DOG BREED IDENTIFICATION

MINOR PROJECT REPORT

Submitted by

Saurabh Rajpurohit

In partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

At

FACULTY OF ENGINEERING, DESIGN AND AUTOMATION

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DECEMBER 2022

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the Minor Project entitled “DOG BREED IDENTIFICATION” by “Saurabh Rajpurohit” in partial fulfilment of requirements for the award of degree of B.Tech. (CSE) submitted in the Department of CSE at GNA University, Phagwara is an authentic record of my own work carried out during a period from AUGUST 2022 to DECEMBER 2022 under the supervision of **Er. Anu Arora.**

Signature of the Student
Saurabh Rajpurohit
GU-2019-3146

ABSTRACT

Dog has been a companion for this world and for human mankind for the last many years. But now looking around and seeing dogs we find that there are many types of dogs who look some way or the other way different from each other.

Since the classification of dogs is becoming very difficult and moreover, these classifications are taken on the deep learning concept and training a fully defined data set helps in training both models which predicts the different accuracy levels at both ends. Since every now and then predictions are taken for every model.

During the study, we came across many types of functionality levels that were not taken in previous studies too. Also, our approach also works on the main concept of transfer learning which deals with data augmentation technique with its properties to increase the size of data set, after which accuracy levels are matched.

ACKNOWLEDGEMENT

I would like to place on record my deep sense of gratitude to Er. Anu Arora, Assistant professor Department of Computer Science and Engineering, GNA University, Phagwara for his/her generous guidance, help and useful suggestions.

I express my sincere gratitude to DR. Anurag Sharma, HOD, Department of Computer Science and Engineering, GNA University, Phagwara, for his stimulating guidance, continuous encouragement and supervision throughout the course of present work.

I am extremely thankful to DR. Vikrant Sharma, Dean (FEDA-E) for providing me infrastructural facilities to work in, without which this work would not have been possible.

Saurabh Rajpurohit

GU-2019-3146

This is to certify that the above statement made by the candidate is correct to the best of my/our knowledge.

Signature of the SUPERVISOR (S)

The B.Tech Viva –Voce Examination of (SAURABH RAJPUROHIT) has been held on _____ and accepted.

Signature of External Examiner

Signature of H.O.D

LIST TO FIGURES

TABLE OF CONTENT

CHAPTER 1

INTRODUCTION

1.1General Introduction about Project

Dog has been a companion for this world and for human mankind for the last many years. But now looking around and seeing dogs we find that there are many types of dogs who look some way or the other way different from each other. Also, if we carry forward this, we found that there are many different kinds of dogs which are looking a bit similar but are of different breeds. These dogs are not only different in their breeds but they also differ in many of their characteristics such as their behavior towards humans, liking and disliking habits of each other, etc. Recently dogs can be classify using an expert-based approach. These are one who has a variety of knowledge of different breeds of dogs. Hopefully, this is very difficult as experts are not available, and secondly, there is a DNA approach. But this DNA approach is expensive and time taking as at present time there are a total of 20,580 dogs breed present in the world. In this project we're going to be using machine learning and deep learning to help us identify different breeds of dogs. To do this, we'll be using data from the Kaggle dog breed identification competition. It consists of a collection of 10,000+ labelled images of 120 different dog breeds. This kind of problem is called multi-class image classification. It's multi-class because we're trying to classify multiple different breeds of dog. Multi-class image classification is an important problem because it's the same kind of technology Tesla uses in their self-driving cars.

1.2 Methodology

In this Python Deep learning project, we will build a model for dog's breed identification using Machine Learning and Deep Learning. We will convert all Images of Dog's to tensors format using 'TensorFlow' and then feed them into the MobileNetV2, a special type of neural network under the Transfer Learning technique, which will help us to identify dog's breed. Transfer learning is a machine learning method where we reuse a pre-trained model as the starting point for a model on a new task. For performing these all tasks Google Colaboratory is used as Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education. More technically, Colab is a hosted Jupyter notebook service that Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education. More technically, Colab is a hosted Jupyter notebook service that requires no setup to use, while providing access free of charge to computing resources including GPUsquires no setup to use, while providing access free of charge to computing resources including GPUs.

1.3 Machine Learning

Machine Learning is the field of study that gives computers the capability to learn without being explicitly programmed. ML is one of the most exciting technologies that one would have ever come across. As it is evident from the name, it gives the computer that makes it more similar to humans: The ability to learn. Machine learning is actively being used today, perhaps in many more places than one would expect. Machine learning is a branch of artificial intelligence (AI) and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy. Machine learning is an important component of the growing field of data science. Through the use of statistical methods, algorithms are trained to make classifications or predictions, and to uncover key insights in data mining projects. These insights subsequently drive decision making within applications and businesses, ideally impacting key growth metrics. As big data continues to expand and grow, the market demand for data scientists will increase. They will be required to help identify the most relevant business questions and the data to answer them.

1.4 Deep Learning

Deep learning is a subset of machine learning, which is essentially a neural network with three or more layers. These neural networks attempt to simulate the behavior of the human brain allowing it to “learn” from large amounts of data. While a neural network with a single layer can still make approximate predictions, additional hidden layers can help to optimize and refine for accuracy. Deep learning drives many artificial intelligence (AI) applications and services that improve automation, performing analytical and physical tasks without human intervention. Deep learning technology lies behind everyday products and services (such as digital assistants, voice-enabled TV remotes, and credit card fraud detection) as well as emerging technologies (such as self-driving cars). As neural network is going to mimic the human brain so deep learning is also a kind of mimic of human brain. In deep learning, we don’t need to explicitly program everything. The concept of deep learning is not new. It has been around for a couple of years now. It’s on hype nowadays because earlier we did not have that much processing power and a lot of data. As in the last 20 years, the processing power increases exponentially, deep learning and machine learning came in the picture. Deep learning is a particular kind of machine learning that achieves great power and flexibility by learning to represent the world as a nested hierarchy of concepts, with each concept defined in relation to simpler concepts, and more abstract representations computed in terms of less abstract ones.

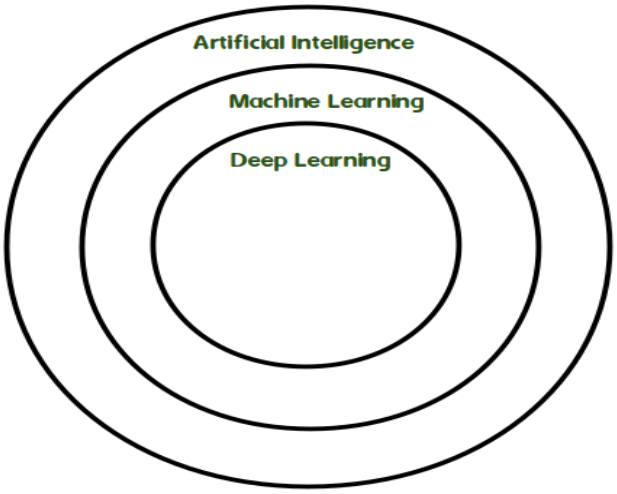

fig 1.1 Sets of AI

1.5 TensorFlow

TensorFlow is a free and open-source software library for machine learning and artificial intelligence. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks. TensorFlow was developed by the Google Brain team for internal Google use in research and production The initial version was released under the Apache License 2.0 in 2015. It is entirely based on Python programming language and use for numerical computation and data flow, which makes machine learning faster and easier. TensorFlow can train and run the deep neural networks for image recognition, handwritten digit classification, recurrent neural network, word embedding, natural language processing, video detection, and many more. TensorFlow is run on multiple CPUs or GPUs and also mobile operating systems.

Features of TensorFlow:

1) Flexible

It is one of the essential TensorFlow Features according to its operability. It has modularity and parts of it which we want to make standalone.

2) Easily Trainable

It is easily trainable on CPU and for GPU in distributed computing

3) Open Source

The best thing about the machine learning library is that it is open source so anyone can use it as much as they have internet connectivity. So, people can manipulate the library and come up with a fantastic variety of useful products. And it has become another DIY community which has a massive forum for people getting started with it and those who find it hard to use it.

1.6 Google Colab

Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education. More technically, Colab is a hosted Jupyter notebook service that Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education. More technically, Colab is a hosted Jupyter notebook service that requires no setup to use, while providing access free of charge to computing resources including GPUsquires no setup to use, while providing access free of charge to computing resources including GPUs. Google Colab is an excellent tool for deep learning tasks. Colab notebooks are portable and can be shared to anyone and hence operated by anyone from anywhere throughout the web.

Operations in Google colab:

- Write and execute code in Python
- Document your code that supports mathematical equations
- Create/Upload/Share notebooks
- Import/Save notebooks from/to Google Drive
- Import/Publish notebooks from GitHub
- Import external datasets e.g. from Kaggle
- Integrate PyTorch, TensorFlow, Keras, OpenCV
- Free Cloud service with free GPU

Hardware & Software Requirement

System requirements are the configuration that a system must have in order for a hardware or software application to run smoothly and efficiently. Failure to meet these requirements can result in installation problems or performance problems. System requirements are also known as minimum system requirements.

Software Requirement

- OS: Window XP, 7, 8, 10 / Linux / Mac OS
- Visual Studio Code / Jupyter Notebook / Google Colab etc.
- Web Browsers if using Google Collab.

Hardware Requirement

- Processor: x86 or x64
- RAM: 512 MB (minimum), 1 GB (recommended)
- Hard disc: up to 3 GB of free space may be required
- GPU

Runtime Environment for Prediction Model

1. Python

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built-in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance.

2. Pandas

Pandas is an open-source Python library that is made mainly for working with relational or labelled data both easily and intuitively. It provides various data structures and operations for manipulating numerical data and time series. Pandas is fast and it has high performance & productivity for users.

- Import pandas as pd

3. NumPy

NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, transform, and matrices.NumPy was created in 2005 by Travis Oliphant. It is an open-source project and you can use it freely. NumPy stands for Numerical Python.

- Import numpy as np

4. Matplotlib

Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays. One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram etc.

- Import matplotlib.pyplot as plt

5. Scikit-learn

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib.

6. TensorFlow

TensorFlow is a free and open-source software library for machine learning and artificial intelligence. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks.

- Import tensorflow as tf

7. TensorFlow hub

TensorFlow Hub is a comprehensive repository of pre-trained models ready for fine-tuning and deployable anywhere. Download the latest trained models with a minimal amount of code with the tensorflow_hub library.

- Import tensorflow_hub as hub

8. MobileNetv2

MobileNet-v2 is a convolutional neural network that is 53 layers deep. You can load a pretrained version of the network trained on more than a million images from the ImageNet database. The pretrained network can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals. As a result, the network has learned rich feature representations for a wide range of images. The network has an image input size of 224-by-224.

9. Keras

Keras is an open-source high-level Neural Network library, which is written in Python is capable enough to run on Theano, TensorFlow, or CNTK. It was developed by one of the Google engineers, Francois Chollet. It is made user-friendly, extensible, and modular for facilitating faster experimentation with deep neural networks. It not only supports Convolutional Networks and Recurrent Networks individually but also their combination.

CHAPTER 2

OBJECTIVES

1. **To get breed name of dog just with picture**.
   Through this project, the breed prediction is made easy. The Deep learning process is fast and accurate.

2. **To provide this prediction facility to everyone.**
   The prediction model can be shared and made available for every single person throughout the web.

3. **To make classification easy for dog competition.**
   Through the Dog Breed Prediction model, classification of dogs has become easier to enroll dogs in contest.

4. **The breeding process is made smoother.**
   The model can be used to predict dogs breed in various breeding clinics so, that correct breeding could be done.

5. **To Make the dog selling business smoother.**
   With the help of the prediction model, correct dog breed could be predicted and buy and selling of dog could be made authentic.

6. **To help in various veterinary operations.**
   Medical practitioner can also take help of the project to insure the treatment they are using on the dog as dogs habitations and habits may differ as per their breeds.

CHAPTER 3

WORKFLOW

We're going to go through the following TensorFlow/Deep Learning workflow:

1. Get data ready.
2. Choose a model to suit the problem.
3. Fit/train a model.
4. Evaluating a model.
5. Improve the model through experimentation.
6. Save, sharing and reloading your model.

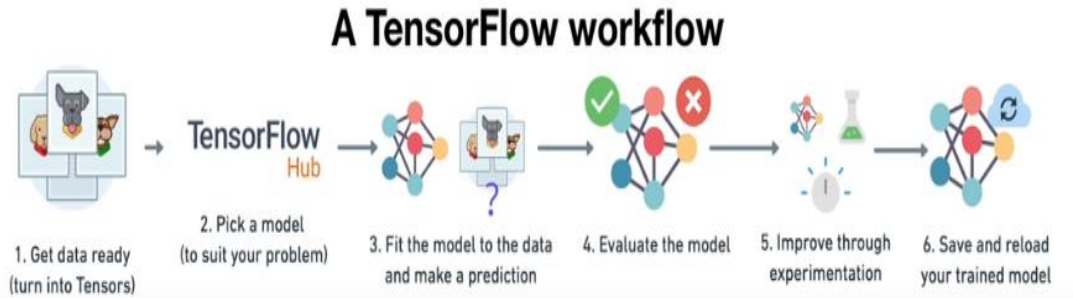

fig 3.1 TensorFlow workflow

3.1 Getting data ready.

The very first thing to start prediction projects the data should be arranged whether it can be download from Kaggle, store, or imported from Google Drive by mounting the drive to the Colab notebook for further operations. Since much of machine learning is getting your data ready to be used with a machine learning model, we'll take extra care getting it setup.

3.1.1 Accessing the data

Now the data files we're working with are available on our Google Drive, we can start to check it out. Let's start with labels.csv which contains all of the image ID's and their associated dog breed (our data and labels).

```
# Checkout the labels of our data
import pandas as pd
labels_csv = pd.read_csv("drive/My Drive/Data/labels.csv")
print(labels_csv.describe())
print(labels_csv.head())
```

```
                                  id               breed
count                          10222               10222
unique                         10222                 120
top     45fef015b1974e98da0173e8260b3482  scottish_deerhound
freq                               1                 126
                                 id             breed
0  000bec180eb18c7604dcecc8fe0dba07       boston_bull
1  001513dfcb2ffafc82cccf4d8bbaba97             dingo
2  001cdf01b096e06d78e9e5112d419397          pekinese
3  00214f311d5d2247d5dfe4fe24b2303d          bluetick
4  0021f9ceb3235effd7fcde7f7538ed62  golden_retriever
```

Looking at this, we can see there are 10222 different ID's (meaning 10222 different images) and 120 different breeds.

fig 3.2 Info about data

3.1.2 Creating validation set

Since the dataset from Kaggle doesn't come with a validation set (a split of the data we can test our model on before making final predicitons on the test set), let's make one. We could use Scikit-Learn's train_test_split function or we could simply make manual splits of the data. Since we're working with 10,000+ images, it's a good idea to work with a portion of them to make sure things are working before training on them all. This is because computing with 10,000+ images could take a fairly long time. And our goal when working through machine learning projects is to reduce the time between experiments. Now let's split our data into training and validation sets. We'll use and 80/20 split (80% training data, 20% validation data). As we are training our model firstly on 1000 images so train and validation split will 800 images for training data and 200 images for validation set.

3.1.3 Pre-processing images

Pre-processing of images means turning images into Tensors, The term tensor is just a fancy word to represent arrays used in tensorflow for fast processing of image data. Our labels are in numeric format but our images are still just file paths. Since we're using TensorFlow, our data has to be in the form of Tensors.

A Tensor is a way to represent information in numbers. If you're familar with NumPy arrays (you should be), a Tensor can be thought of as a combination of NumPy arrays, except with the special ability to be used on a GPU. Because of how TensorFlow stores information (in Tensors), it allows machine learning and deep learning models to be run on GPUs (generally faster at numerical computing).

To pre-process our images into Tensors we're going to write a function which does a few things:

1. Takes an image filename as input.
2. Uses TensorFlow to read the file and save it to a variable, image.
3. Turn our image (a jpeg file) into Tensors.
4. Resize the image to be of shape (224, 224).
5. Return the modified image.

3.1.4 Creating data batches

Now we've got a function to convert our images into Tensors, we'll now build one to turn our data into batches (more specifically, a TensorFlow BatchDataset). A batch (also called mini-batch) is a small portion of your data, say 32 (32 is generally the default batch size) images and their labels. In deep learning, instead of finding patterns in an entire dataset at the same time, you often find them one batch at a time.

Let's say you're dealing with 10,000+ images (which we are). Together, these files may take up more memory than your GPU has. Trying to compute on them all would result in an error. Instead, it's more efficient to create smaller batches of your data and compute on one batch at a time. TensorFlow is very efficient when your data is in batches of (image, label) Tensors. So we'll build a function to do create those first.

3.2 Pick a Model

Now our data is ready, let's prepare it modelling. We'll use an existing model from TensorFlow Hub. TensorFlow Hub is a resource where you can find pretrained machine learning models for the problem you're working on. Using a pretrained machine learning model is often referred to as transfer learning. Building a machine learning model and training it on lots from scratch can be expensive and time consuming. Transfer learning helps eliviate some of these by taking what another model has learned and using that information with your own problem. Since we know our problem is image classification (classifying different dog breeds), we can navigate the TensorFlow Hub page by our problem domain (image). We start by choosing the image problem domain, and then can filter it down by subdomains, in our case, image classification. Doing this gives a list of different pretrained models we can apply to our task. Clicking on one gives us information about the model as well as instructions for using it. For example, clicking on the mobilenet_v2_130_224 model, tells us this model takes an input of images in the shape 224, 224. It also says the model has been trained in the domain of image classification.

3.3 Fitting/Training the model

3.3.1 Building the model

Before we build a model, there are a few things we need to define:

- The input shape (images, in the form of Tensors) to our model.
- The output shape (image labels, in the form of Tensors) of our model.
- The URL of the model we want to use.

Now we've got the inputs, outputs and model we're using ready to go. We can start to put them together. There are many ways of building a model in TensorFlow but one of the best ways to get started is to use the Keras API. Defining a deep learning model in Keras can be as straightforward as saying, "here are the layers of the model, the input shape and the output shape, let's go!"

Knowing this, creating a function which:

- Takes the input shape, output shape and the model we've chosen's URL as parameters.
- Defines the layers in a Keras model in a sequential fashion (do this first, then this, then that).
- Compiles the model (says how it should be evaluated and improved).
- Builds the model (tells it what kind of input shape it'll be getting).
- Returns the model.

We use model.build() whenever we're using a layer from TensorFlow Hub to tell our model what input shape it can expect. In this case, the input shape is [None, IMG_SIZE, IMG_SIZE, 3] or [None, 224, 224, 3] or [batch_size, img_height, img_width, color_channels]. Batch size is left as None as this is inferred from the data we pass the model. In our case, it'll be 32 since that's what we've set up our data batches as.

3.3.2 Setting up the model layers

There are two ways to do this in Keras, the functional and sequential API. We've used the sequential. The Keras documentation states the functional API is the way to go for defining complex models but the sequential API (a linear stack of layers) is perfectly fine for getting started, which is what we're doing. The first layer we use is the model from TensorFlow Hub (hub.KerasLayer(MODEL_URL). So, our first layer is actually an entire model (many more layers). This **input layer** takes in our images and finds patterns in them based on the patterns mobilenet_v2_130_224 has found. The next layer (tf.keras.layers.Dense()) is the **output layer** of our model. It brings all of the information discovered in the input layer together and outputs it in the shape we're after, 120 (the number of unique labels we have). The activation="softmax" parameter tells the output layer, we'd like to assign a probability value to each of the 120 labels somewhere between 0 & 1. The higher the value, the more the model believes the input image should have that label. If we were working on a binary classification problem, we'd use activation="sigmoid". Loss is a measure of how well a model is learning (higher the loss worse the prediction). Optimizer lowers the loss, Adam optimizer is one of the best optimizer. Metrics judges how well it is peformoning(accuracy).

3.3.3 Creating callbacks

Callbacks are helper functions a model can use during training to do things such as save a models progress, check a models progress or stop training early if a model stops improving. The two callbacks we're going to add are a TensorBoard callback and an Early Stopping callback.

TensorBoard Callback

TensorBoard helps provide a visual way to monitor the progress of your model during and after training. It can be used directly in a notebook to track the performance measures of a model such as loss and accuracy.

To set up a TensorBoard callback and view TensorBoard in a notebook, we need to do three things:

1. Load the TensorBoard notebook extension.
2. Create a TensorBoard callback which is able to save logs to a directory and pass it to our model's fit() function.
3. Visualize the our models training logs using the %tensorboard magic function (we'll do this later on).

Early Stopping Callback

Early stopping helps prevent overfitting by stopping a model when a certain evaluation metric stops improving. If a model trains for too long, it can do so well at finding patterns in a certain dataset that it's not able to use those patterns on another dataset it hasn't seen before (doesn't generalize).

It's basically like saying to our model, "keep finding patterns until the quality of those patterns starts to go down."

3.3.4Training a model

Our first model is only going to be trained on 1000 images. Or trained on 800 images and then validated on 200 images, meaning 1000 images total or about 10% of the total data.

We do this to make sure everything is working. And if it is, we can step it up later and train on the entire training dataset.

The final parameter we'll define before training is NUM_EPOCHS (also known as **number of epochs**). EPOCHS is chances to go through training-dataset

NUM_EPOCHS defines how many passes of the data we'd like our model to do. A pass is equivalent to our model trying to find patterns in each dog image and see which patterns relate to each label.

3.4 Evaluating Model

Before we scale up and train on more data, let's see some other ways we can evaluate our model. Because although accuracy is a pretty good indicator of how our model is doing, it would be even better if we could could see it in action. Making predictions with a trained model is as calling predict() on it and passing it data in the same format the model was trained on. Making predictions with our model returns an array with a different value for each label. In this case, making predictions on the validation data (200 images) returns an array (predictions) of arrays, each containing 120 different values (one for each unique dog breed). These different values are the probabilities or the likelihood the model has predicted a certain image being a certain breed of dog. The higher the value, the more likely the model thinks a given image is a specific breed of dog. Now we've got a list of all different predictions our model has made, we'll do the same for the validation images and validation labels. Remember, the model hasn't trained on the validation data, during the fit() function, it only used the validation data to evaluate itself. So, we can use the validation images to visually compare our models predictions with the validation labels. Since our validation data (val_data) is in batch form, to get a list of validation images and labels, we'll have to unbatch it (using unbatch()) and then turn it into an iterator using as_numpy_iterator(). Now we've got ways to get:

- Prediction labels
- Validation labels (truth labels)
- Validation images

3.5 Improve the model through experimentation

start with 1000 images, make sure it works, increase the number of images as per experimentation requirements and further train the model on full data at last so that model’s performance could be improved.

3.6 Saving and reloading a model

After training a model, it's a good idea to save it. Saving it means you can share it with colleagues, put it in an application and more importantly, won't have to go through the potentially expensive step of retraining it. The format of an entire saved Keras model is h5. So we'll make a function which can take a model as input and utilise the save() method to save it as a file to a specified directory. If we've got a saved model, we'd like to load it, let's create a function which can take a model path and use the tf.keras.models.load_model()

function to load it into the notebook. Because we're using a component from TensorFlow Hub (hub.KerasLayer) we'll have to pass this as a parameter to the custom_objects parameter.

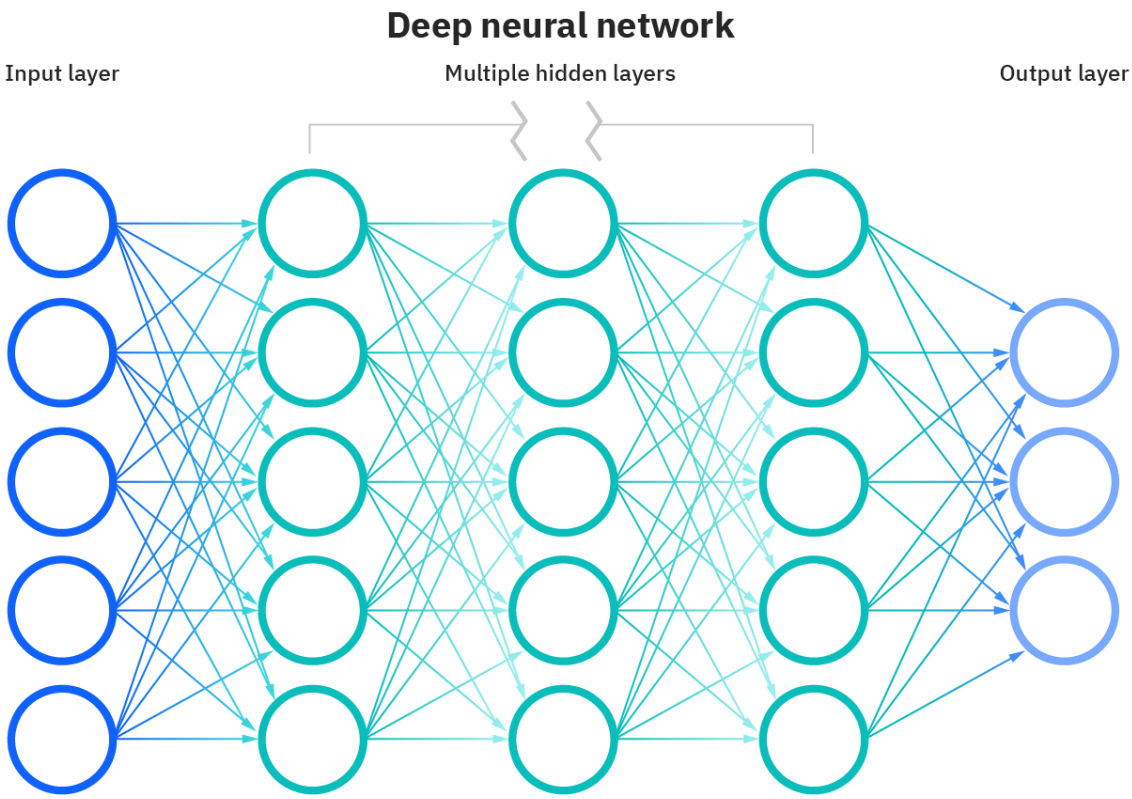

fig 3.3 Neural Network Architecture

Artificial neural networks (ANNs) are comprised of a node layers, containing an input layer, one or more hidden layers, and an output layer. Each node, or artificial neuron, connects to another and has an associated weight and threshold. If the output of any individual node is above the specified threshold value, that node is activated, sending data to the next layer of the network. Otherwise, no data is passed along to the next layer of the network. Neural networks rely on training data to learn and improve their accuracy over time. However, once these learning algorithms are fine-tuned for accuracy, they are powerful tools in computer science and artificial intelligence, allowing us to classify and cluster data at a high velocity.

CHAPTER 4

PROCESS AND RESULT

screen shots

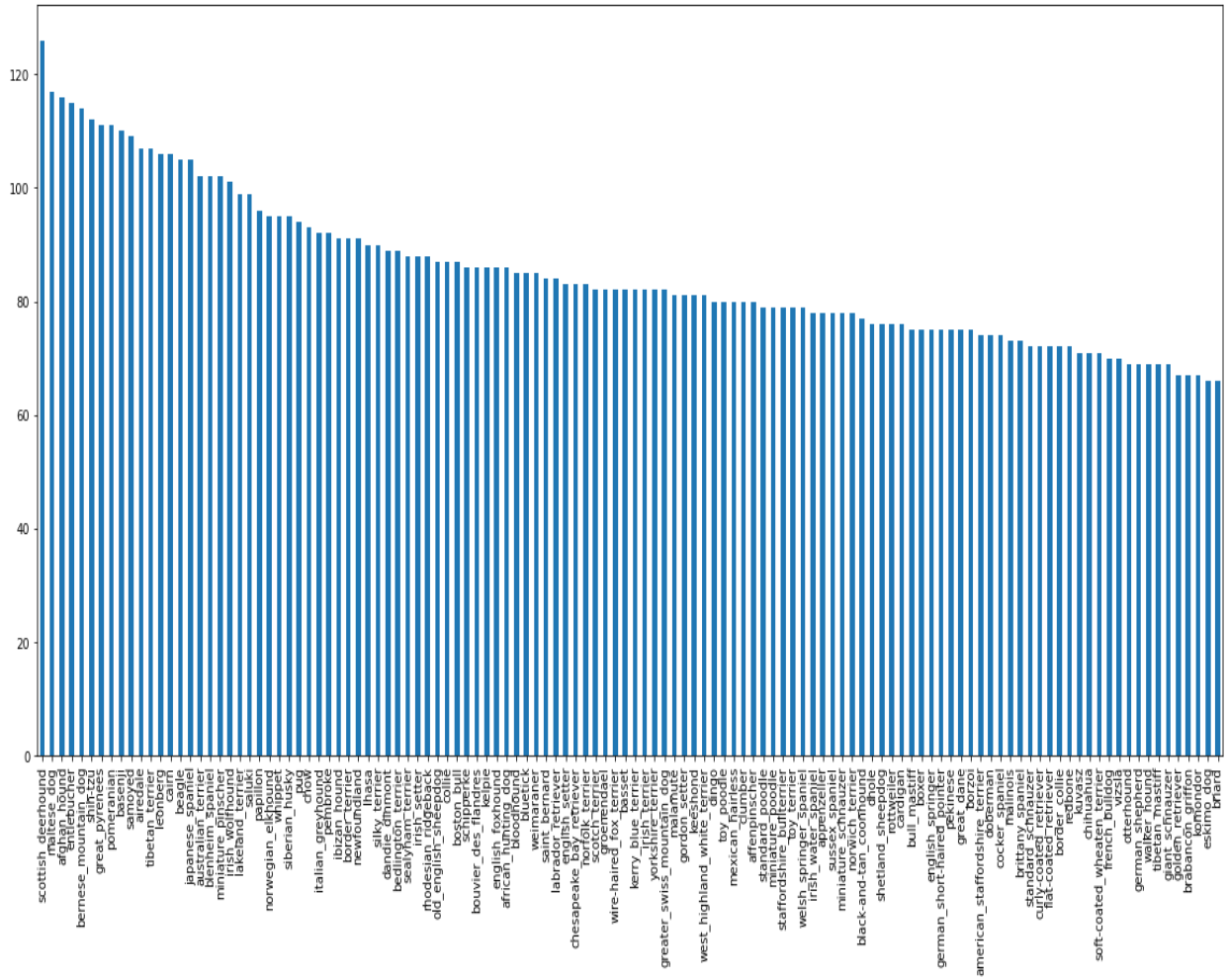

fig 4.1 Images per unique label

If we were to roughly draw a line across the middle of the graph, we'd see there's about 60+ images for each dog breed and on average 82 images for each breed. This is a good amount as for some of their vision products Google recommends a minimum of 10 images per class to get started. And as you might imagine, the more images per class available, the more chance a model has to figure out patterns between them.

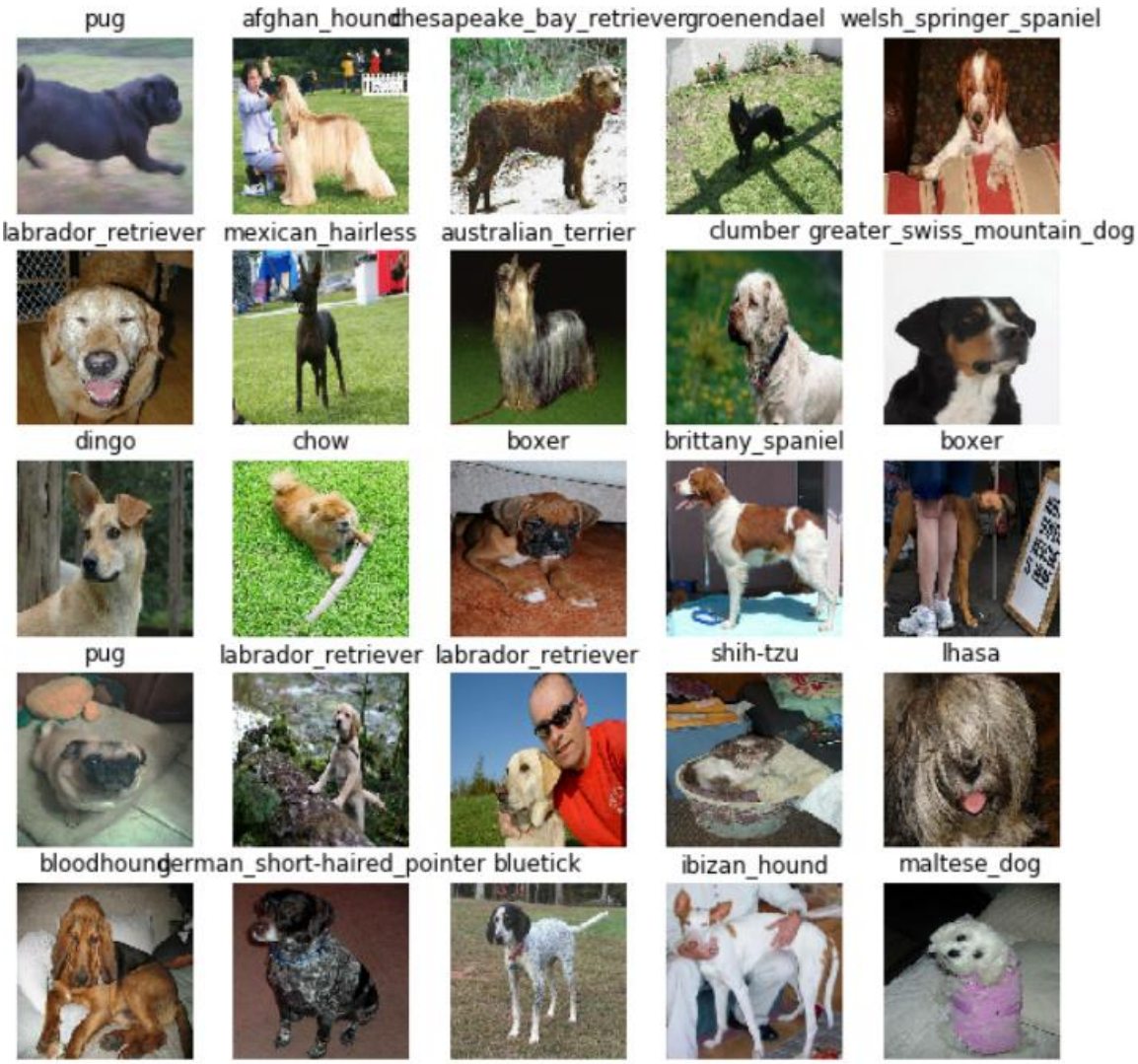

fig 4.2 Train Images

Visualize training images from the training data batch for a glance of the image

Data we have got for training the model

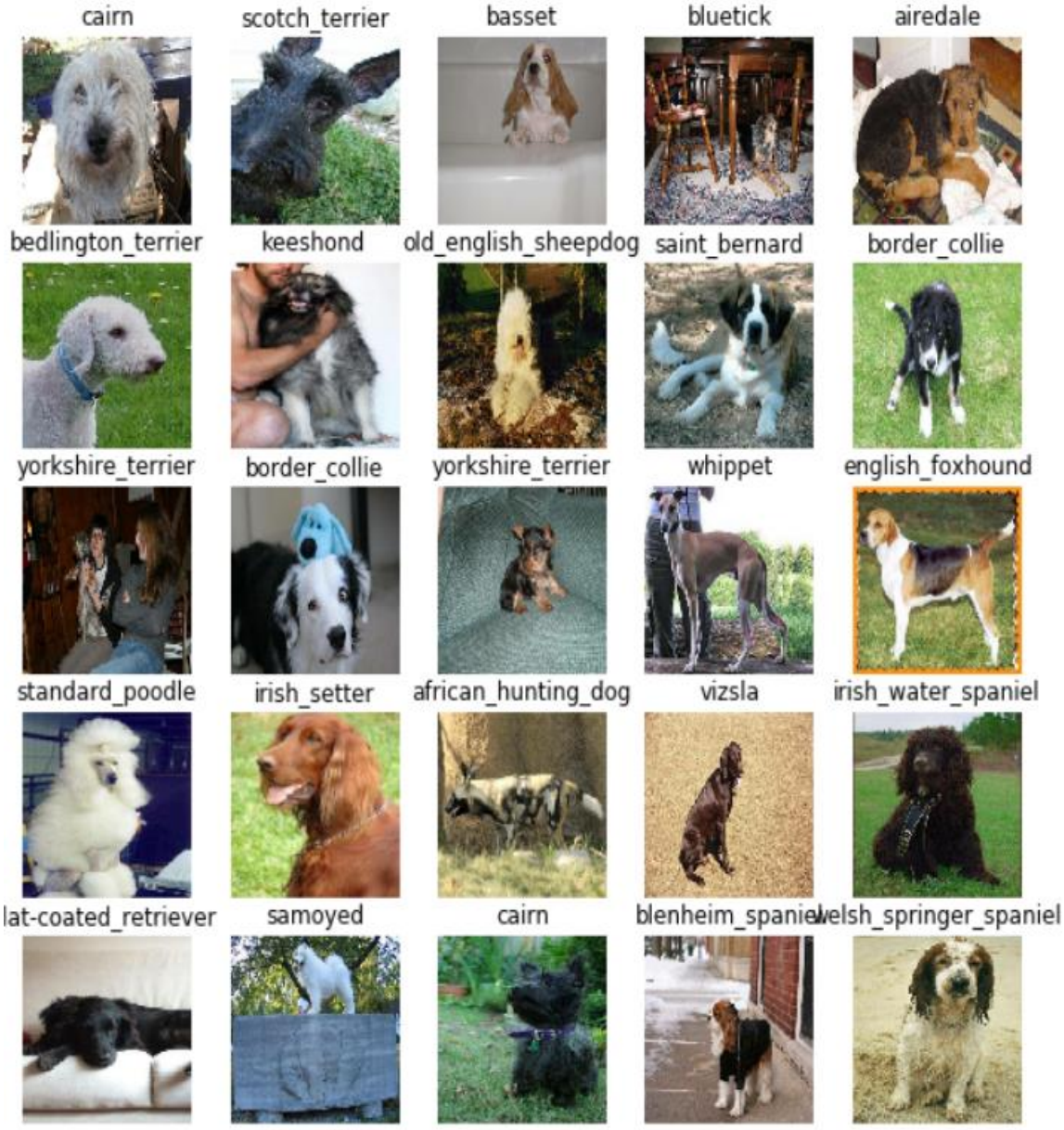

fig 4.3 Validation Images

Visualize validation images from the training data batch for a glance of the image

Data we have got for validating the model

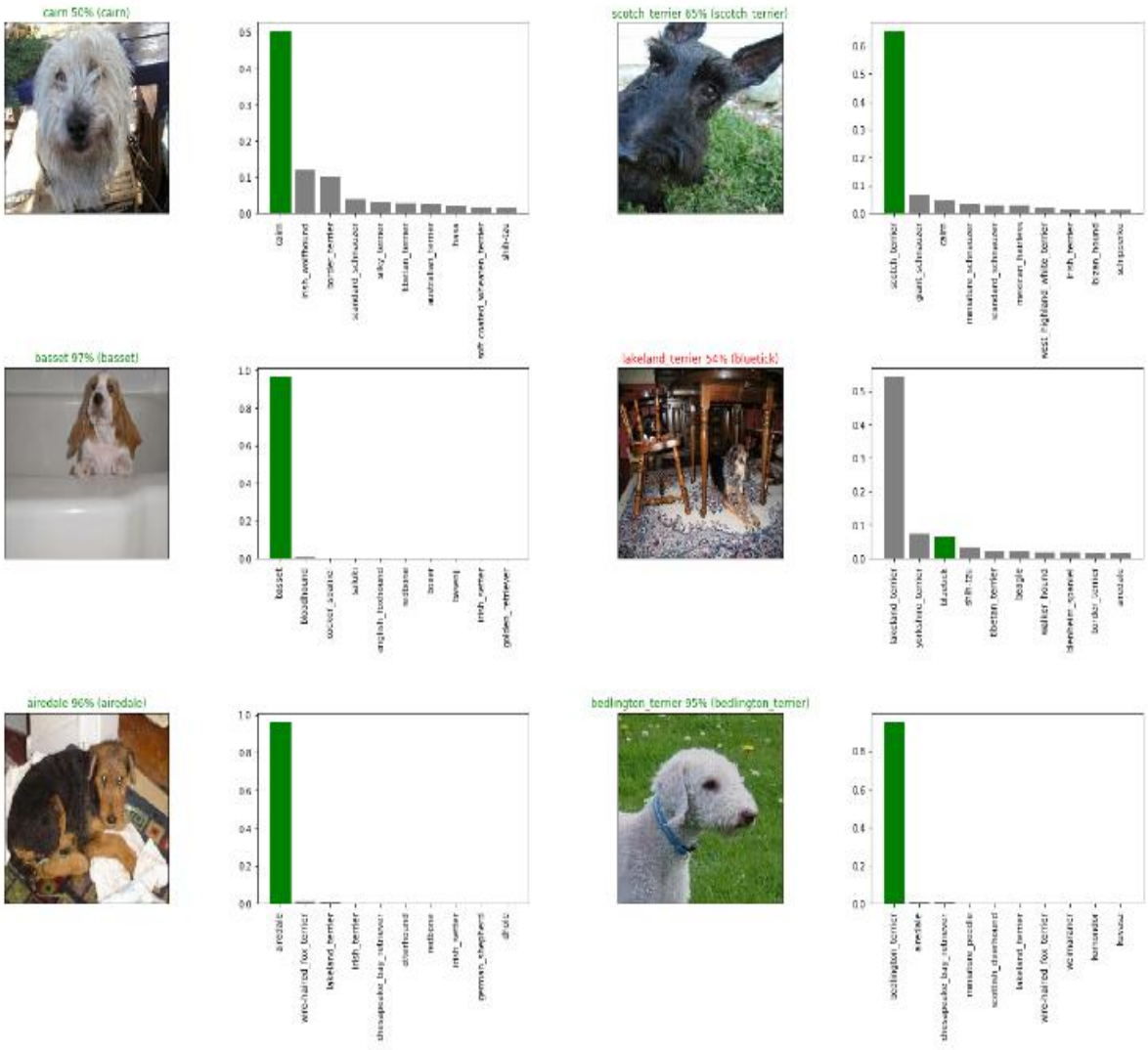

fig 4.4 Predictions on data with values

Since we're working with a multi-class problem (120 different dog breeds), it would also be good to see what other guesses our model is making. More specifically, if our model predicts a certain label with 24% probability, what else did it predict?

fig 4.5 Prediction result on custom data(images)

These are the custom images provided by me and the above written titles are the titles which model has predicted

CHAPTER 5

CONCLUSION

In this project following the instructions and codes given by the capstone project material, I have tried to build a system for dog breed recognition. The API for the web or mobile application is left for future work. in this work, I tried to build convolution networks from scratch and by doing I learned the challenges and realized the reasons for existing more promising methods in particular transfer learning. After I took advantage of different transfer learning options available and I have improved my model significantly.

As discussed above this model can be used in various medical fields for pre classification of breed before conducting any further operations on it apart from that this model can also be used in dog business sector to provide authenticity to buyer as well as seller and one more use of the model could be that it can be used for various dog competitions and contests so, that enrolment process could be made smooth and dogs could be classified at the time of registration.

Future Scope

Although this project tries to cover all prediction methods, these are some of the few points where the scope of the project can be expanded -

1. In the future, we can make API for the web or mobile application.
2. This system provides great accuracy but for experimentation another models can also be used and checked if performance improves.
3. For advanced prediction more data of other animals can be added and the role of model can be expanded so that it could be able to predict between various animals and their breeds.
4. This Dog Breed Prediction can be advanced to be used by any medical organization in the world for conducting accurate research on dogs.

REFERENCE

1. [https://colab.research.google.com: Official Documentation from Google Colab]
2. [https://www.tensorflow.org: Official Documentation from tensorflow]
3. [https://www.tensorflow.org/hub: Official Documentation from tensorflow hub]
4. [https://keras.io/api/applications: Official Documentation of Keras]
5. [TensorFlow tutorial for experts by Randy Moore]
6. [A Thousand Brains by Jeff Hawkins]