Machine Learning Engineer Nanodegree

Final Report

Ishaan Upadhyay, June 10th, 2020

Definition

Domain Background

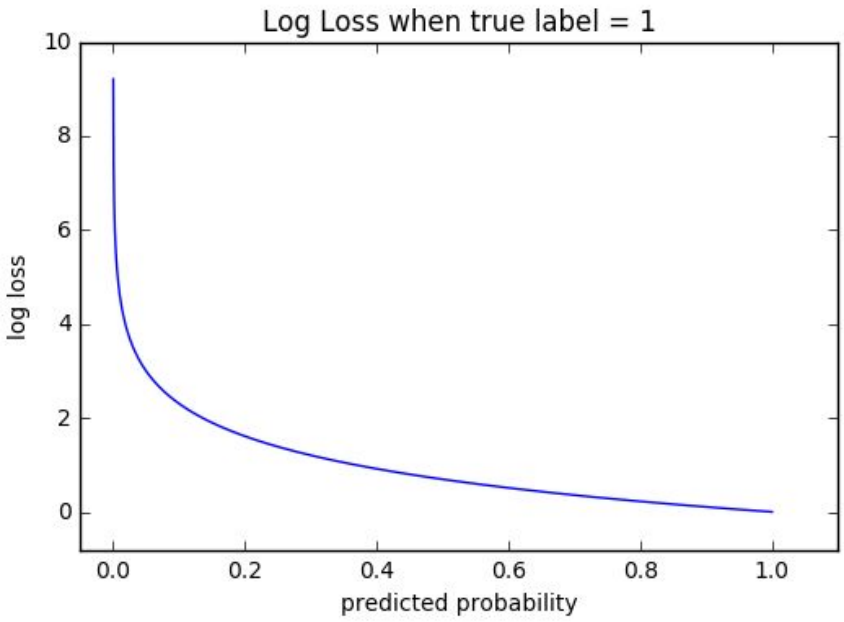
Dog breed classification is an open issue in the image recognition field, contained as a subset of several datasets such as ImageNet on which some of the best convolutional models have been trained. Dog breed classification is problematic because of some breeds having minimal variation between each other, as well as some breeds having entirely different colours. In this project, the task given is to construct a classifier that can classify dogs into different classes based upon estimated breed, as well as give an estimate of which breed a human being resembles. This could eventually be deployed in a web-app or pipeline.

Problem Statement

Given an image, first, identify whether or not it contains a dog or a human using a detector. If a dog is detected, identify the breed it belongs to. If a human face is detected, identify which dog breed it most closely resembles. The product must be suitable for usage in a web app.

Evaluation metrics:

The loss will be evaluated using cross-entropy loss, a combination of negative log likelihood loss and logarithmic softmax. The cross-entropy loss function is common in classification problems with any number of classes, and works best here. To explain cross-entropy loss, I've used this graph (citation below)



Cross-entropy loss works with multiple class classification problems well because it strongly penalizes incorrect predictions which are also overconfident. Thus, a model will learn not to ascribe too strong of a prediction to any one class, which benefits the output. When the softmax function is used on the 133 outputs, only the maximum value remains as 1, which is indicative of maximum confidence. Thus, to ensure that our model is not overconfident in deployment, the cross-entropy loss function is suitable.

Accuracy (the number of correctly classified images divided by the total number of images to classify) will only be used to evaluate if the model has been trained properly after training is finished.

Analysis

Data exploration:

Sample images from both datasets

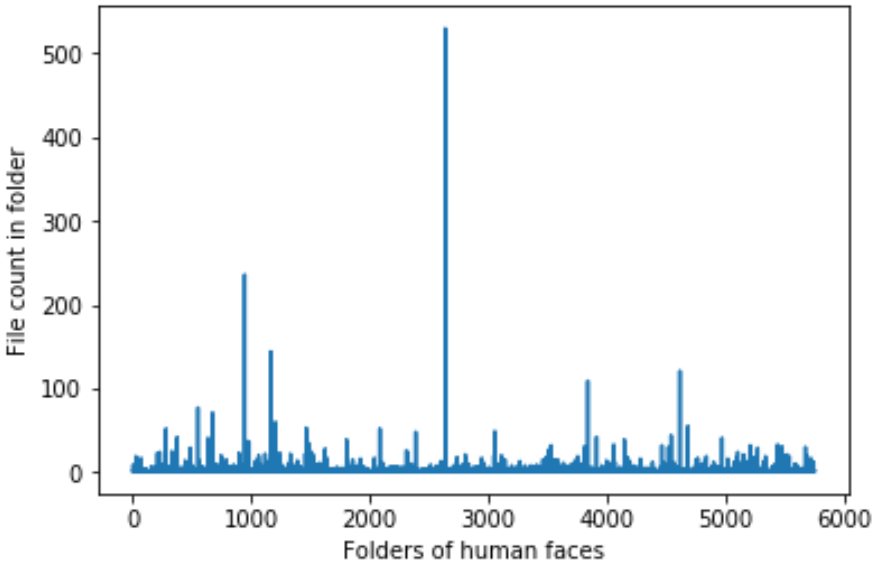
Both the datasets (dig images and human faces) used for this capstone project consist entirely of .jpg images, though they are sorted into folders in slightly different ways. They are all fully RGB (therefore, 3 input channels for any model).

Human dataset: In the human dataset, there is no segregation into training, testing or validation sets. There is a total of 13,233 images across all 5,749 folders. Each folder is labelled with a person's name and contains images of that person. All pictures are 250 by 250 pixels, although the location of the face, angle, lighting and overall position may vary.

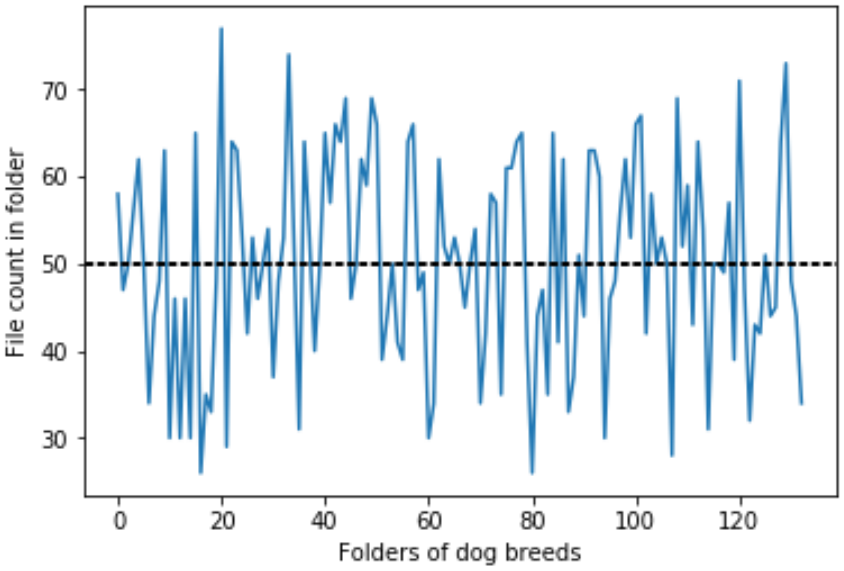
Dog dataset: The dog dataset is split into training (6,880 files), testing (836 files) and validation sets (835 files) with a total of 8351 images. Each folder is split into 133 subfolders, each for one breed. The image sizes, however, are not consistent. For example, a few file sizes include 640 by 359 pixels, 300 by 237 pixels and 418 by 500 pixels. This may pose a few challenges during image augmentation.

Exploratory Vizualization

One feature that was explored was the number of files located in each folder, to determine whether or not the datasets were balanced. To do this, matplotlib's pyplot functionality was used alongside os.walk(). For the dog dataset, only the training set has been graphed.



From the above graph. It is apparent that the human dataset is heavily imbalanced, with an average and median file count of 1 per folder. However, there are several folders that include more than 50 files, 4 with more than 100 files and 1 with more than 500 files (the folder containing pictures of George W. Bush). However, since the human dataset is not being used for training in any capacity, this is not of great importance. The end functionality of the code is simply to relate a human face to a dog breed, with no training steps. As such, the imbalance of this dataset is not of consequence.



The above graph shows the file count per folder for the dog training dataset. In contrast to the human dataset, the mean of the file count is actually close to the middle of the graph (located at 50.22, median located at 50), but it is still very imbalanced. A few folders have fewer than 30 files, while the majority have more than 40. This may result in some difficulty in classifying certain dog breeds. As detailed in the notebook, some breeds have minimal variation from one another, and one breed alone may have drastically different colours.

Algorithms and Techniques:

Firstly, to define whether or not an input image contains a dog or human, OpenCV's implementation of face detection based on Haar feature-based cascade classifiers will be used alongside a pre-trained VGG16 model. Both models have been extensively tested, and 118 of the possible VGG16 predictions correspond to dog breeds from the ImageNet dataset.

Two convolutional neural networks (CNNs) will be used for the actual image classification problem. CNNs are best suited for this type of task because unlike other machine learning architectures, they retain dimensional data - that is, the location of each pixel. This is done by using convolutions, essentially looking at the image as a combination of several smaller images of a defined size (the kernel size). Pooling layers allow for the shrinking of the image into smaller and smaller vectors that retain the same spatial coherence. One will be trained from scratch. Dropout and batch normalization layers have also been implemented to prevent overfitting and make training more

efficient. The other CNN will be defined using transfer learning, with a pre-trained ResNet50 model as the base.

The loss function used was cross-entropy loss, and the suitability of this function was discussed in the Metrics section of this report. The optimization function used was stochastic gradient descent, which has been stated to generalize better than adaptive optimization functions such as Adam.

For any model, excluding the Haar cascades, for predictions, images will be resized to 224 by 224 pixels and normalized using a predefined PyTorch transform which is essential to these image classification problems, as it is universally used across all of the predefined PyTorch models.

Benchmark

These have been outlined in the notebook.

- For the CNN model created from scratch, it must exceed 10% accuracy. Even a model simply making a random guess will guess correctly 1 in 133 times, less than 1% accuracy. If it exceeds 10% accuracy, we will know that it is working correctly.

- For the transfer learning model, it must exceed 60% accuracy.

Methodology

Data Preprocessing

In all cases, the images have been resized to 224 by 224 pixels before being fed into the model, accompanied by a predefined PyTorch transform to normalize the 3 colour channels. Also, as a final step, all images are converted into Tensors. This was done to follow successful pre-trained models.

For the training dataset, images were augmented by using random rotations of up to10 degrees in either direction, random cropping and random horizontal flipping. This was

done to prevent any potential overfitting.

Implementation

To provide a solution to this problem, the following steps have been taken:

When an image is provided, it will be passed through the OpenCV Haar cascades for face detection followed by the predefined VGG16 model for dog detection to determine whether or not it contains a dog or a human. If a dog is detected, its predicted breed is returned by the transfer learning model. If a human is detected, the matching breed is returned by the transfer learning model.

Secondly, a model will be trained from scratch on the dog dataset, using a PyTorch custom CNN. The additional algorithms to be used will be max pooling and batch normalization, layers that are seen in prominent models like VGG16. The loss function, as outlined above, was cross-entropy loss, which is especially suited to multi-class classification problems. The optimization was based upon simple stochastic gradient descent, which generalizes better than other adaptive optimizers.

Thirdly, a model was created using transfer learning based upon a pre-trained ResNet50 model. This is ideal because the ImageNet dataset upon which it was trained already contains a sub-problem of identifying data The optimizer and loss function are the same as outlined above.

Refinement

In previous iterations of the CNN which was created from scratch, batch normalization was only implemented towards the end of the forwarding process, alongside an epoch count of only 15. Due to this, its test accuracy was only 13%. Tweaking the model to add batch normalization layers in between every convolutional layer, increasing epoch count to 50 and increasing the learning rate to 0.003 increased test accuracy to 22%.

Results

Model Evaluation and Validation

The model created from scratch did not perform very well even with 50 epochs of

training, as it produced a test loss of 3.287 and accuracy of only 22%, however, there was a substantial improvement over previous iterations which only achieved 13%. It met the criteria outlined in the Benchmark section by exceeding the 10% minimum. However, this model was still not accurate enough for any proper usage to solve the issue at hand.

The transfer learning model, on the other hand, achieved a test loss of 0.488868 and accuracy of 85% in half as many epochs (25), indicating that using transfer learning with a pre-trained CNN is much more efficient than creating a model from scratch, As there are more than 23 million trainable parameters for ResNet50, and upwards of 103 million for the scratch model architecture, given the accuracy results they achieved, it can be presumed that the parameters are appropriate.

The model was tested with a grayscale image as well as an RGB image of my own face, and the difference in the colourspace made no difference to the model, as it still returned appropriate matching dog breeds. It also correctly assigned the 2 test images of dogs provided.

For pictures of dogs, results from this model can generally be trusted. Dog breed classification is problematic due to low inter-class variation in several cases. Towards the end of the iPython notebook, the model has tried to classify 3 dogs as either bullmastiffs or mastiffs, a problem even a human would have difficulty solving. For humans, there is no way of evaluating the model's predictions.

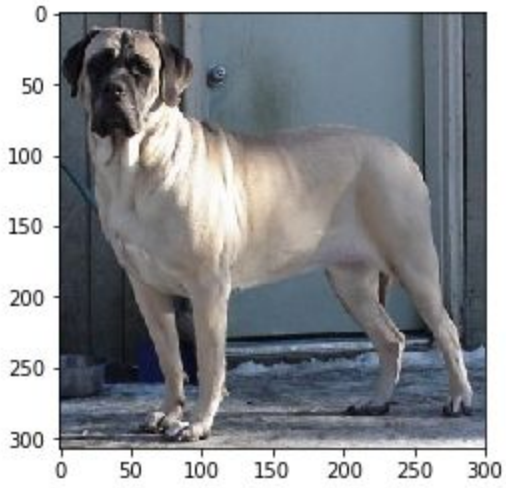
Justification

The final results found are significantly better than the metrics outlined in the Evaluation section. The model created and trained from scratch more than doubled the accuracy requirement with 50 epochs of training. The transfer learning model exceeded the 60% accuracy benchmark by 25%, producing results that were 85% accurate.

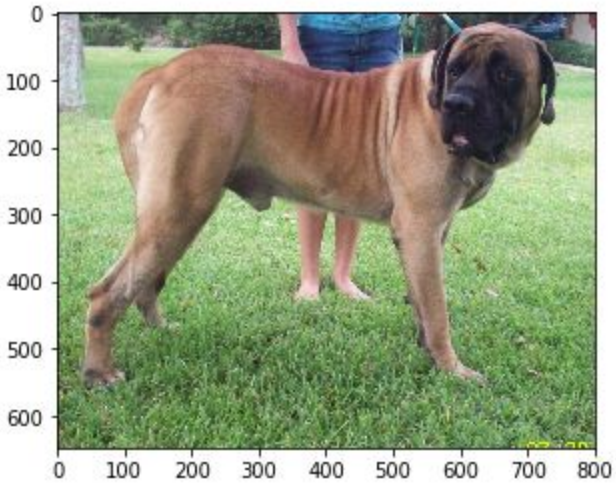
Due to the inherent difficulty of dog breed classification, this final solution can be deemed satisfactory for the amount of time that it took. The weakness of the solution is breeds that closely resemble each other to the point that a human would have difficulty distinguishing them, as well as dogs with extremely different coats who belong to the same breed.

Conclusion

```
What's up, dog?
Your predicted breed is Mastiff! Hopefully, it's the right one...
```




```
What's up, dog?
Your predicted breed is Bullmastiff! Hopefully, it's the right one...
```



Examples demonstrating minimal inter-class variance

Reflection

This project was based around a very interesting central domain and problem statement, and also incorporated several relevant parts of the machine learning

process, including building models from scratch, using predefined models with transfer learning, and a form of mock deployment.

Perhaps the most difficult part was reading over all the different image augmentation possibilities and deciding which ones to implement, In the end, only 3 were selected (random resize crop, random rotation, random horizontal flipping) as well as the predefined PyTorch transform.

The most interesting part was researching the different predefined models, as well as the loss functions which could be used for training. These are all award-winning models, whose classification power can be harnessed right at home, without a supercomputer. There were also many more loss functions than originally expected, so a simpler one was selected.

Improvement

Future improvements to the model could include the following:

- Incorporation of different loss functions/optimizers in an ensemble
- Hyperparameter tuning
- Creating a secondary model to deal with conflicting cases, where 2 breeds closely resemble each other
- More epochs for accuracy increases
- Working on a larger input dataset
- Different image augmentation algorithms
- Weighted combinations of several CNNs, or different transfer learning models.

References

Bonner, A. (2019, June 01). The Complete Beginner's Guide to Deep Learning: Convolutional Neural Networks. Retrieved from https://towardsdatascience.com/wtf-is-image-classification-8e78a8235acb

Dog Breed Identification. (n.d.). Retrieved from https://www.kaggle.com/c/dog-breed-identification/overview/description

Hosays, K., Hosays, K., Ahernesays, I., McCloudsays, T., & ResNet. (2017, December 29). SGD Adam?? Which One Is The Best Optimizer: Dogs-VS-Cats Toy Experiment. Retrieved from https://shaoanlu.wordpress.com/2017/05/29/sgd-all-which-one-is-the-best-optimizer-dogs-vs-cats-toy-experiment/

Loss Functions¶. (n.d.). Retrieved from https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html#cross-entropy

Sebastian Ruder. (2020, March 20). An overview of gradient descent optimization algorithms. Retrieved from https://ruder.io/optimizing-gradient-descent/

Torchvision¶. (n.d.). Retrieved from https://pytorch.org/docs/stable/torchvision/index.html