John (Christian) Vega
Nicholas Zermeno
Math 445
12/08/2022

Final Project: Dog Breed Classifier

Introduction

Our project involved classification of dog breeds from images of 120 different breeds. THe link to the dataset can be found at this url: https://www.kaggle.com/datasets/jessicali9530/stanford-dogs-dataset.
Given the task of this project was significantly large and we had a significant time constraint on which to finish, we decided to stick to one type of model to try to solve the problem. That model is known as a Convolutional Neural Network. The challenges that were foreseeable for this project included that we had a lack of computing power and a lack of images for a network of such a scale to have enough examples to properly learn from. Another significant challenge we faced was specifically regarding configuration of hardware to run networks this massive. After spending much time and effort to have tensorflow utilize the gpu, John was unsuccessful at being able to configure his graphics processing unit to be able to run the networks at optimal performance. This constraint made the task of development and testing much more difficult.

Related Works

According to Wang and Xin at url:
https://jivp-eurasipjournals.springeropen.com/articles/10.1186/s13640-019-0417-8
One method of classifying images used in this paper is using what is known as time frequency composite weighting. The way that time frequency composite weighting works is by frequency domain and time domain weighting based on blurred image data. Features are classified by this algorithm according to different criteria including point features, line features and regional characteristics according to the representation of these features on the image data, depending on the region size of the feature extracted it is either classified as a global or local feature. After the features are extracted using this technique, they are then run through a classifier based on a convolutional neural network. The network is then evaluated based on the M3CE loss function with the SoftMax function as the discriminant. The backpropagation algorithm is then run on the entire network in order to derive the gradient of the entire network.

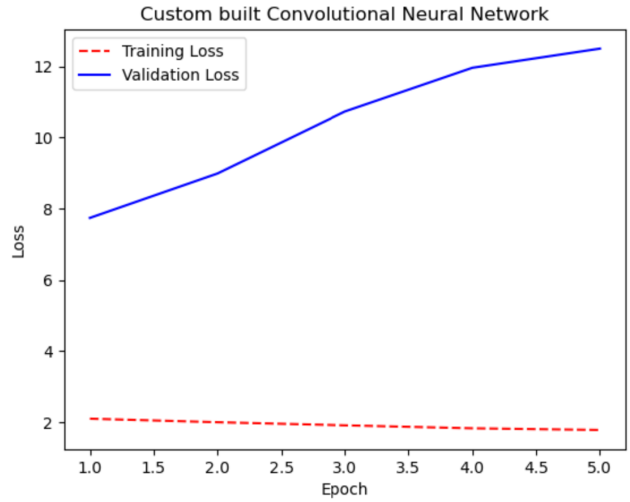
Methodology

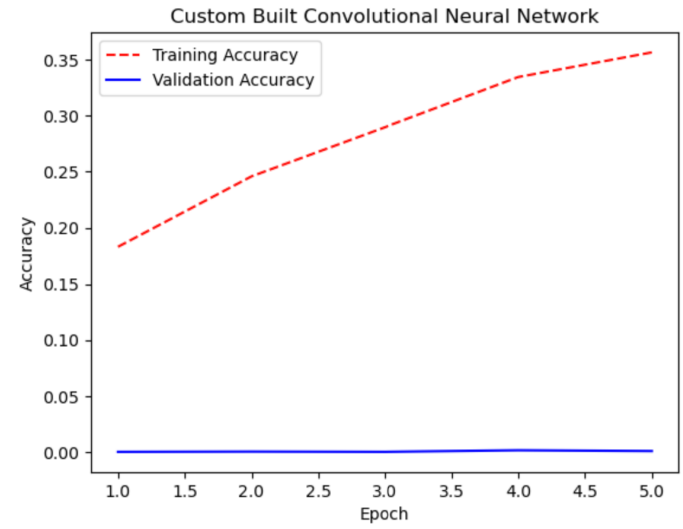
The preprocessing of data for the task of image classification is much different than processing data that are more fit to traditional machine learning algorithms. Unlike missing values and values in the wrong format we often encounter in traditional machine learning problems, the issues we have to address when dealing with image data are not nearly as intuitive. For example, one has to have a thorough knowledge of the methodology of deep learning neural networks to know in advance that a massive amount of examples are required for models to properly learn how to classify objects. In the case of this project we had around 20 and a half thousand images relatively evenly distributed among the 120 different breeds. To address this issue of insufficient number of examples we used image augmentation. Image augmentation is a technique that is used to artificially expand the size of a dataset by creating modified versions of the images in the dataset. In addition to this, the images were also in a very raw form, meaning that the pixels of

the image data contained a fair amount of noise. To address this problem we ran various filtering methods to the image, which included sobel edge detection as well as non local means filtering and median filtering. Sobel filtering works by specifically looking for vertical as well as horizontal lines in images. The non local means filter works by removing noise from the image while preserving sharp images, the tradeoff with this filter though is that textured regions of an image will lose detail. The median filter analyzes pixels in a certain local neighborhood and evaluates if the pixel value is representative enough of that region, if it is deemed as not representative enough it reassigns the pixel value as the median of that region. Given the size of our data set after augmentation even with only 10 of the 120 breeds, we had serious time constraints on run time due to time complexity of running the median filter and non local means filter on every image in the 10 breeds we selected to test. Due to this we only ended up testing our images to feed the network with the sobel edge detection. The structure of the network we built consisted of a 2D convolutional layer with 16 filters and a 3 by 3 kernel size, the padding was set to same to put a one pixel shell around the image with an input shape of 200 by 200 by 3. The images were shrunk from their original size using interpolating area technique. The next layer is a max pooling layer with a 4 by 4 pool size and strides 4 by 4 with padding as same. The second convolutional 2D layer has 32 filters with a 3 by 3 kernel size and padding same. In the middle of the second layer is a max pooling layer identical to the first layers max pooling layer. A dropout is used of 0.1. After this another convolutional layer is used with 64 filters on a 3 by 3 kernel size followed by a global average pooling layer. Next is the flatten layer followed by the 10 node dense output layer with softmax activation. The other layers all use the relu activation function.

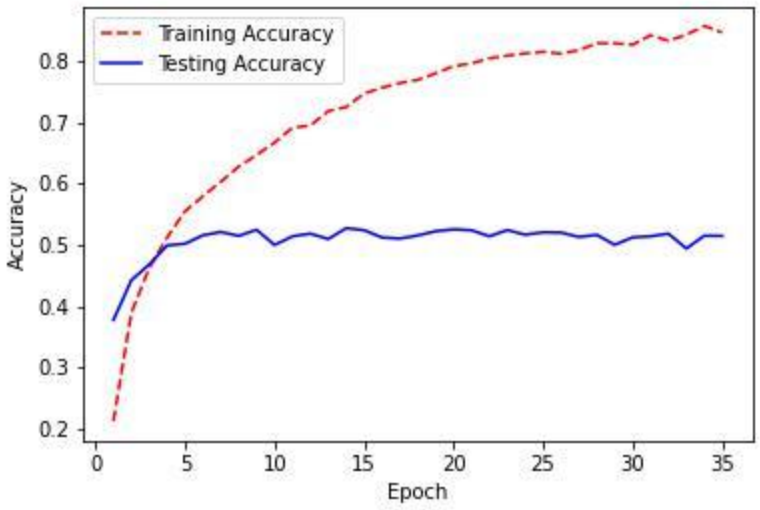
Results

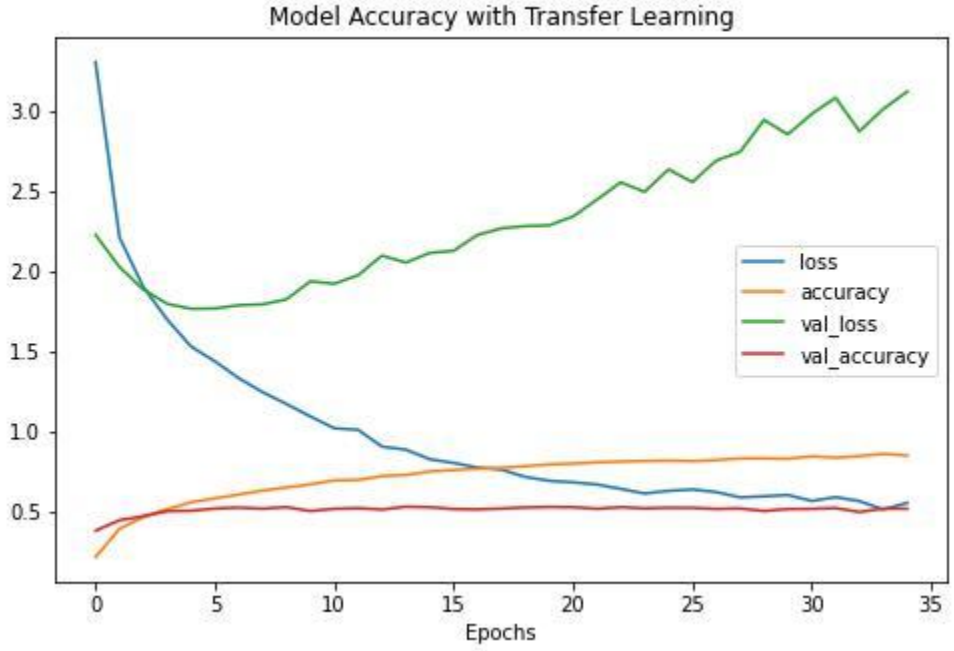
We started building our convolutional neural network on the entire dataset consisting of 120 different breeds and initially got a poor model of about 1 percent accuracy. To improve on this we introduced augmentation to our data to feed our network more examples and as a result could only run 10 breeds due to hardware limitations. Our results improved, but still left much to be desired with an accuracy of about 35% with signs of overfitting. We can clearly see this evidence of overfitting in the graphs below with the custom built network, The validation loss is much greater than the training loss and the validation accuracy is much lower than the training accuracy





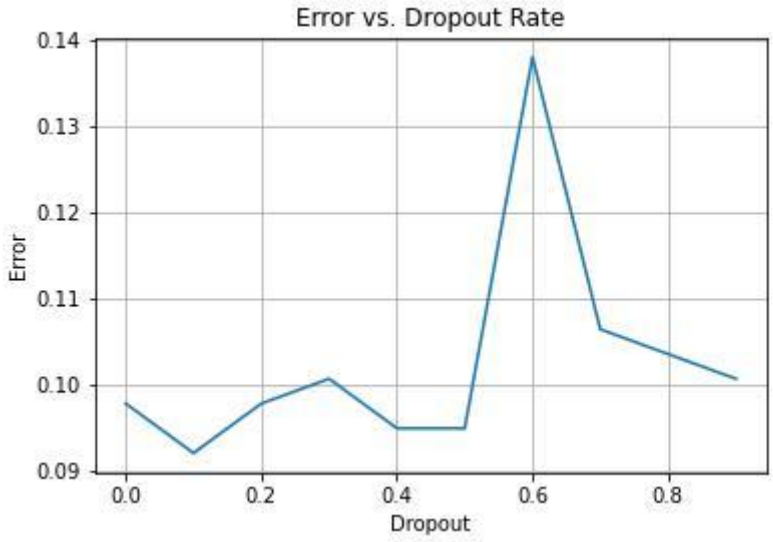
Undeterred, we decided to use a model that used transfer learning. This allowed us to achieve a solid foundation in which we could build upon. A computational constraint existed for this process as we were unable to fine-tune our model that we had constructed ourselves; it took about an hour for us to fit our data onto our model and if there was an error it proved to be quite difficult to make many adjustments. Consequently, we were able to train a model of 84.66% on the training data and 51.5% on the testing set as shown below.



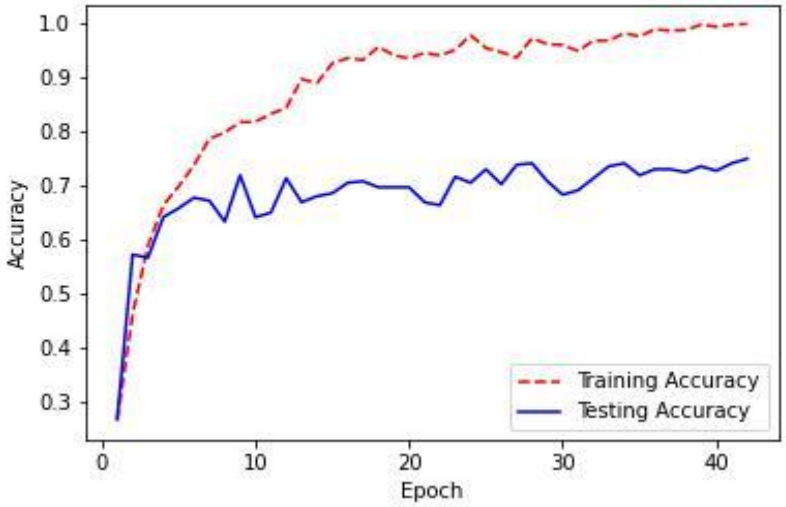


The loss function was calculated by using categorical crossentropy as our categories were one hot encoded. Clearly, there were signs of overfitting occurring in this model so we had to continue and hopefully achieve better results. So we decided to build upon a model that used augmented data in its fitting. This enabled us to train on a bigger dataset while still using the original images. There was a computational constraint, however, as we were unable to load the data onto my graphics card which was processing all the information. The epochs were taking anywhere from 2 seconds to 1 minute and a half using my graphics card and John was simply unable to run more than 5 considering he was using the processing power of his CPU. This

meant in order to feasibly continue, we had to reduce the dataset to a selection of 10 different breeds. The 10 breeds were not chosen in any order, we simply drew from a list and decided to build upon those.
After experimenting with a couple of different models with different dropout rates for the convolutional neural network, we decided on using one with a metric of 0.1.



Putting all of these tools together, this 10 breed model with the transfer learning and augmented data was able to achieve an accuracy of 100% on the training set and 75.1% on the testing set.



Still, there was evidence of overfitting as there was still a 25% discrepancy between the testing and training data. This model did perform better than our original attempts though. Something to note is it could have been possible to train a model that had very poor accuracy. This would be similar to how our homework went with the digits data; if we introduced classes that are similar such as 1's and 7's, in this case breeds that are similar, then we would certainly have a harder time being able to distinguish between them.

Conclusion

In conclusion, we believe we would certainly be able to increase the performance of our original model if we had the computational means to do so. Our small dataset would certainly benefit from the increase in data and allow our model to learn more and thus perform better. Many models are trained on millions of different samples and that is how they are able to achieve the accuracies that they can, but we did not have access to that type of data for this project. A 25% error isn't so bad considering what we were working with. The images were very noisy in that they had multiple dogs, people, strange positioning and much more which would have certainly needed more than 20,000 images to fully capture. We learned a lot from this project on how something we learned from class can be applied to real world applications. Everyday we can experience something that uses computer vision such as unlocking our phones with our face or having an online album of pictures that is able to recognize different people that are in it.