CS5600 Bee Project

Bridger Jones

November 7, 2021

Introduction

This project was focused on training ANNs (Artificial Neural Networks) and CNNs(Convolutional Neural Networks) to classify images of bees and audio of bees. The networks were trained on different sets of data and then evaluated on other sets reserved for testing. Overall this was a very big learning experience and took quite a bit of computation time. I think my desktop will be happy that I am finished with this project.

Technology Specifications

This just a brief list of the hardware that was used for training and evaluating these models. My GPU is not CUDA enabled so I had to use my CPU.

1. Operating System: macOS Big Sur 11.6
2. Processor: 3.7 GHz 6-Core Intel Core i5
3. Memory: 40 GB 2667 MHz DDR4

Experimentation

Before I get into the more detailed nitty gritty of the models that I settled on. I spent quite a bit of time just messing with the models themselves and finding a balance of what generally worked and what did not. Since I was working with a CPU, I knew that time was potentially going to be an issue. There were a few frustrations that were both comical and irritating. I wrote a script that would sequentially run all of the python training scripts. I set this to run overnight and set my computer to remain awake until it finished the process. I woke up the next day to find that while I had trained and evaluated all of these models, I had forgotten to call the save method in my pipeline. These mistakes did however let me write a pretty good script to handle all of the training while I was away at classes for the day.

Some general discoveries that I found was that batch sizes of 8,16, and 32 were the easiest to train and keeping my epochs to 50 made it so my primary computer was not training for days on end. At one point I attempted to train for 300 epochs but that was taking really long to do one model and did not seem to get me better results. I also decided to go with a learning rate of 0.01 for all of my models. After I decided to stick with these hyperparameters, I decided to use this as the basis of my test for all of these models. I wrote a pipeline model for all of these that trained and tested each network on batch sizes of 8,16, and 32. The networks with the best mean accuracy across the different datasets were the ones that were saved. So for my final training and evaluation step, I wrote a script that would do this while I was gone all day at school. The entire process took about 6-7 hours according to my logs. The only model that was not setup this way was the keras one. I found that the keras model took a lot longer to train and was a lot more demanding on my CPU.

One of my crazy, random attempts to make my models more accurate was to increase the layers. I think I had 5 hidden layers within my ANN at one point and it pretty much bulldozed into an unbreakable rock. I never got to see how it actually performed because of how long it was taking. I let it run for about 8 hours to see if it would finish and it didn't in that time. Unfortunately I also need to use my CPU for other things like work and school so I couldn't just let it run forever.

Another random attempt was to increase the number of neurons. I increased my max neurons to 512 on each layer at one point. That eventually got to a point where each epoch was taking about 30s each to train. I also don't think the lower neurons were ever really being trained well because it did not increase the accuracy.

Image Classification

4.1 ANN Analysis

The artificial network was set with with an input layer, two 128 neuron layers with relu activation and then an output layer with softmax activation. The optimizer was a Stochastic Gradient Decent with batch sizes varying from 8,16,32. The learning rate was 0.01. The loss function was categorical crossentropy. The model was trained on three datasets in sequence for 50 epochs each.

4.1.1 Training Accuracy Data

Dataset	8	16	32
BEE1_gray	0.539	0.560	0.572
BEE2_1S_gray	0.779	0.805	0.806
BEE4_gray	0.776	0.791	0.783

4.2 CNN Analysis

The CNN was setup with an input layer with relu activation, a pool layer, a 128 neuron layer with relu activation, and then an output layer with softmax activation. The optimizer was a Stochastic Gradient Decent with batch sizes varying from 8,16,32. The learning rate was 0.01. The loss function was categorical crossentropy. The model was trained on three datasets in sequence for 50 epochs each.

4.2.1 Training Accuracy Data

Dataset	8	16	32
BEE1	0.613	0.674	0.686
BEE2_1S	0.892	0.904	0.902
BEE4	0.723	0.728	0.729

4.3 Keras CNN Analysis

This model was a CNN created with keras. It used a different optimizer and a few other features that the other CNN did not use. This was trained on only one dataset and only for 50 epochs with a batch size of 64. A key difference with this model compared to the other ones is that it used the Adam optimizer and L2 regularization in the form of weight decay. The learning rate was set at 0.01 like the other models but it also had a weight decay rate of 0.001. This model still used categorical cross entropy as its loss function. This model took quite a while to train on my computer and maxed out at around 400% of CPU usage at the peak of training. I did not want to run this one too many times as it made my computer really work hard. I did not train this one multiple times with different variables so its data will be displayed in the final model data section. Overall it had really good performance based compared to the other models.

Audio Classification

5.1 ANN Analysis

The artificial network was set with with an input layer, one 128 neuron layers with relu activation and then an output layer with softmax activation. The optimizer was a Stochastic Gradient Decent with batch sizes varying from 8,16,32. The learning rate was 0.01. The loss function was categorical crossentropy. The model was trained on three datasets in sequence for 50 epochs each. Training on the audio seemed to take a lot longer than training on the images.

5.1.1 Training Accuracy Data

Dataset	8	16	32
BUZZ1	0.440	0.443	0.441
BUZZ2	0.630	0.627	0.629
BUZZ3	0.725	0.743	0.736

5.2 CNN Analysis

The CNN was setup with an input layer with relu activation, a pool layer, a 128 neuron layer with relu activation, and then an output layer with softmax activation. The optimizer was a Stochastic Gradient Decent with batch sizes varying from 8,16,32. The learning rate was 0.01. The loss function was categorical crossentropy. The model was trained on three datasets in sequence for 50 epochs each.

5.2.1 Training Accuracy Data

Dataset	8	16	32
BUZZ1	0.566	0.563	0.570
BUZZ2	0.620	0.658	0.668
BUZZ3	0.728	0.735	0.738

Model Selection

To select the final models, the mean was taken of the accuracy performance on each dataset. The best one was saved across all the different batch sizes. This is the final data for the selected models and their evaluation against the testing data set.

6.1 Best Image ANN Evaluation Data

The best image ANN was the one that had a batch size of 32.

Dataset	Accuracy
BEE1_gray	0.572
BEE2_1S_gray	0.806
BEE4_gray	0.783

6.2 Best Image CNN Evaluation Data

The best image CNN was the one that had a batch size of 32.

Dataset	Accuracy
BEE1_gray	0.686
BEE2_1S_gray	0.902
BEE4_gray	0.729

6.3 Best Image Keras CNN Evaluation Data

The best image CNN was the one that had a batch size of 64. A key difference to note is that this model was trained using the Adam optimizer instead of the SGD. Its evaluation accuracy on BEE4 was 81.68%.

6.4 Best Audio ANN Evaluation Data

The best audio ANN was the one that had a batch size of 32.

Dataset	Accuracy
BUZZ1	0.441
BUZZ2	0.629
BUZZ3	0.736

6.5 Best Audio CNN Evaluation Data

The best audio CNN was the one that had a batch size of 32.

Dataset	Accuracy
BUZZ1	0.570
BUZZ2	0.668
BUZZ3	0.738

Conclusion

Overall the model that had the best performance on images was the keras CNN. Unfortunately this was the model that took the longest to train on my CPU and seemed to be the one that required a lot more computation resources. The best performance on the audio was the CNN. I learned a few interesting things from this project. One of the big things that stuck out to me the most was how it seems that we are not only limited by hardware, but by the vastness of dimensionality of the hyperparameter selection. There are so many possible different combinations I could try aside from the hardware choices itself. I can definitely see why a CUDA enabled GPU farm is going to out perform anything that I could ever do. I also really understand why this is called ”bulldozing” because there is no systematic way to predict the results. If I were to spend more time on this project I think I would have perhaps tested out some different optimizers. Adam seemed to do pretty well with the keras model. I have used other ones such as AdamW, Adadelta, and AdaGrad with some success in other classes. The audio seemed to be the hardest to train on. Without listening to the audio itself I am sure that there must be a lot of noise in the data other than buzzing.

Something I definitely struggled with on this assignment was the selection of layers and epochs. Too me it would make sense that training for more epochs should only increase the accuracy of the model. However, I found that generally, I was getting better performance around 50 epochs. This was also a fairly good balance so my CPU was training models all week. It would be an interesting project to look into heuristics for training and selecting the datasets based on performance of the model. If that is even possible. I think another interesting application of this would be the identification or classification of what causes the best recognition in the datasets themselves. I know there are quite a few transformations that can be done to the images themselves. It would have been interesting to test out some of those differences between the different models as well.