Capstone Proposal

Dog Breed Identification

Neetu Murmu

Domain Background

Image classification is a task of assigning an image a particular label or a class from a fixed set of labels. This is one of the core problems in Computer Vision, and has many practical applications in fields like agriculture, surveillance, traffic sign detection etc.

For the task of image classification, we use Deep Learning techniques which helps us to train the machine to be able to 'see' and 'classify' the image. These techniques work by extracting multiple features from the image dataset to identify different patterns in them.
In deep learning, Convolution Neural Networks (CNN) are the widely used algorithm for image recognition, which we're going to implement to solve the given problem.

Problem Statement

Our objective for the capstone project is to build a Machine Learning model that can identify the dog breed. We would explore how we can use CNN model to identify the dog breed for an input dog image (or even an input human image for fun!)

We are also given a ready to use Haar Cascade face detector which we can use to detect faces for the human images and dog images we are given. We measure their accuracy in terms of accuracy for each dataset.

Datasets and Inputs

For this problem, we are given a dog images dataset and a human images dataset that we can use to train our dog breed identifier model. We need to transform our input images before we can pass it to the model.
Transformation includes feature processing and feature engineering techniques, the latter of which is not necessary but we do it in order to provide better quality features to our model which is ultimately going to affect our model performance.

Some image transformation techniques we're using are:

- Image Resizing
- Normalization
- Augmentation etc.

Solution Statement

We will use deep learning to build our model, and in deep learning, we're going to use CNN model for our image classification task.

We will implement different types of CNN model for our image classification task:

1. A model built from scratch and
2. A model built on top of a pre-trained model such as VGG16.

Model built from Scratch

First, we write a model from scratch using a combination of :

1. Convolutional Layers
2. Pooling Layers
3. Dense Layers (Fully Connected Layers)

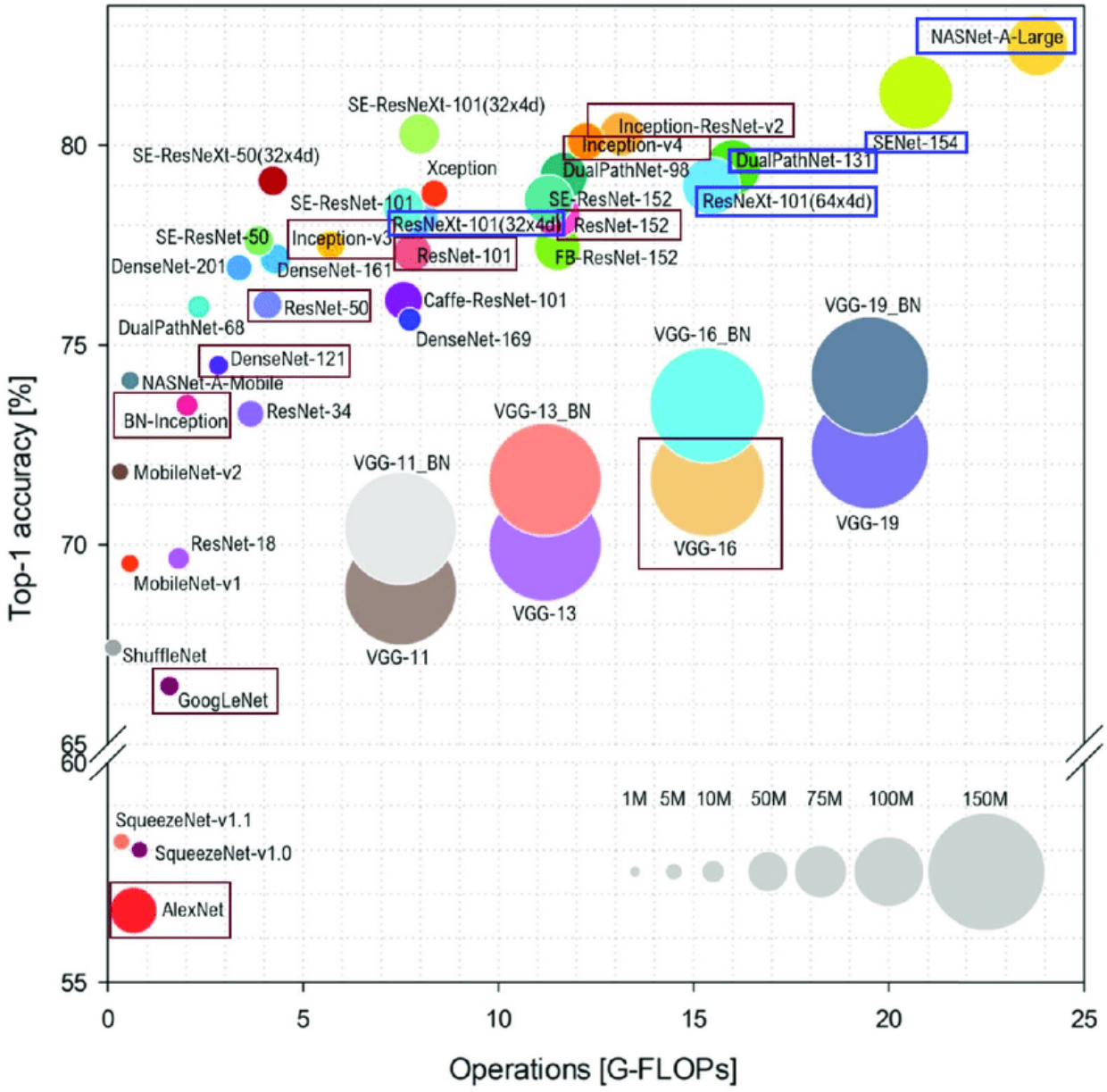
We use convolutional layers to extract feature maps from the input images, and then we use pooling layers to reduce the image size to reduce computational cost.

We stack a few pairs of convolutional layers and a pooling layer to extract the features from our images and then pass the extracted features to fully connected layers, and this is where the classification process starts. After a few FC layers we'll have a final output layer which will give us the final prediction.

Model built using Transfer Learning

Transfer learning is the reuse of a pre-trained model on a new problem. Transfer learning is effective because for most of the image classification tasks, primitive features and goals are similar like the ability of the model to identify straight lines, circles and other common building block features.
To build a model using transfer learning, we take a pre-trained model which is trained on the similar dataset as our problem dataset and stack a few dense layers on top of it so we can fine tune the model according to our dataset.

For the given problem statement, I've chosen ResNet50 for transfer learning, it has decent performance compared to other algorithms and relatively lower number of FLOPs as shown in the picture below.



Source: [Researchgate.net]

Benchmark Model

We can choose the VGG16 model trained on ImageNet as a benchmark model because of the similarity with our given dataset. It achieves 92.7% top-5 accuracy after being trained on 14 million images belonging to 1000 classes.
Of course, it'll be very difficult to get that kind of accuracy with the size of the dataset we're presented with. But, at least we can get the sense of how good or bad our model is compared to this model.

Evaluation Metrics

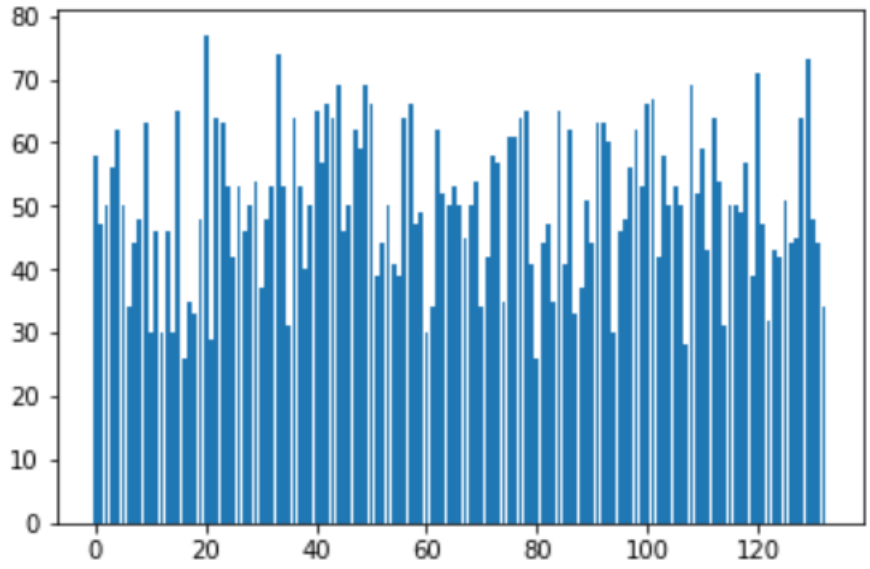
For evaluating the model, we'll take plain accuracy score i.e.

Accuracy = number of correct predictions / number of total instances

After analyzing the data, we find that there are not huge differences in the number of training examples for each dog breed category, so picking plain accuracy as an evaluation metric is a safe choice.

Class with minimum training example had: 26 training examples
Class with maximum training example had: 77 training examples

The bar chart below shows the number of training examples in each category(133).



Justification

Model built from Scratch

This CNN model developed from scratch had an accuracy of 11% (96/836)

Few points on training:

- Model seemed to overfit very quickly with more than 3 Convolutional layers.
- Not all the augmentation technique seemed to be contributing to performance, some techniques that did help: Horizontal Flip and Random Resized Crop

The model developed from scratch has very low accuracy compared to benchmark model defined above, but it's expected since we have very less amount of data to work with, but further fine tuning the model parameters and hyperparameters might improve the model to some extent.

Model built using Transfer Learning

The CNN model developed using transfer learning had an accuracy of 81% (679/836)

- This model also had an improved performance with the augmentation techniques chosen above.

The model built with transfer learning has reasonable accuracy compared to the benchmark model considering that it's a very basic transfer learning model. We can further try to improve the accuracy by tuning hyperparameters(epochs, optimizer etc.) and by making the network deeper.

Project Design Outline

Steps to develop the dog breed classifier:

- **Data Collection**

  Data is already available in the workspace for us to develop the model.

- **Data Exploration**

We explore the data to find out if there is an imbalance in data which will help us in making the right decision about modelling the solution e.g. what evaluation metric to use, how to split the data etc.

- **Prepare Data Loader**

  We prepare a data loader using our dataset to train and evaluate our model. Here we can configure things like batch size, whether we want to shuffle our data etc.

- **Data Processing**

  Before we pass the data to the model, we need to perform certain processing steps like data normalization, resizing, converting input to tensor etc.

- **Data Augmentation**

  To diversify our dataset we do some data augmentation for, e.g. for a dog image, we can do blurring, image rotation, color adjustment etc. to create new images because while testing our model, it's possible we can receive a similar image with maybe picture taken at a different angle or so on.

- **Model Training**

  We train 2 types of model:
  - CNN model built from scratch
  - CNN model built using transfer learning

- **Writing Algorithm**

  We write simple algorithm with logic as following:
  If a dog is detected, predict its breed
  else if human face is detected predict which dog breed would it look like if they were dog
  Else return error (if no dog or human face was detected)

  Models used for dog detection: VGG16
  Model used for human face detection: Haar Cascade
  Model used to predict dog breed: CNN model we built with transfer learning

References

https://www.researchgate.net/figure/Ball-chart-reporting-the-Top-1-accuracy-vs-computational-complexity-ie-floating-point_fig2_345237258

https://www.geeksforgeeks.org/vgg-16-cnn-model/

https://docs.opencv.org/master/db/d28/tutorial_cascade_classifier.html