Programming Assignment 4 – Corgi CNN

Tyler Filbert

Network architecture

I decided to use 2 sets of convolutional layers, rectifier, and pooling layers. and 2 fully connected layers for my network. As for the number of channels, kernel size, and number of features, I decided these values after doing some tuning. I ran through numerous iterations of output and decided that the values that are now implemented were the best I could find. As for the specific architecture, I followed a guide to implementing a CNN with PyTorch (https://pyimagesearch.com/2021/07/19/pytorch-training-your-first-convolutional-neural-network-cnn/) and they used a similar structure. This made it easy for me to understand and work with this model. When training, I used the optimizer Adam and the loss function of cross entropy.

Implementation notes

To start with the hyperparameters, again, I did a lot of testing with these and found that 3 epohcs and a batch size of 5 was a good number to run the program quickly and also achieve good accuracies. I also used a very small learning rate of 0.0002 which helped my model not overfit. I had to tune these parameters because the model was very prone to overfitting if any of these values were too small or large, greatly affecting performance.

I decided to convert the class labels to ints to make calculations much simpler. In this script, I did not convert them back to the breeds because I just needed to compare the ints to get the accuracies.

When training, I decided to chop off any batches that didn't have my desired batch size of 5, to make things consistent and so I didn't have to add more logic to deal with a variable batch size. The rest of the training process was very standard, just processing the input and getting the loss, etc. When running through the validation and testing set, I decided to still use the provided dataset script to load the images. I just used a batch size of 1 to run through examples one by one. I found this was a great way to easily get the input in a format that is usable.

Performance

The overall accuracy of my model on the test set for an average run is 75%. These values can range from the 60s to high 70s or even 80s. This is because the model may overfit and may not learn a great amount depending on how the images are shuffled or may learn very well. I have tuned my model to not overfit greatly, so after 3 epochs the training accuracy is only around 80%, but I have found this is a good spot to prevent overfitting.

Running the CNN

To run this program, you can just run cnn.py with no arguments. To the terminal it will print epoch by epoch training and validation accuracies, and then a final test accuracy. I have provided the dataset in the zip file so it should know where the training and testing sets are and have no problem loading the images. If there is a problem loading the training and testing images, please set a valid path to testing data on line 74 of cnn.py and a valid path to training data on line 119 in cnn.py