Final Report

Tarun Singh and Edward Tyles
CS 4824
Virginia Polytechnic Institute and State University
Blacksburg, VA 24060

Abstract

Binary classification is the task of classifying the elements of a set into two groups on the basis of a classification rule. A project by Furkhen Gulsen focused on using deep learning with python to create a binary classification to differentiate between humans and horses. This binary classification system was not perfect, and our goal for this binary classification problem was, after recreating the project, to leverage research from other papers to improve the accuracy of his binary classification system. The first aim of our project was to recreate the results from the paper by Gulsen, as well as to improve on his methods to increase the accuracy for binary classification in our own project. In our project we classified bikes and cars, rather than using Gulsen's datasets, to compare the effectiveness of his model and have a potentially practical application in the realm of self-driving cars. After recreating his paper, we achieved higher accuracies with his model for our dataset, implying that our images were well suited for his model and easier to classify. For the second aim of our project, we implemented a more complex model known as VGG16, which uses a 16 layer pre-trained neural network, and used that with a re-trained top layer. This methodology allowed for a significant improvement on our test and validation accuracies.

Project Setup

The first part of our project was recreating a deep-learning binary classification Kaggle project by Gulsen - available at https://www.kaggle.com/codeblogger/deep-learning-with-python, while repurposing it to classify cars and bicycles to compare the end accuracy of the model. The project we recreated used a 2-layer neural network with datasets to classify images of humans and horses. This was done with forward and backward propagation in the learning process to find weights and biases for the LNN to yield better predictions. The second part of our project was finding a research paper about LNN improvements for image classification, and using its methods as a basis for improving our original project's accuracy.

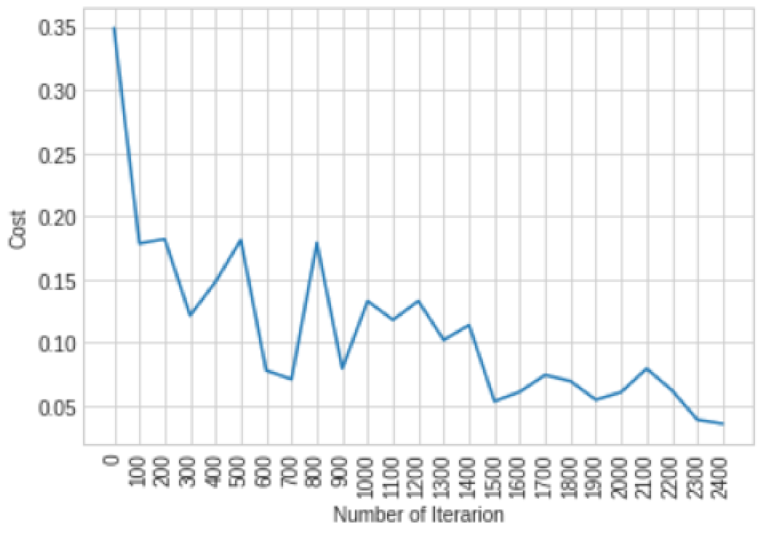
For our data, we selected datasets of cars and bicycles, with about 10,000 images in each, at the following links:

- Cars: `https://www.kaggle.com/prondeau/the-car-connection-picture-dataset/notebooks`
- Bicycles: `https://www.kaggle.com/tysonpo/bike-ads-images-prices-specifications`

However, please note that we did not use the entire datasets - rather, we used 1,200 images from each, because the image processing logic from the Kaggle project we were recreating took too much time for us to feasibly test the program otherwise.
Afterwards, we organized the datasets in Google Drive for use in Google CoLab, processed the datasets for use in the 2-layer LNN, and then implemented the LNN.

Initial Results

We calculated the train and test error for the 2-layer LNN with our data initially split 85% for training and 15% for testing. This yielded the following result:



train accuracy: 97.6470588235294%; test accuracy: 88.33333333333333%

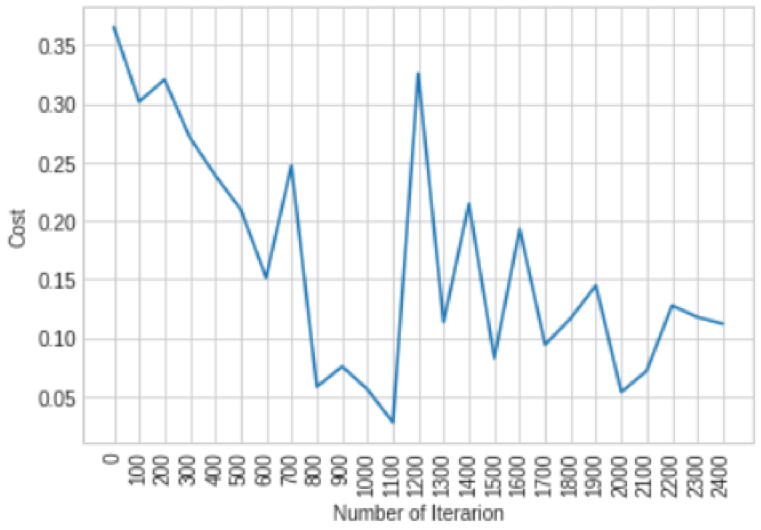
Running a cross-validation with 5 sections for the Keras LNN implementation - which had 3 layers: 2 ReLu, and 1 sigmoid - yielded a mean accuracy of: 0.8656862735748291

We noted at this point that while our initial model train and test accuracies were approximately 12% and 8% higher than the Kaggle project we were following, the Keras cross-validation mean accuracy was about the same (within 1%). This result made us think that our dataset was easier to classify with this 2-layer LNN, but that the Keras library LNN implementation was equally effective at classifying our images as it was at classifying the reference project's images. We concluded that the Keras library was more consistent and stable in its performance.

Attempts for Improvement

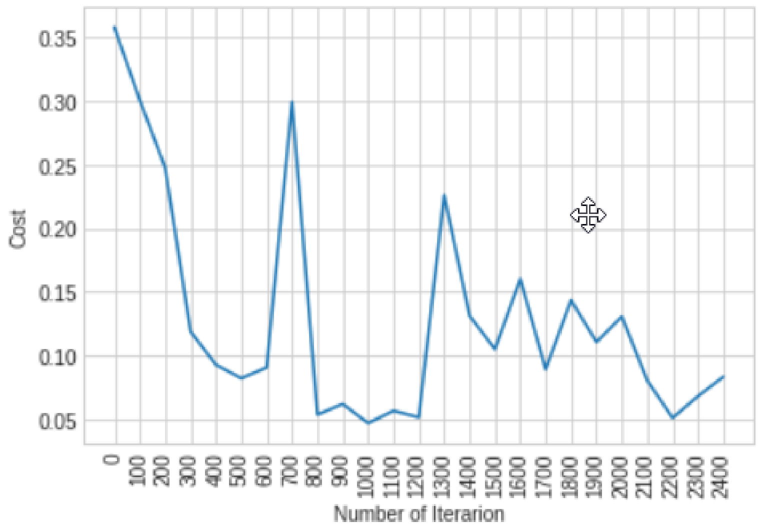
In trying to improve the test accuracy, we first tried using different test sizes compared to the training data size. As seen in the results below, there was no clear correlation between the split of our test/train data and the accuracy reached, except for when we increased the amount of training data, which seemed to result in overfitting.

With 75/25 train/test data split:



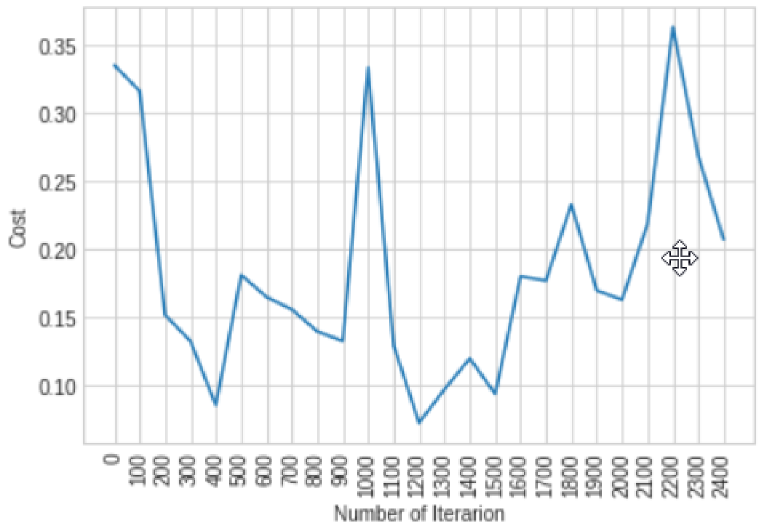
train accuracy: 96.88888888888889%; test accuracy: 83.0%

With 60/40 train/test data split:



train accuracy: 98.05555555555556%; test accuracy: 85.20833333333333%

With 90/10 train/test data split:

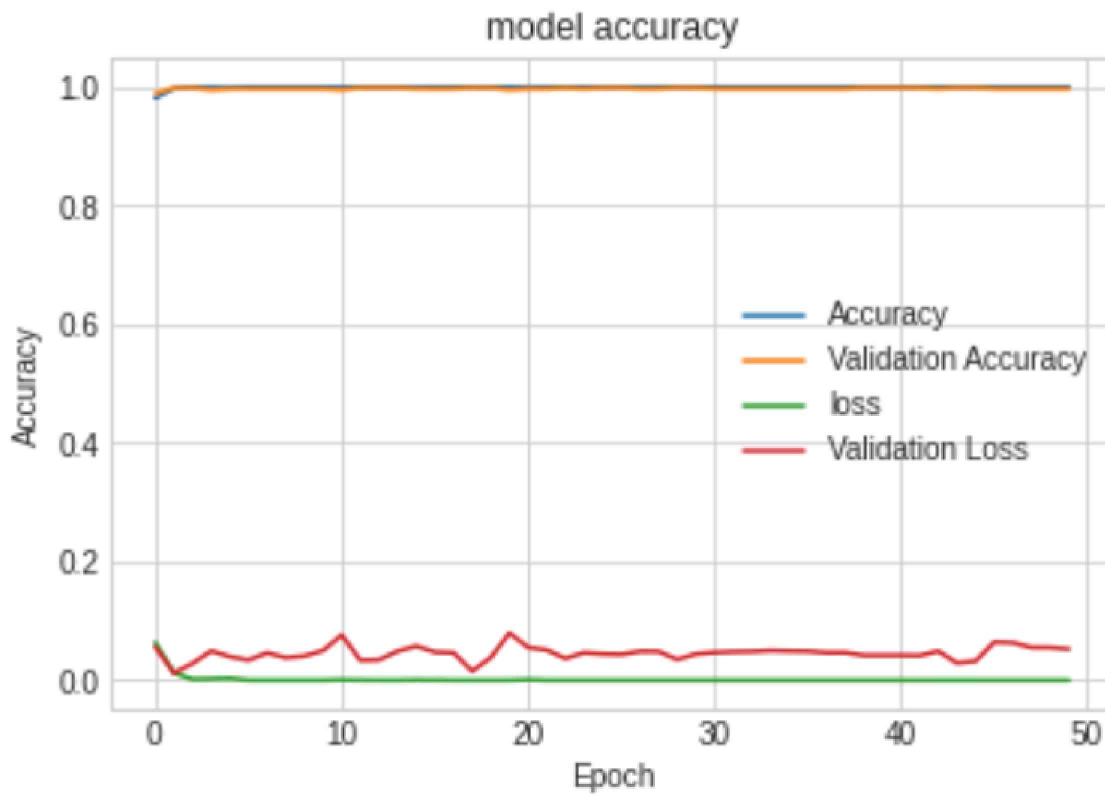


train accuracy: 78.05555555555556%; test accuracy: 71.66666666666667%

Our next step was to improve the LNN itself - this was in accordance with the proposed feedback received on our project. We decided to improve the Keras LNN by adding more layers, so we conducted research on how to improve image classification accuracy in an LNN, which yielded the paper, "Boosting image classification through semantic attention filtering strategies" by Enrique Fidalgo et. al. (https://doi.org/10.1016/J.PATREC.2018.06.033). This paper referenced methods of how to improve image classification by using multi-layered, pre-trained models that we decided to implement ourselves.
We found that they used the pre-trained VGG16 model to improve results, and followed their reference to the Keras model of it available online. The model we used was released in the publication, "Very Deep Convolutional Networks for Large-Scale Image Recognition" by K. Simonyan and A. Zisserman (arXiv:1409.1556).

Final Results

We adopted the VGG16 model by pre-pending it, exclusive of its top layer, to a LNN structure of 1 ReLu layer and 1 sigmoid layer. We then re-trained the new top layer structure of the model, and achieved approximately 99% test accuracy when initially fitting our model with a 75/25 train/test split, and approximately 100% accuracy rates when using 5-fold cross-validation. The results of the cross-validation are shown in the graph below.



These results were better than expected, as we did not expect train or validation accuracy to reach 99%. We predict that this is because the VGG16 model was particularly well optimized for our dataset of cars and bicycles. We were satisfied with this high performing model, and decided that VGG16 was an excellent base model to use for binary classification of cars and bicycles.

Contributions

Tarun researched other classification methods, studied different papers, implemented the code for filtering and resizing images, organized the spotlight presentation, and helped write the paper. Edward organized the datasets to work in Google CoLab, implemented the code for the LNN, tested train/test data splits, implemented VGG16, and helped write the paper.