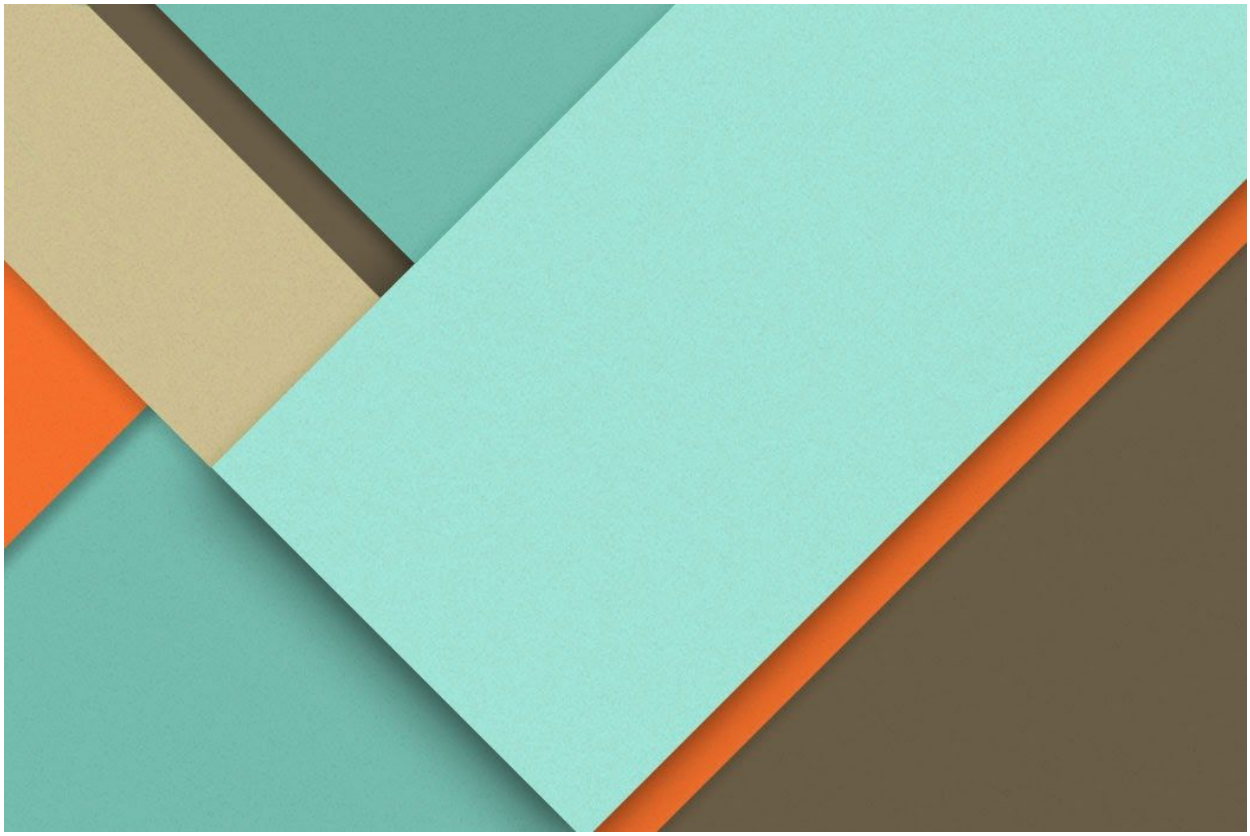


Capstone Project Proposal

Prepared for: Udacity's Machine Learning Engineer Nanodegree Program
Prepared by: Anantharam Vemuri
October 27, 2020
Proposal number: 001

Domain Background

Artificial Intelligence (AI) gives machines the ability to improve with experience. Machine Learning (ML) is a way of displaying AI and is more data-centered. ML is a field in AI that gives computers the ability to take decisions without being explicitly programmed.

A machine learning approach to image classification involves identifying and extracting key features from images and using them as input to a machine learning model. With this method, the computers are taught to recognize the visual elements within an image.

I am interested in learning about the methods and algorithms used to solve problems involving image data. Moreover, images give us immediate feedback which would help us assess the model quickly.

Solving the dog breed classification problem is not easy even for a human. There are similar looking dogs belonging to different breeds and differently looking dogs from the same breed. It would be interesting to see how a Neural network could solve this type of problem.

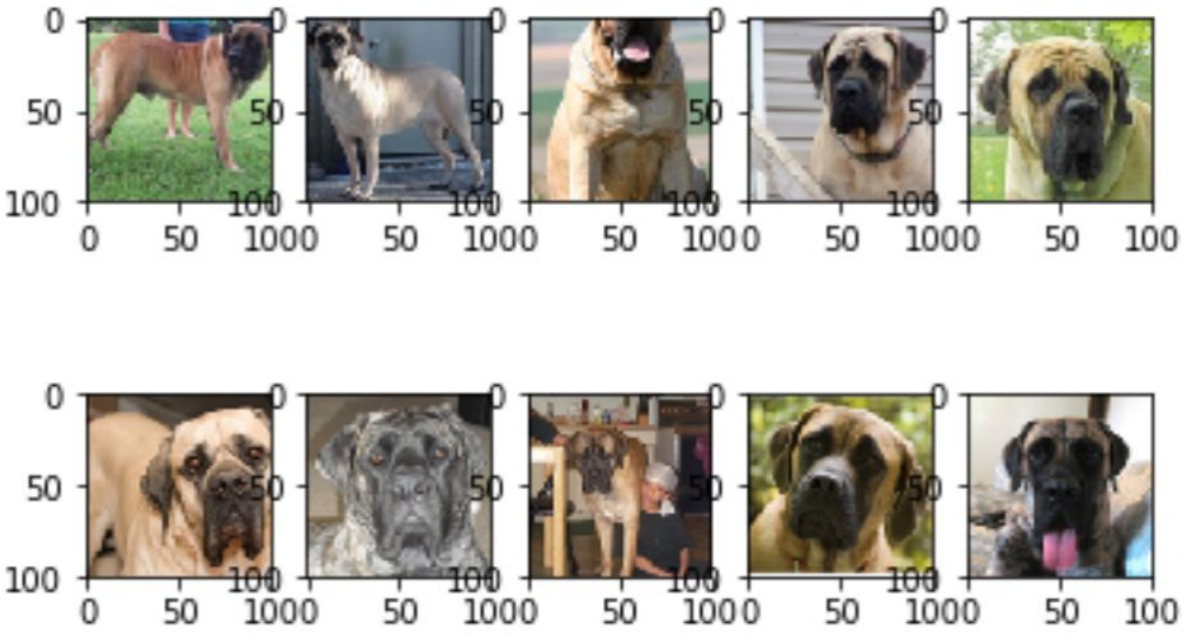
Later, I would like to build an android app which could potentially take an image as an input and output the dog breed. This approach to image classification has many potential uses and can be used to solve many real life problems.

Problem Statement

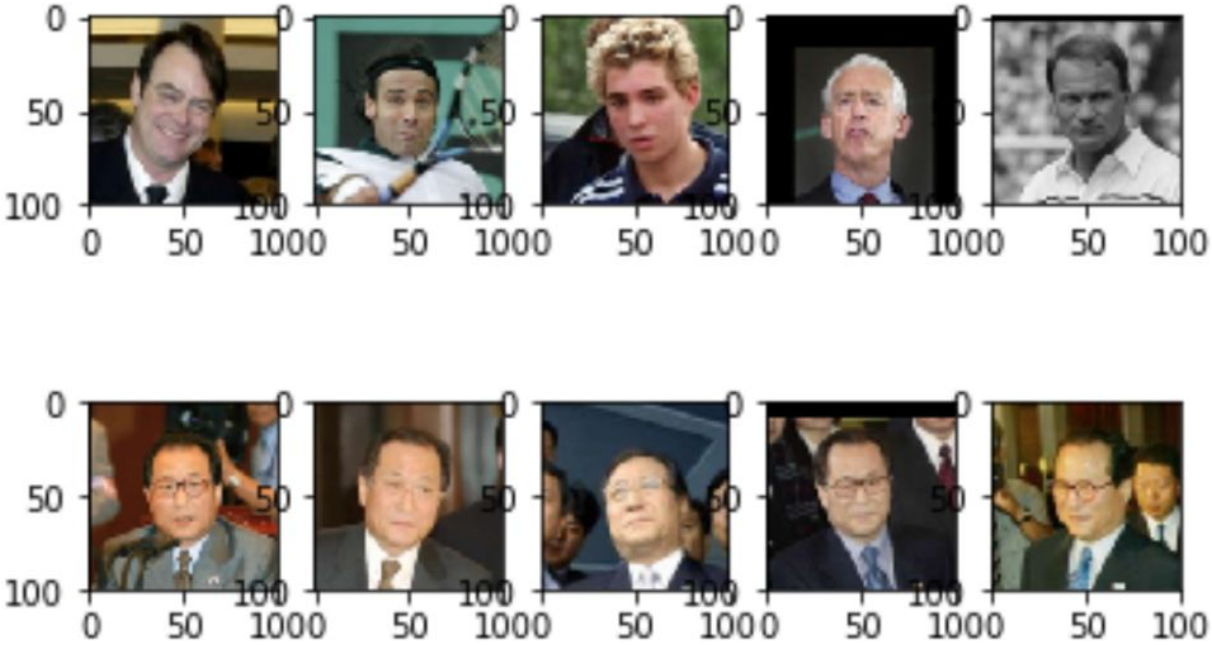
Given an image, check whether the image contains a dog or human. If a dog is detected in the image, an estimation of the dog's breed is output. If a human is detected, an estimate of the closest dog breed is output.

Datasets and Inputs

The **dog dataset** consists of 8351 dog images of different breeds. A sample of 10 dog images are displayed below.

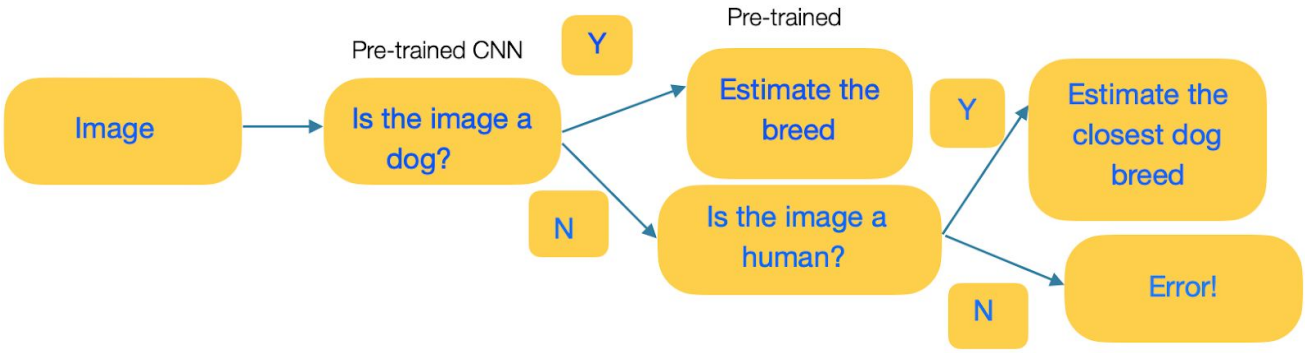


The **human dataset** consists of 13233 human images. A sample of 10 human images are displayed below.



Solution Statement

CNNs are used for solving this problem. Convolutional Neural Networks (CNNs) are a class of Neural networks used for analysing computer imagery. It falls under the Deep Learning umbrella.
The flowchart for the solution is given below:



A pre-trained VGG16 model is used to detect if a dog is in the image. OpenCV's implementation of the 'Haar feature-based cascade classifier' is used to detect human faces in images. A custom CNN is built to estimate the dog breed. Later, to increase the accuracy, a transfer learning based CNN is used to estimate the dog breed.
If the image contains neither a dog nor a human, an error message is output.

Benchmark Model

- The CNN used to Classify Dog Breeds (from Scratch) should attain an accuracy of > 10%. Even if the model picks a dog breed randomly, that would give us a probability of 1/133 which is < 1% accuracy. So if the model gives an accuracy of 10% then it clearly is not random.

- The CNN created using transfer learning should attain an accuracy of > 60%.

Evaluation Metrics

Accuracy alone may not be enough for a multi-class classification problem. Cross-entropy could be more effective. It measures the extent to which the predicted probabilities match the given data.
In addition to this, I am also looking into Cohen's Kappa metric.

Project Design

1. Import the datasets: I will be using the 'dog dataset' and 'human dataset' provided by Udacity.
2. Detect humans: Using OpenCV's pre-trained 'Haar feature-based cascade classifier' humans are detected. This is later called in step 6.
3. Detect Dogs: Dogs are detected using a pre-trained VGG16 model. This is later called in step 6.
4. Create a CNN to Classify Dog Breeds (from Scratch): The data is split into train, test and validate. A ConvNet is built from scratch and is trained to classify the given input image into one of the 133 dog breeds. This ConvNet should attain a minimum accuracy of 10%.
5. Create a CNN to Classify Dog Breeds (using Transfer Learning): The CNN built in the previous step will have too less of an accuracy to be of much actual use. So, transfer learning is performed on a CNN whose architecture is known. Here we replace the last layer in our CNN instance with a dense layer specific to this problem. This ConvNet should attain a minimum accuracy of 60%.
6. Write the Algorithm: The methods from steps 2, 3 and 5 are called according to the flowchart given in the 'Solution Statement' section.
7. Test the Algorithm: The algorithm is finally tested with sample images and evaluation metrics like cross-entropy.

References

1. 'Introduction to Machine Learning with Python', Andreas C. Muller & Sarah Guido
2. https://medium.com/nanonets/how-to-easily-build-a-dog-breed-image-classification-model-2fd214419cde