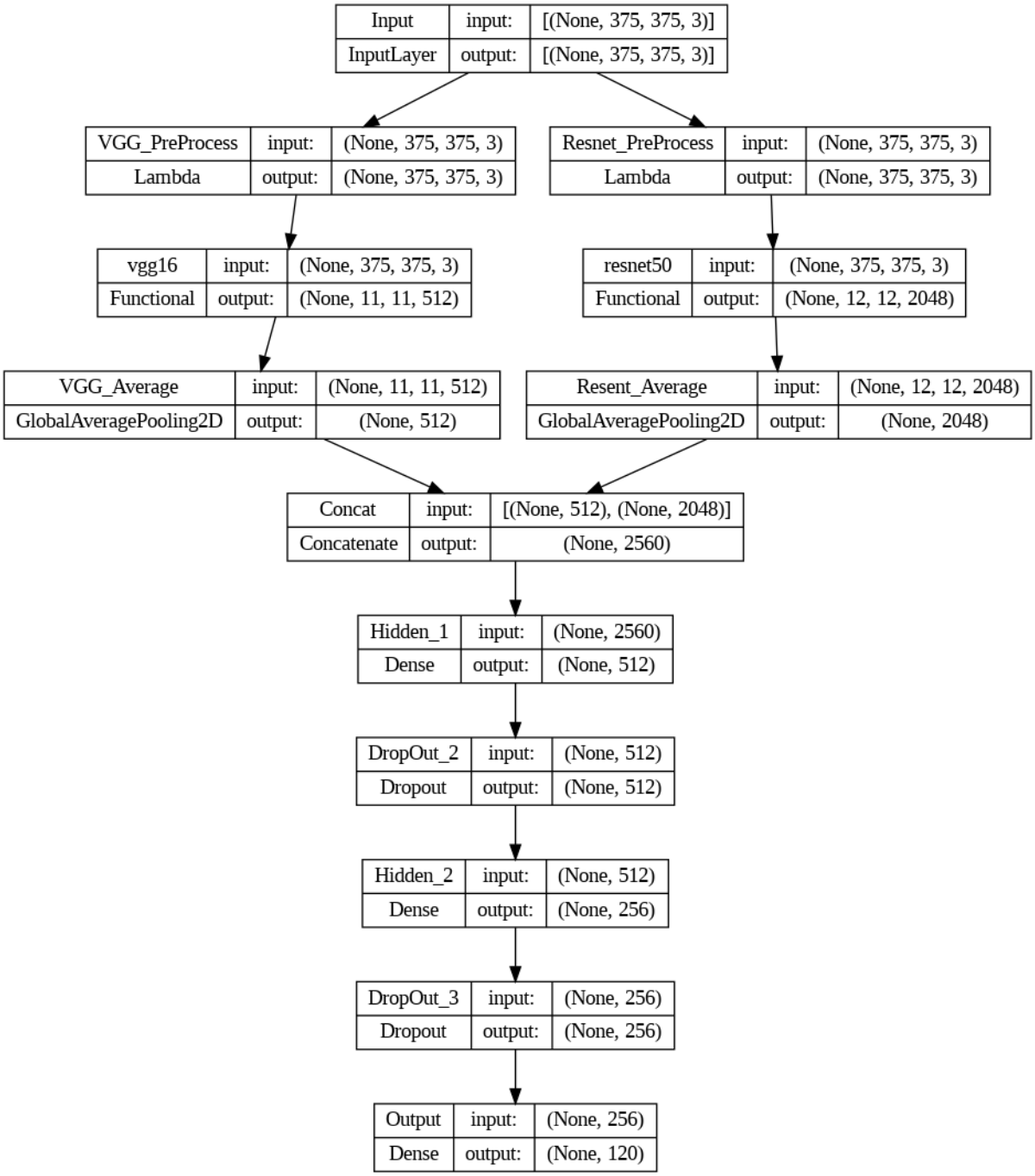


Figure 3. Example network architectures for ImageNet. **Left**: the VGG-19 model [41] (19.6 billion FLOPs) as a reference. **Middle**: a plain network with 34 parameter layers (3.6 billion FLOPs). **Right**: a residual network with 34 parameter layers (3.6 billion FLOPs). The dotted shortcuts increase dimensions. **Table 1** shows more details and other variants.

PROPOSED WORKING MODEL



This Proposed model is built in following steps:

1. Data Collection
2. Data Pre-Processing
3. Model Building

1. Data Collection

We make use of the Dog Breed data available on Kaggle for Competitions.

In [5]:
```
! kaggle competitions download -c dog-breed-identification
```

```
Warning: Your Kaggle API key is readable by other users on this system! To fix this, you can run 'chmod 600 /root/.kaggle/ka
ggle.json'
Downloading dog-breed-identification.zip to /content
100% 690M/691M [00:07<00:00, 130MB/s]
100% 691M/691M [00:07<00:00, 98.0MB/s]
```

In [6]:
```
! unzip -q dog-breed-identification.zip
```

In [7]:
```

df=pd.read_csv("labels.csv")
```

In [8]:
```
df.sample(5)
```

Out[8]:

	id	breed
6257	9e13beffb5b1aeda1aceeec82648b058	shetland_sheepdog
6194	9c10668d7a4038549c8349f9037bf33d	pekinese
3809	5e39f1e798ab4ec2b7264d07591e62f0	gordon_setter
9406	ebfc57d72a4167ec2f738f2adec52e31	toy_poodle
3190	4ec829615f7cc33a25c62e8f111db50d	english_setter

2. Data Pre-Processing:

We will Pre-Process the data using **Image Data Generator (IDG).**

In [23]:
```
idg=tf.keras.preprocessing.image.ImageDataGenerator(validation_split=0.1, rotation_range=30, horizontal_flip=True)
```

In [24]:
```
train_idg=idg.flow_from_dataframe(dataframe=df,
                                  directory='train/',
                                  x_col='id',
                                  y_col='breed',
                                  target_size=input_shape,
                                  batch_size=64,
                                  subset='training',
                                  )
```

```
Found 9200 validated image filenames belonging to 120 classes.
```

In [25]:
```
val_idg=idg.flow_from_dataframe(dataframe=df,
                                directory='train/',
                                x_col='id',
                                y_col='breed',
                                target_size=input_shape,
                                batch_size=64,
                                subset='validation',
                                )
```

```
Found 1022 validated image filenames belonging to 120 classes.
```

3. Model Building

We will build the model by using VGG16 and Resnet50 models. After building model we will save the model architecture and its weights

In [35]:
```
model_json=final_model.to_json()
with open("model.json", "w") as json_file:
    json_file.write(model_json)
final_model.save_weights("model.h5")
```

CHAPTER 4

SOURCE CODE

model.py

```
import tensorflow as tf

input_shape=(375, 375)
input=tf.keras.layers.Input((375, 375, 3), name="Input")
'''2 VGG Model'''
#2.1 Adding VGG_PreProcess
vgg_pre_process=tf.keras.layers.Lambda(tf.keras.applications.vgg16.preprocess_
input, name='VGG_PreProcess') (input)
#2.2 Downloading VGG Model
vgg_model=tf.keras.applications.VGG16(include_top=False, input_shape=(375,
375, 3))
vgg_model.trainable=False
vgg=vgg_model (vgg_pre_process)

#2.3 #Adding VGG GAP Layer

'''3 REsNet Model'''
vgg_avg=tf.keras.layers.GlobalAveragePooling2D(name="VGG_Average") (vgg)
#3.1 ResNet PreProcess
resnet_pre_process=tf.keras.layers.Lambda(tf.keras.applications.resnet50.prepr
ocess_input, name="Resnet_PreProcess") (input)
#3.2 #Downloading ResNet Model
resnet_model=tf.keras.applications.ResNet50(include_top=False,
input_shape=(375, 375, 3))
resnet_model.trainable=False
resnet= resnet_model (resnet_pre_process)
#3.3 #Adding ResNet GAP Layer
resnet_avg=tf.keras.layers.GlobalAveragePooling2D(name="Resent_Average")
(resnet)

'''4 Concatinating VGG and ResNet'''
concat=tf.keras.layers.Concatenate( name="Concat") ([vgg_avg, resnet_avg])

dense1=tf.keras.layers.Dense(512, activation="relu", name="Hidden_1") (concat)
drop2=tf.keras.layers.Dropout(0.25, name="DropOut_2") (dense1)
dense2=tf.keras.layers.Dense(256, activation="relu", name="Hidden_2") (drop2)
drop3=tf.keras.layers.Dropout(0.25, name="DropOut_3") (dense2
output=tf.keras.layers.Dense(120, activation="softmax", name="Output") (drop3)

final_model=tf.keras.models.Model(inputs=[input],
                                  outputs=[output])
```

app.py

```
#importing Necessary Libraries

import streamlit as st
import tensorflow as tf
from tensorflow.keras.preprocessing import image
from tensorflow.keras.models import load_model
import numpy as np
from PIL import Image
from labels import class_names
from model import final_model as model

st.set_option('deprecation.showfileUploaderEncoding', False)
st.title('Dog Breed Prediction')
st.text('Upload the Image')


input_shape=(375, 375)
model.load_weights("model.h5") #Loading Weights from pre trained Model

#Accepting Image to upload
uploaded_file = st.file_uploader("Choose an Image...",type='JPG')
st.write("Upload Any Breed", class_names)
if uploaded_file is not None:
    img=Image.open(uploaded_file) #Opening the given Image
    st.image(img,caption='Uploaded Image')

    if st.button('PREDICT'):
        st.write('Result ...')
        #Resizing the image to our model accepted input shape
        img=img.resize(input_shape)
        st.image(img)
        test_image = image.img_to_array(img) #Converting the Image to Array
        #Expanding Dimentions from (375, 375, 3) --> (1, 375, 375, 3)
        test_image = np.expand_dims(test_image, axis = 0)
        ##Predicting the image using pre Trained DL Model
        prediction = model.predict(test_image,batch_size=32)
        #exporting max probability element/class/breed
        st.write("Predicted label : ", class_names[np.argmax(prediction[0])])
        #Exporting the Probability the breed have
        confidence = round(100 * (np.max(prediction[0])), 2)
        st.write('Confidence : ',confidence)
```

labels.py

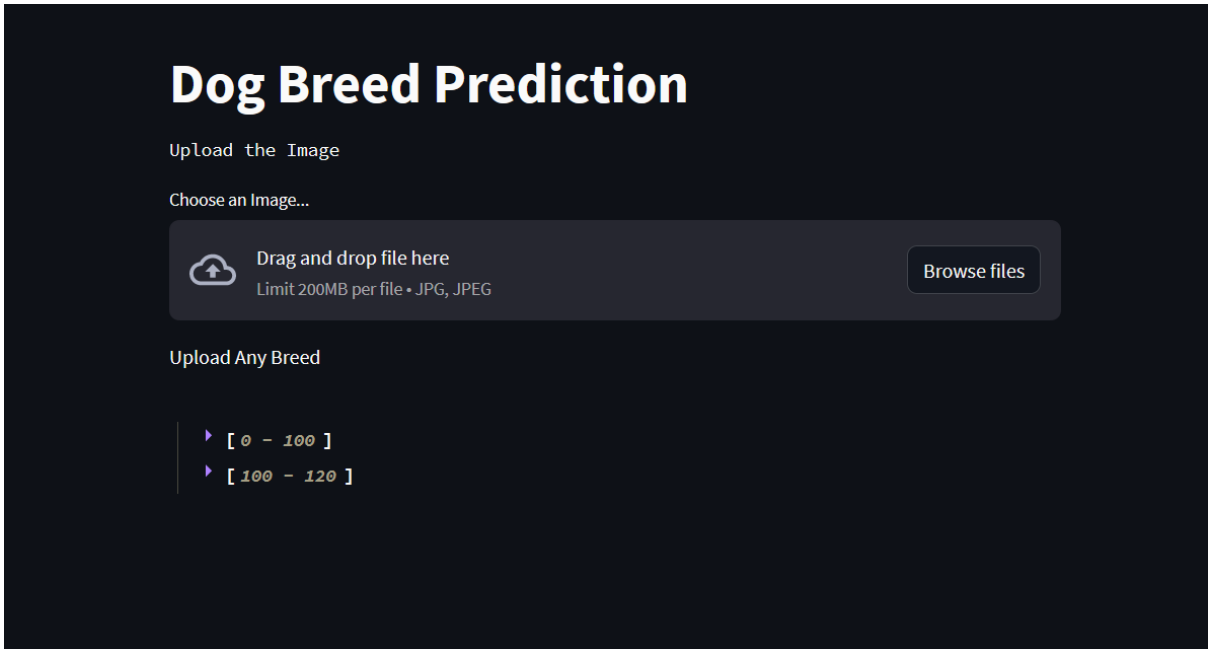
```
class_names=['affenpinscher',
 'afghan_hound',
 'african_hunting_dog',
 'airedale',
 'american_staffordshire_terrier',
 'appenzeller',
 'australian_terrier',
 'basenji',
 'basset',
 'beagle',
 'bedlington_terrier',
 'bernese_mountain_dog',
 'black-and-tan_coonhound',
 'blenheim_spaniel',
 'bloodhound',
 'bluetick',
 'border_collie',
 'border_terrier',
 'borzoi',
 'boston_bull',
 'bouvier_des_flandres',
 'boxer',
 'brabancon_griffon',
 'briard',
 'brittany_spaniel',
 'bull_mastiff',
 'cairn',
 'cardigan',
 'chesapeake_bay_retriever',
 'chihuahua',
 'chow',
 'clumber',
 'cocker_spaniel',
 'collie',
 'curly-coated_retriever',
 'dandie_dinmont',
 'dhole',
 'dingo',
 'doberman',
 'english_foxhound',
 'english_setter',
 'english_springer',
 'entlebucher',
 'eskimo_dog',
 'flat-coated_retriever',
 'french_bulldog',
```

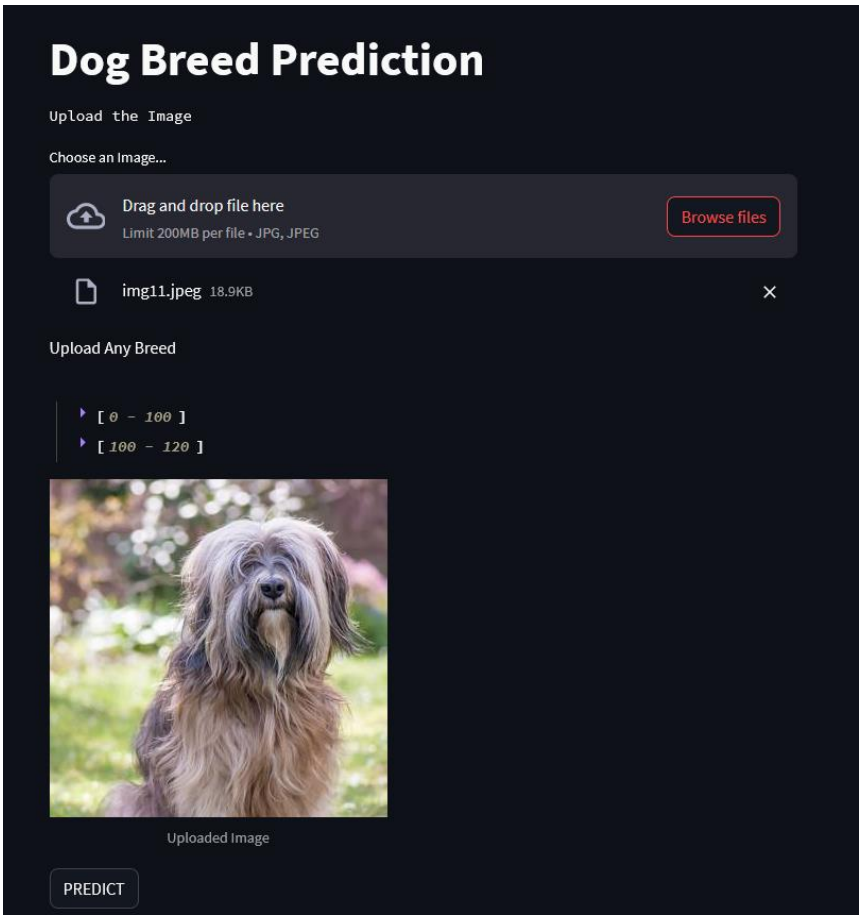
```
'german_shepherd',
'german_short-haired_pointer',
'giant_schnauzer',
'golden_retriever',
'gordon_setter',
'great_dane',
'great_pyrenees',
'greater_swiss_mountain_dog',
'groenendael',
'ibizan_hound',
'irish_setter',
'irish_terrier',
'irish_water_spaniel',
'irish_wolfhound',
'italian_greyhound',
'japanese_spaniel',
'keeshond',
'kelpie',
'kerry_blue_terrier',
'komondor',
'kuvasz',
'labrador_retriever',
'lakeland_terrier',
'leonberg',
'lhasa',
'malamute',
'malinois',
'maltese_dog',
'mexican_hairless',
'miniature_pinscher',
'miniature_poodle',
'miniature_schnauzer',
'newfoundland',
'norfolk_terrier',
'norwegian_elkhound',
'norwich_terrier',
'old_english_sheepdog',
'otterhound',
'papillon',
'pekinese',
'pembroke',
'pomeranian',
'pug',
'redbone',
'rhodesian_ridgeback',
'rottweiler',
'saint_bernard',
'saluki',
```

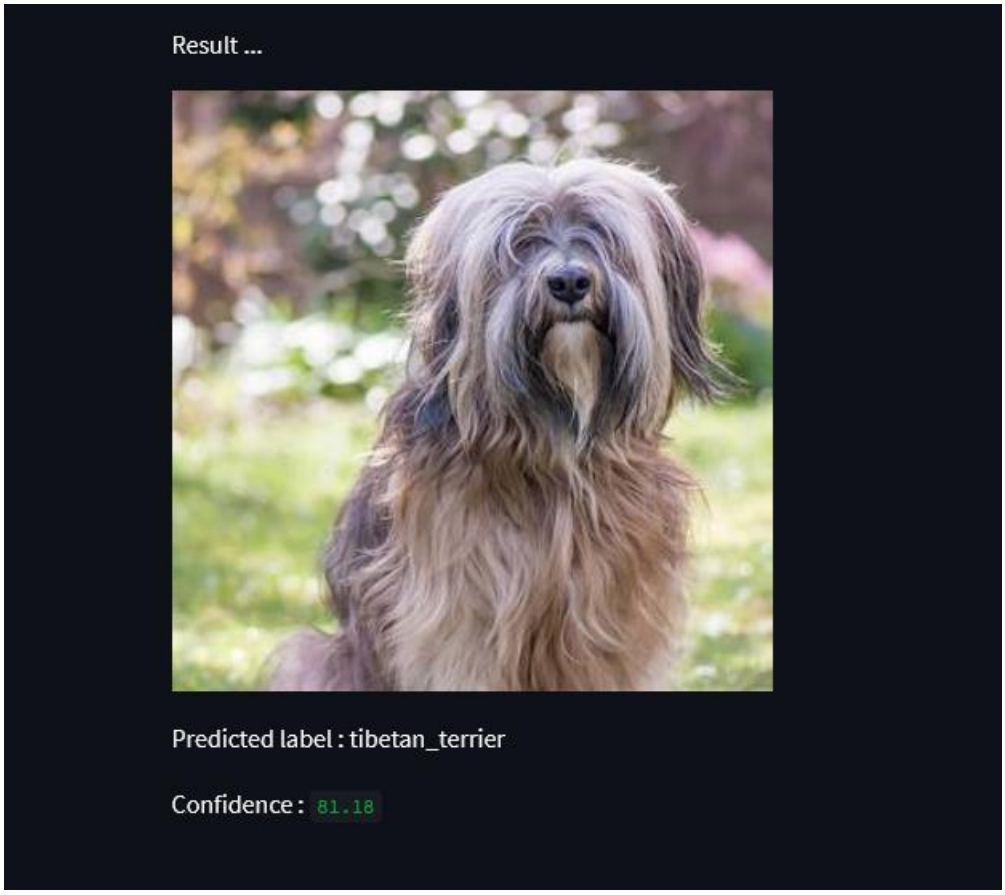
```
'samoyed',
'schipperke',
'scotch_terrier',
'scottish_deerhound',
'sealyham_terrier',
'shetland_sheepdog',
'shih-tzu',
'siberian_husky',
'silky_terrier',
'soft-coated_wheaten_terrier',
'staffordshire_bullterrier',
'standard_poodle',
'standard_schnauzer',
'sussex_spaniel',
'tibetan_mastiff',
'tibetan_terrier',
'toy_poodle',
'toy_terrier',
'vizsla',
'walker_hound',
'weimaraner',
'welsh_springer_spaniel',
'west_highland_white_terrier',
'whippet',
'wire-haired_fox_terrier',
'yorkshire_terrier']
```

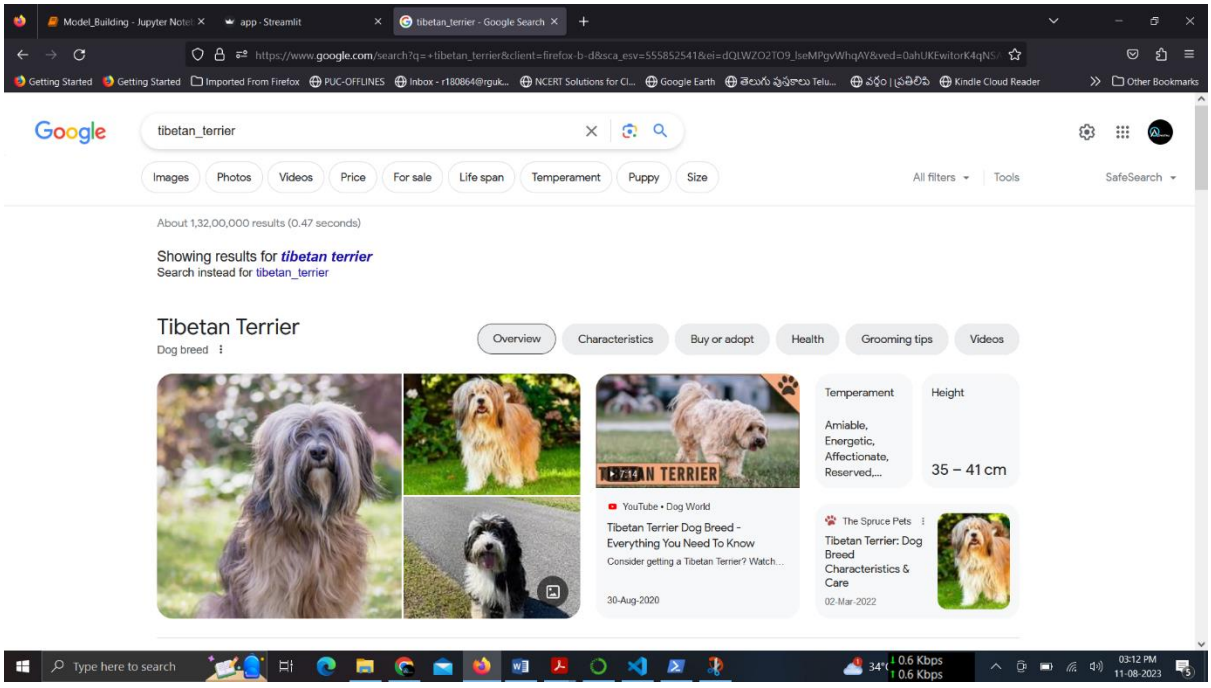
CHAPTER 5

RESULT



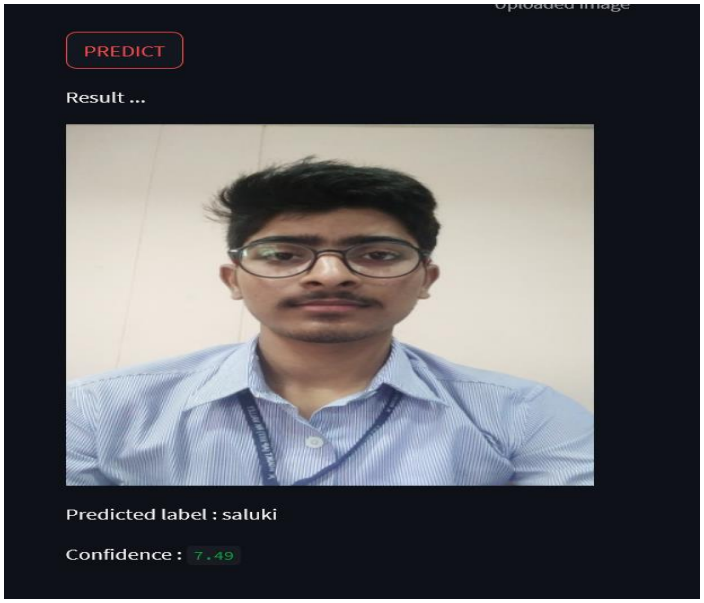






Model is giving the confidence 80% and above for all the 120 dog breeds.

If we give anything outout of the 120 dog breeds, ofcourse it will predict it as one of the 120 dog breeds. But the Confidence level is very low. Lets test it by giving human being Image.



Given the Confidence level very low that is 7.49.

CHAPTER 6

CONCLUSION

Overall, we consider our results to be a success given the high number of breeds in this fine-grained classification problem. We are able to effectively predict the correct breed over 50% of the time in one guess, a result that very few humans could match given the high variability both between and within the 120 different breeds contained in the dataset.

6.1 CONTRIBUTIONS

We see two major contributions to the literature in this project. First, this is the first time deep learning and convolutional neural networks have been used for dog face key- point detection according to the literature. Our key-point predictions were 4.62 pixels from the ground truth points on average. The accuracy of our key-point detection allowed us to succeed with our classification algorithms and is promising for future work in the area. Our vast experimentation with classification algorithms also provides novel contributions to the literature. The use of color histograms in feature extraction has never been done before; however, we found them to be unsuccessful due to the variety in color for individuals of the same breed. We also experimented with a variety of linear classifiers such as logistic regression and K-nearest neighbors for predicting dog breeds. These contributions add a variety of techniques to the literature for dog breed identification some which should be explored further as will be discussed next.

CHAPTER 7

SCOPE & FUTURE WORK

Future work should further explore the potential of convolutional neural networks in dog breed prediction. Given the success of our key-point detection network, this is a promising technique for future projects. That said, neural networks take an enormous time to train and we were unable to perform many iterations on our technique due to time constraints. We recommend further exploration into neural networks for key-point detection, specifically by training networks with a different architecture and batch iterator to see what approaches might have greater success. Also, given our success with neural networks and key-point detection, we recommend implementing a neural network for breed classification as well since this has not been performed in the literature. We were unable to experiment with this approach due to the time constraints of neural networks but believe that they would match if not improve upon our classification results. Ultimately, neural networks are time consuming to train and iterate upon, which should be kept in consideration for future efforts; still, neural networks are formidable classifiers that will increase prediction accuracy over more traditional techniques.

CHAPTER 8

REFERENCES

[1] A. Angelova and S. Zhu. Efficient object detection and seg- mentation for fine-grained recognition. 2013 Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2013.

[2] J. L. et al. Dog breed classification using part localization. Computer Vision: ECCV 2012, pages 172–185, 2012.

[3] N. Z. et al. Deformable part descriptors for fine-grained recognition and attribute prediction. 2013 IEEE International Conference on Computer Vision, 2013.

[4] N. Z. et al. Part-based r-cnns for fine-grained category detection. Computer Vision: ECCV 2014, pages 834–849, 2014.

[5] O. M. P. et al. Cats and dogs. 2012 IEEE Conference on Computer Vision and Pattern Recognition, 2012.

[6] P. N. B. et al. Searching the world's herbaria: A system for visual identification of plant species. 2008.

[7] P. N. B. et al. Localizing parts of faces using a consensus of exemplars, 2013.

[8] R. F. et al. Birdlets: Subordinate categorization using volumetric primitives and pose-normalized appearance. Pattern Recognition: 36th German Conference, GCPR 2014, 2014.

[9] E. Gavves. Fine-grained categorization by alignments, 2013.

[10] D. Nouri. Using convolutional neural nets to detect facial key-points tutorial, 2014