Machine Learning Engineer Nanodegree

Capstone Project Report

Neetu Murmu
Jun 20th, 2021

Definition

Project Overview

Computer vision is a field of computer science that works on enabling computers to see, identify and process images in the same way that human vision does, and then provide appropriate output.
In this project we will perform image classification, which is one of the core problems in Computer Vision. It's a task of assigning an image to a particular label or a class from a fixed set of labels. It has many practical applications in fields like agriculture, surveillance, traffic sign detection etc.

In deep learning, Convolution Neural Networks (CNN) are the state-of-the-art algorithm for image recognition, which we're going to implement in this project.

Problem Statement

Our objective for the capstone project is to build a Deep Learning model that can identify the dog breed given an input image. 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.

So, our overall goal will be to build a system which takes an input image and:

- if a **dog** is detected in the image, returns the predicted breed.
- if a **human** is detected in the image, returns the resembling dog breed.
- if **neither** is detected in the image, it provides output that indicates an error.

Metrics

For the model we develop, we'll use accuracy as an evaluation metric.

Accuracy = Total correct predictions/Total input samples

In the data exploration section below, we will see that our dataset is not much imbalanced so plain accuracy is a good choice for the evaluation metric.

Analysis

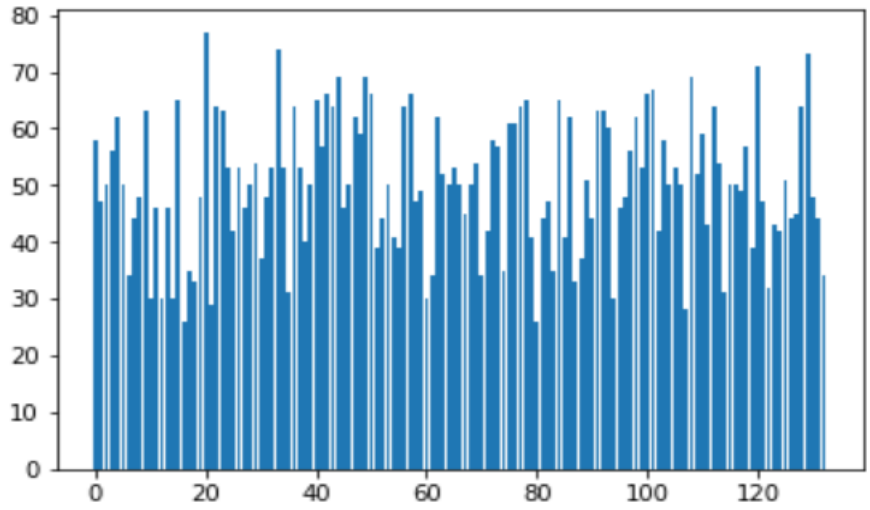
Data Exploration

We're already given a dataset for this project in the workspace that we can use for training the model and evaluation. Data are already split in train, validation and test set, with each type of data available in a different directory.

Since our aim in this project is to perform a classification task, we should check if our dataset is balanced i.e. whether each category in our dataset has reasonable numbers of training examples or not.
Further, we need to make decisions depending on whether the dataset is balanced or not e.g. data sampling, feature processing and algorithm to use for training the data.

Exploratory Visualisation

Below, I present the visualization of the number of training examples for each dog breed category, as mentioned above, to check if the data is balanced.



X-axis: Dog breed type (0-133)
Y-axis: Number of training examples for a dog breed (Min: 26, Max: 77)

As we can see, although the data is not very uniform, there isn't a huge imbalance. We can use this information to decide which evaluation metric to use, and whether we need to sample more data.
And, since our data is somewhat balanced, we can choose accuracy as the evaluation metric and also it doesn't seem like we need to collect more data for a dog breed type because each dog category seems to have a reasonable amount of data to train the model on.

Algorithms and Techniques

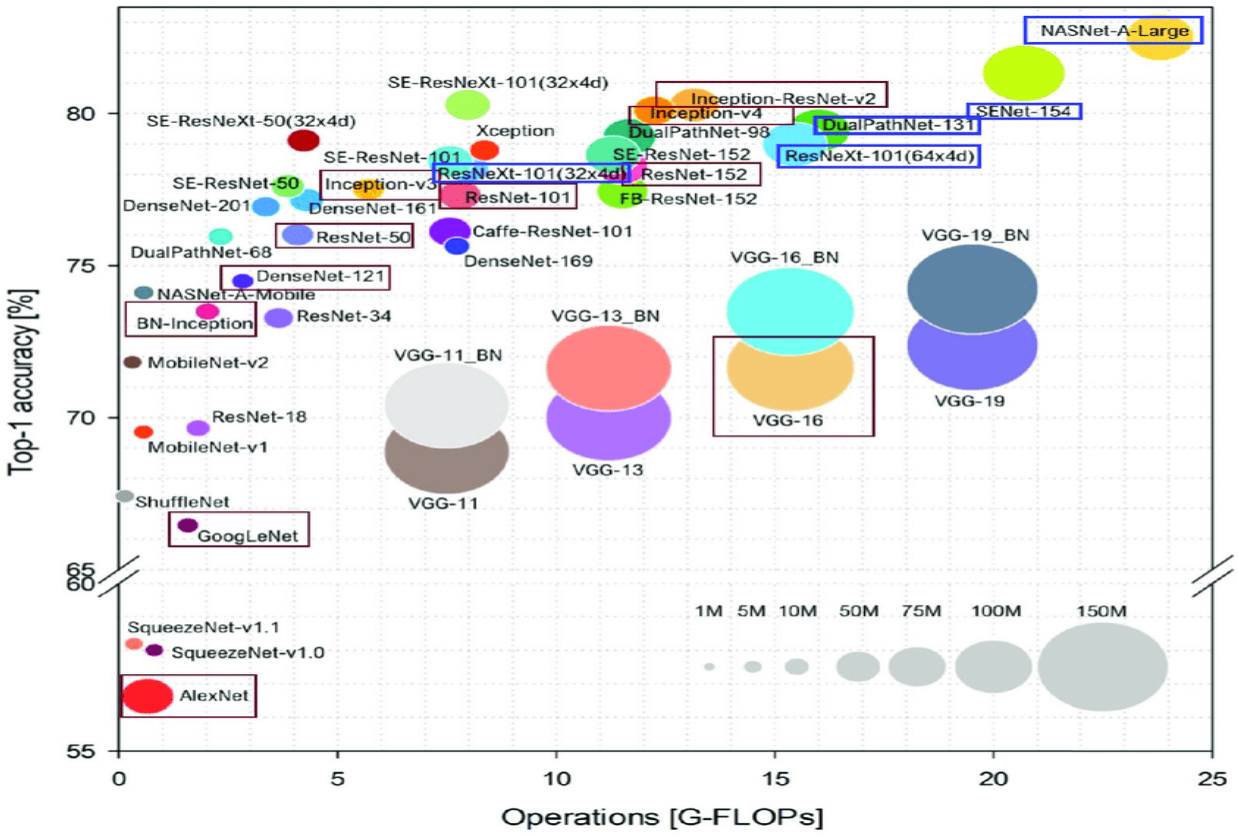
For the problem we are given, we'll choose a deep learning technique called CNN (Convolutional Neural Network) because they tend to have better performance compared to the other algorithms for image recognition.

CNNs work by extracting multiple features from the image dataset to identify different patterns in them, and use those identified patterns in the images to classify the input images.

We will use 2 types of CNNs in this project:

1. CNN built from scratch
2. CNN built using transfer learning

We will use ResNet50 for transfer learning since it has decent accuracy and a reasonable number of FLOPs according to the image below.



Source: Researchgate.net

Benchmark

We can choose the VGG16 model trained on ImageNet as a benchmark model because of its similarity with the given dataset. It achieves 92.7% top-5 accuracy after being trained on 14 million images belonging to 1000 classes which also contains dog breed categories.

Methodology

Data Processing

We need to do some data processing before we can pass it to our model. Data processing steps include:

1. Image resizing - because our model expects uniform images
2. Data Conversion to Tensor - Since we're using pytorch to build the model, we need to pass data as tensors.
3. Data Normalization - so we can have all the input values in the same range of 0-1.

In addition to the data processing methods mentioned above, we also do some data augmentation in order to diversify our dataset. Data augmentation includes:

1. Random cropping of the image
2. Random horizontal flipping

Note, that these are the data augmentation techniques that helped improve the performance. There are also some other techniques that I tried but didn't make much of the difference in model performance e.g. Gaussian blur, Random Rotation etc.

Implementation

Data Processing : We make use of PyTorch transforms to process and augment the data.

Data Loading or Batching: We use PyTorch dataloader to batch the dataset before sending it to the model.

*Algorithm Implementatio*n: CNN can also be implemented using PyTorch nn library.

Refinement

To build the CNN from scratch, I started with a few basic convolutional(3), pooling(3) and dense layers(2).

To refine the model further:

- I tried to make the network deeper → Tends to overfit the data
- Tried Adam and SGD optimizer → SGD seems to work better.
- Did more data augmentation, some improved the model (horizontal flip, random crop) some didn't improve much(gaussian blur, rotation)
- Picked the right epoch number by trying out different values.

Results

Model Evaluation and Validation

To evaluate the model properly, we split the data into 3 sets, train, validation and test set.

Train and Validation are used to pick the correctly tuned model and test set, which is untouched, is used to evaluate the model performance.

Justification

The CNN 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.

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.

Conclusion

Improvement

We can further improve the model by:

- Sampling more data using data augmentation since we have a very small dataset.
- Making the network deeper.
- Tuning hyperparameters.

References

https://pytorch.org/vision/stable/transforms.html

https://pytorch.org/tutorials/beginner/basics/data_tutorial.html

https://pytorch.org/tutorials/beginner/pytorch_with_examples.html

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