Project

Convolutional Neural network for Dog breed Classification

Osazee Ero

27th November 2020

I. Definition:

Project Overview:

This project involves the application of convolutional neural network for predicting different kinds of dog breeds. This task is very relevant in this era as several similar types of Dog breeds exist, and it could be sometimes difficult for humans to distinguish them. The project involves exploration of several CNN architecture and techniques for training them.

Problem statement:

The primary task is to distinguish between several types of dog breed using Convolutional neural network (CNN. The end goal of the project would be an application that:

- Predict the type of dog breed in an image if present
- Guess the resemblance of a human to a type of dog breed

Metrics:

The performance of the trained model would be the accuracy metric since we have a balanced dataset.

II Analysis

Data Exploration:

Two dataset was employed for the implementation of this project:

Dog dataset: This dataset consists of 133 classes of different dog breeds with a total of 8351 images. Some samples of this dataset are shown below. The images are of different sizes and needs to be further preprocess to make them suitable for CNNs. The datasets are split into train, validation, and test datasets with about 6680 images for training, 835 for validation and 836 images for training.

Another dataset also used is the Human dataset (These datasets were provided in the Udacity workspace) used for testing for human faces, this dataset is all the same size.

Exploratory visualization:

Algorithm and techniques:

The major algorithm employed to achieve the task at hand is the convolutional neural network which is a type of deep learning method suitable for computer vision tasks that uses learned matrix weights for extracting useful features from images. These features can then be used with any machine learning classifier such as artificial neural network for several computer vision tasks such as classification of different dog breeds.

To ensure this approach is suitable for the given task, we would start by first experimenting with a pretrained convolutional neural network specifically the VGG16 architecture, which is a popular type of deep CNN that was trained on the ImageNet dataset which consist of over 14 million images belonging to over 1000 classes including several dog breeds.

The next step would be to train a custom CNN architecture for the task to see if we could get better results. Lastly, we would apply transfer learning to the VGG-19 architecture which is a variant from the VGG CNN architecture type. Also, an off-the-shelf human detector is used for human face detection.

Benchmark:

The final trained model would be compared with a pretrained CNN architecture (VGG-16) which was trained with over 14 million images belonging to over 1000 classes including several dog breeds, and a custom model trained from scratch.

III Methodology

Data preprocessing

Similar preprocessing was done for the three models experimented with (Benchmark model -VGG-16, custom CNN, and Transfer-learning VGG-19n models). The several steps include first loading the data into memory using Pytorch Datasets Image Folder helper function. Then several transforms were applied to the image such as resizing the image to the required CNN input size, several data augmentation operation such as horizontal flipping, and rotation. Also, the datasets were normalized using pretrained normalization values from ImageNet dataset.

Implementation:

a. **Benchmark model:** The VGG16 was imported from pytorch model's library. Then custom code was written to predict a given dog breed class from a given image path.
b. **Custom trained model:** A custom model gotten from inspiration of the AlexNet architecture was constructed from scratch. The network is shown below. It consists of

series of convolutional layers with batch normalization to prevent dying gradients in the network. I also introduce several dropouts to avoid overfitting in the network.

```
Net(
  (conv1): Conv2d(3, 32, kernel_size=(11, 11), stride=(4, 4))
  (bn1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (conv2): Conv2d(32, 64, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
  (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (conv3): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (conv4): Conv2d(128, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (conv5): Conv2d(64, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (pool): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
  (fc1): Linear(in_features=1152, out_features=512, bias=True)
  (fc2): Linear(in_features=512, out_features=300, bias=True)
  (fc3): Linear(in_features=300, out_features=133, bias=True)
  (dropout): Dropout(p=0.5)
)
```

c. **Transfer-learning model (VGG-19_bn):** This involved loading a pretrained network then freezing the weights to prevent retraining them. I then replaced the classification layer with a custom classification head. After which I trained the classification head for the task at hand.

I employed the Adam optimizer and the cross-entropy loss function in all the trained models.

Refinement:

For the custom trained model, I initially experimented with several CNN architecture and hyper-parameter values. I also tried the Stochastic gradient descent optimizer which performance was less than the Adam optimizer. The transfer learning model was straight-forward as from the initially experiment using the pertained model, the VGG network architecture was shown suitable for the task at hand.

IV Results

Model Evaluation and Validation

For comparison purposes, the benchmark pretrained dog-detector model (VGG-16) was employed in detecting presence of dogs and faces in images: A dog detection accuracy of 100% and human detection accuracy of 2% was reported. This is expected as the classifier was tailored for detecting the presence of dog in an image. The 2% reported accuracy for human images was false positives, where the classifier thought it was a dog, but it was instead a human. The custom trained model had an accuracy of 20% for the dog breed classification task, although the model was trained for only 50 epochs, and from initial observation the model performance was increasing steadily with each epoch, justifying that more epochs would result in better model performance.

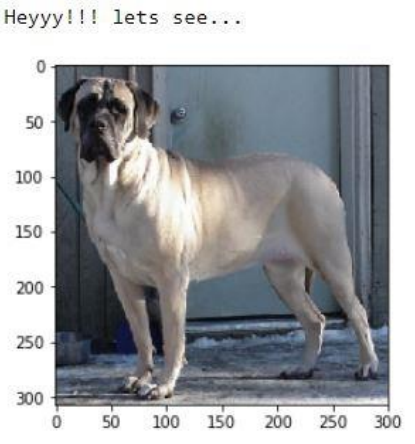
The best model was the model from transfer learning which an accuracy of about 79% is reported for only 5 training epochs. Observation from training logs also showed that the model performance can be improved with further training.

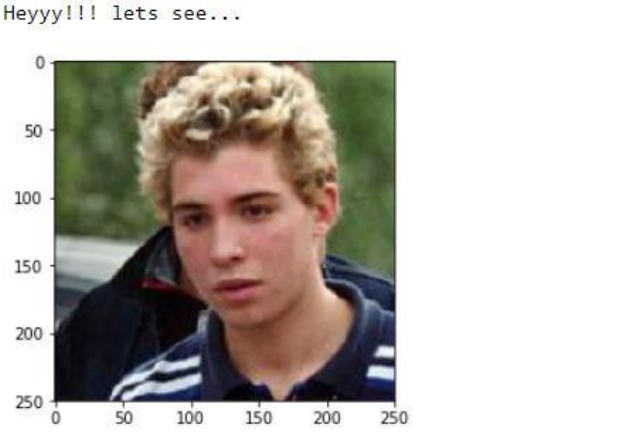
Justification

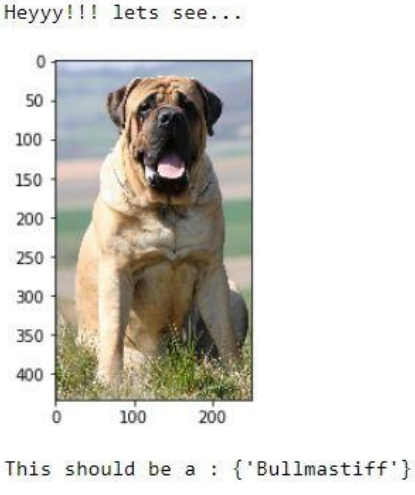
From observation of the results gotten, CNN is a very effective approach for dog breed classification.

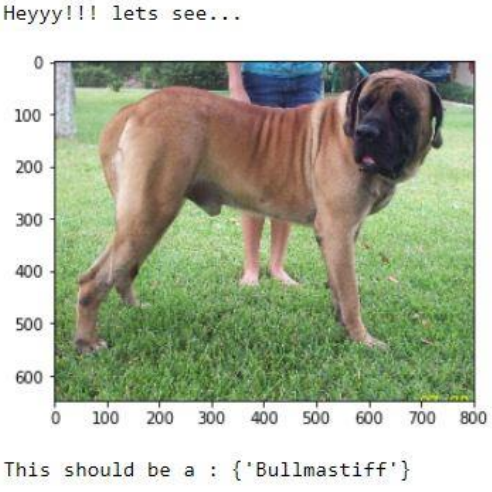
V Conclusion

Free-form Visualization









Reflection:

To summarize the entire project pipeline:

First load the given dataset and implement several preprocessing operations on the given datasets

Design on a suitable architecture for the classification tasks at hand. This is a very important stage as a poorly selected CNN network would result in poor model performance.

Lastly, experiment on several hyper-parameters for fine-tuning the model to attain the desired performance.

The most challenging part of this project was coming up with a good custom network. I add help from several blog posts and github repository.

Improvements

The algorithm can be improved by doing the following: Training the model for more epochs, sourcing for more diverse dataset, and experimenting on several data augmentation types and hyperparameters selection.

References:

https://pytorch.org/tutorials/

https://medium.com/

https://github.com/

https://classroom.udacity.com/me