Dog Breed Classifier Report

I. Definition

Domain Background:

Image classification is a widely used and extremely useful technique that forms the basis of the field of computer vision. Other computer vision techniques such as localization, image segmentation and object detection rely on image classification.

In practice, image classification is used in autopilot in the aviation and automotive industries, in manufacturing settings, to detect proper PPE is being worn, and in medicine to diagnose potential illness in medical imagery with greater accuracy than the human eye.

Detecting dog breeds is a popular image classification problem. There is a Kaggle competition with almost 9,000 entries, where the goal is to minimize multiclass loss1.

Problem Statement:

The classification of dog breeds can be problematic. Images of dogs can be indistinguishable by breed to the human eye, a dog breed classifier mitigates this issue.

The objective of this project is to develop an app that can take an image as an input and determine whether or not the supplied image is a dog. If so, the app should be able to determine what pure breed of dog is shown in the image.

Datasets and Input:

Two large datasets are to be used in this project. A human dataset and a dog dataset. These datasets are provided by Udacity. The dog dataset is a subset of ImageNet2 used as the dataset for Kaggle competition.

Dog Dataset: There are 8351 images, split into train, test and validation with 6,680, 836 and 835 images in each respectively. Each subset contains 133 different folders corresponding to the breed of dog. The breakdown of images per dog breed is not uniform.

Human Dataset: There are 13233 images, the dataset is split into 5750 folders, corresponding to different people. Once again, the breakdown of images per human is not uniform.

Sample images from the dog dataset

Sample images from the human dataset

Metrics:

The model should be evaluated as mentioned before using accuracy (what percentage of images in test dataset were correctly classified) and also using Multi Class Log Loss. Multi Class Log Loss, such as cross entropy loss is as important if not more important than accuracy as we are using a dataset with uneven numbers of images belonging to each class.

A problem with Multi Class Log Loss however is interpretability. F1 score, which uses weightings to deal with data imbalance can be a useful tool as it offers a level of interpretability not provided by MCCLL.

II. Analysis

Algorithms and Techniques:

Multiple algorithms are used in the building of our dog detector app.

We require an algorithm to determine whether the image provided contains a human, dog or neither. We use a pre-trained VGG-16 algorithm, that has been trained on the ImageNet dataset and is an accurate classifier of 1000 categories, of which dogs and humans are included.

A human face detector is also needed for this we use a face specific pre-trained Haar feature-based cascade classifier from Open CV.

A pre-trained CNN is used later on in transfer learning as CNNs are proven to be useful in image classification. The model chosen was ResNet50, a model that has won image classification competitions in the past. Since our dataset is not large enough to train a deep neural network, transfer learning works extremely well in this case.

Benchmark:

The Dog Breed Classification can be benchmarked vs. models on the Kaggle competition page. Our “from scratch” model also serves as a method of benchmarking; one would expect a transfer learning model to

perform much better than a model built from scratch. A naïve predictor could even be used to give a baseline benchmark. For the purpose of this project, a minimum of 70% accuracy was required.

III. Methodology

Data Preprocessing:

Data was pre-processed using Torchvision transforms. Different transforms were used for training and validation/testing datasets. Different transforms were used on the training dataset to make random alterations to the image to increase reduce bias while training the model.

For training data:

1. Image was resized to 256 pixels
2. Random crop of the image to 224 pixels
3. Random flip of image horizontally
4. Randomly rotate the image within a range of 10 degrees
5. Image converted to tensor
6. Image normalized with (mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])

For validation/testing data:

1. Image was resized to 256 pixels
2. Centre crop of image to 224 pixels
3. Image converted to tensor
4. Image normalized with (mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])

Implementation:

To train the "*from scratch"* model, 5 convolutional layers of the same format were used from 3 channels to 256 channels increasing.

3 -> 16 -> 32 -> 64 -> 128

1. Apply batch normalization to reduce data to 3 channels.
2. Then, for each convolution layer,
    a. Apply 2d convolution to apply filters to data and create feature map.
    b. RelU used activation function the ensure all outputs of conv layer are positive.
    c. Max pooling is applied to reduce dimensionality by selecting the most important features from given pools.
    d. Finally, batch normalization is applied to speed up and increase stability by normalizing the output of pooling.
3. After convolution layers, a dropout layer is included to prevent overfitting.
4. Add a fully connected layer to output 133 features

To train the Dog Classifier, transfer learning is used as a refinement of the initial implementation of the "*from scratch"* model. A **ResNet50** pretrained model was decided on. Shown below is a diagram of the architecture of Resnet50.

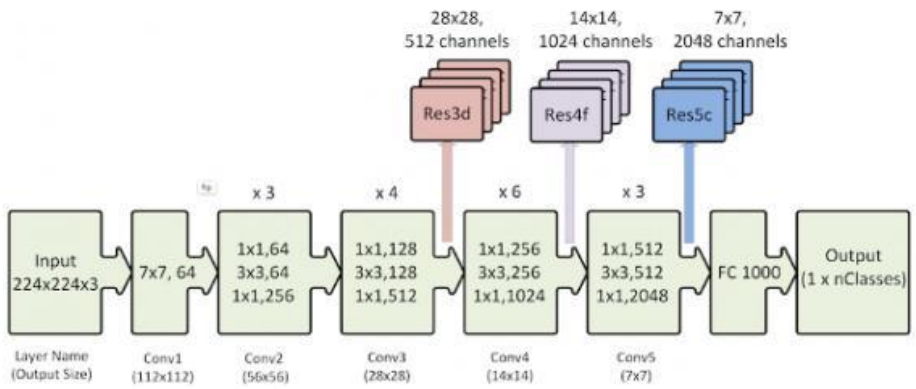
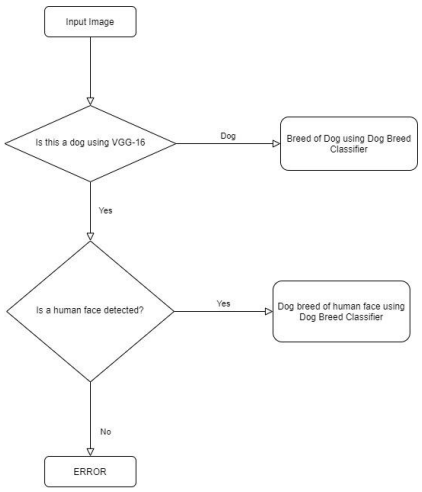


Image link:https://www.eenewsautomotive.com/news/resnet-50-misleading-machine-learning-inference-benchmark-megapixel-images

Since the model is pre-trained, freezing parameters and training only the Fully Connected layer was sufficient. The loss function specified is **Cross Entropy Loss** and the optimizer used is **Adam Optimizer,** with a learning rate of 0.001. The training ran for 10 epochs with the training and validation loss outputted at every epoch.

Once the models are decided upon, the app is implemented as follows:



1. An image is passed in to the application, our VGG16 CNN determines whether or not the image contains a dog
2. If it does, we pass this image to our transfer learning model. Here the breed of dog in the image is outputted.

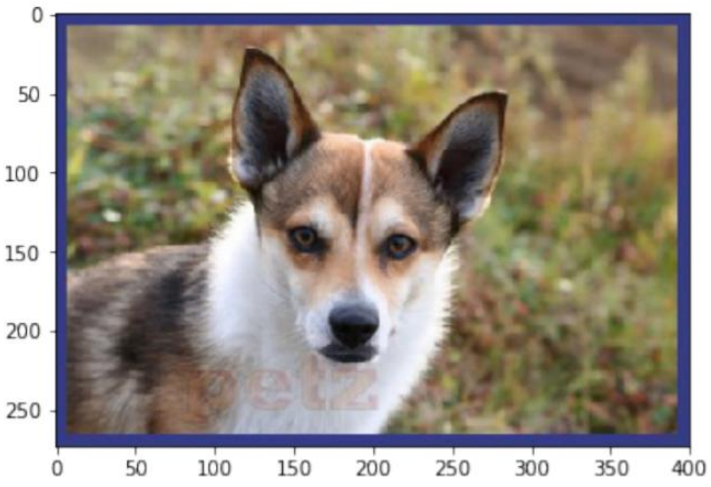
3. If there is no dog present, the input image is passed to the human face detector, if a human face is detected, the app will output what breed of dog the human face looks like. Otherwise, an error is returned.

IV. Results:

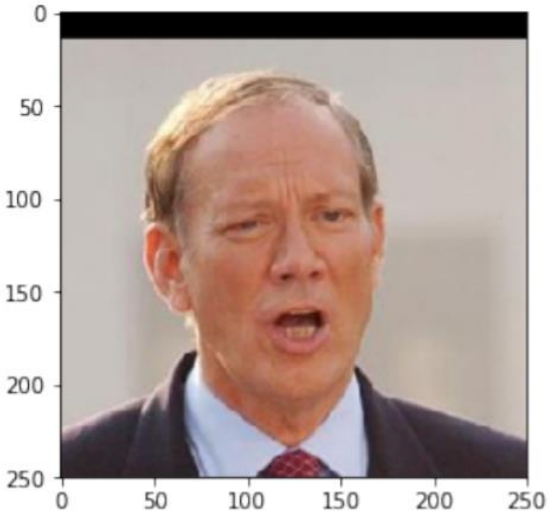
The transfer learning model finished with a Training Loss of 1.107469 and a Validation Loss: 0.615962. These results can be hard to interpret on their own, however we know that our scratch model finished with a Training Loss of 3.590706 and a Validation Loss of 3.597659. Benchmarking our two models is possible since the both use the same loss function Cross Entropy Loss. Our transfer model has a significantly lower Cross Entropy Log Loss and stands out as a worthwhile model.

The transfer model accuracy is 83% versus 21% from the scratch model. As mentioned earlier, accuracy is not a perfect metric, however used in conjunction with Cross Entropy Loss, it is a useful tool in measuring the usefulness of a model.

A more rudimentary test was performed for testing the dog application. 10 dog images and 10 human images were inputted to the app. The app performed well, as expected for dogs and did not misidentify any human faces.




```
The breed of this dog is Norwegian lundehund
```




```
The dog breed this human resembles is:  Bull terrier
```

V. Conclusion:

Reflection

The goal of this project was to create and app to take in an image of a human or a dog and make a correct classification. With that in mind, I feel this app was a relative success. Predicting dog breeds is not an easy task and some images of different breeds of dogs look very similar to the untrained eye. As for assigning a dog breed to an image of a human, it is very difficult to assess how well that aspect of the app worked. What breed of dog a human resembles is definitely open to interpretation, but maybe at a push you could see a resemblance between some of the images?!? For example, I think the man in the above example looks more like a bull terrier than a chihuahua.

Improvements

Potential improvements to this app:

1. Explore other transfer learning model alternatives: Resnet50 was first trained in 2015 and there are likely models trained in the past year or two that could prove better classifiers of dog data. Another consideration with Resnet50 is, due to its depth, it can be slow to train.
2. An increased number of epochs could be used to further improve accuracy.
3. A different optimizer or a different learning rate of Adam optimizer could be tested and implemented if necessary
4. More layers could be added on to the transfer model as well as the fully connected layer.

References:

1. Dog Breed Identification, Kaggle https://www.kaggle.com/c/dog-breed-identification/overview

2. ImageNet, https://www.image-net.org/

3. Resnet, https://arxiv.org/abs/1512.03385

4. Haar feature-based cascade classifier, https://github.com/opencv/opencv/tree/master/data/haarcascades

5. EE News Automotive, https://www.eenewsautomotive.com/news/resnet-50-misleading-machine-learning-inference-benchmark-megapixel-images