Midterm Challenge Write up

John Salloum

AI Statement:

I primarily used AI in this midterm as someone to consult for errors and model recommendations. For part 1, the simple cnn model, my initial accuracy was only 1% which was quite low. I went to chat gpt and gave it my code, asking for areas where I may have done anything incorrectly. What I had done was set the normalization weights for train and test to different values, which caused the issue. After matching them the simple cnn model ran with around 35% accuracy. For part 2 I asked for some potential models to try out, in which it recommended the Resnet models as well as something called mobilnet, all of which I tried.

Running the code;

The code was run on a scc desktop with 4 cores and 1 gpu, the github repo being cloned everytime and the environment being set on every run.

Part 1: Simple cnn

For the simple model I decided to use 2 convolutional layers. Layer 1takes in 3 images and outputs 32 feature maps. Layer 2 takes in those feature maps and creates 64 of them. Both use ReLu activation. We then follow with our max pooling layer and our dropout layer set to a probability of 25%. Lastly we have 2 fully connected layers, going from 16,384 units to 512, and then 512 units to 100. The rest of the code defines the forward pass of the model. Initially this model had an accuracy of on;y 1%, which was due to me incorrectly implementing the data transformation. After a quick fix, we were able to achieve an accuracy of around 30%, training for 5 epochs with a batch size of 8.

Part 2: Pretrained model

For this part I decided to use REsNEt-18 as my pretrained model. Using ResNet I had to modify the model to output 100 images versus the 512 it usually does. The model was trained using a SGD optimizer, one we used for a majority of the models, as well as a l;earning rate of 0.001. The same batch size and number of epochs were also used. This model performed about the same, if not slightly worse with an accuracy of 29% on kaggle. The more interesting runs were to come in part 3

Part 3: Pretrained model with Weights

Model: Resnet18

For our first run we used the exact same parameters as part 2, only now we introduced pretrained weights here. The first run with the same parameters as before proved fruitful, bolstering an accuracy of roughly 36%. At this point, the goal was to fine tune the ResNet-18 model using different parameters. We had 4 more separate runs of this model, each with 25, 50, and 100 epochs respectively. We also changed the batch size from 8 to 16, 32, 64, and even tried 128 on one run to no avail. The combination that produced the highest accuracy was a batch size of 16, 50 total epochs, and a learning rate of 0.001 using the sgd optimizer. This gave us an accuracy of 43.8%, our highest by far. The rest were within 1-2%, but that combination gave us the best result.

Model: Resnet34

After the first few runs I decided to try and run Resnet-34 to see if there were any improvements. However, despite the deeper architecture, the model did not show significant improvement. The best validation accuracy I achieved with ResNet-34 was around 41-42%, which was slightly worse than ResNet-18. This suggested that simply increasing model depth without augmentation or advanced fine-tuning strategies did not guarantee better performance.

Model: Mobilnet

In addition to the ResNet family models, I also experimented with MobileNet using pretrained weights, one of the models recommended to be by generative ai as I was hitting a wall with the aforementioned two models. MobileNet is a lightweight and efficient architecture, designed primarily for resource-constrained environments. It uses depth wise separable convolutions, which significantly reduces the number of parameters compared to ResNet models. This being said, this model for me ended up being the worst of all the pretrained models used, pasting an accuracy of only 37%. This is likely due to the model being originally designed for simpler classification tasks and smaller datasets, which made it less effective on our data. Likewise I could not use any data augmentation as recommended by both the professors and by fellow classmates, which made this model even harder to run.

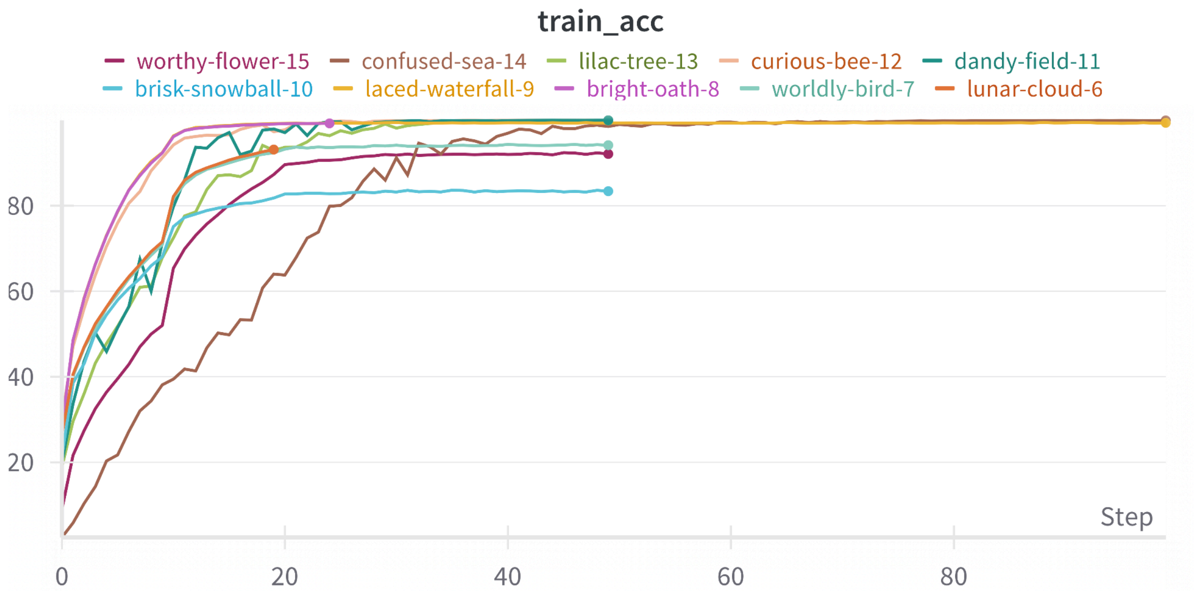
Model: Resnet 50

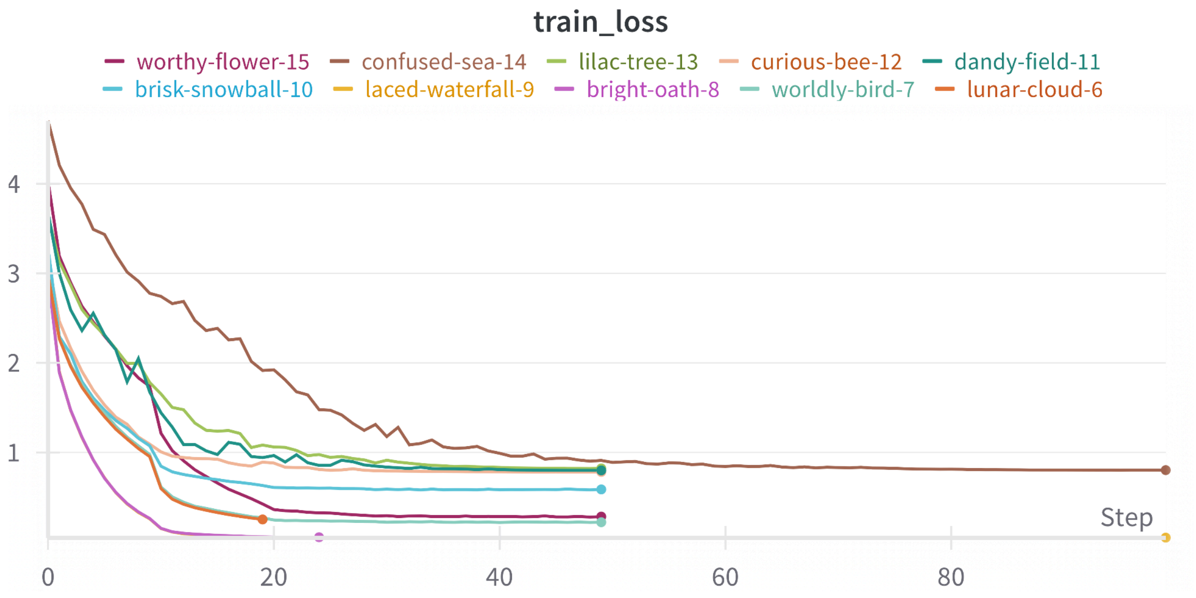
The last Resnet model tried was ResNet-50, one I decided to test out due to its deeper and wider architecture. Here is also where I switched from using the sgd optimizer to the adam optimizer. While this model did show some improvement over both ResNet-34 and Mobilnet, the accuracy did not pass that of ResNet-18. Before wrapping up, I wanted to try one more version of this model.

Model: Resnet 50 Wide

My last run was using Resnet50 wide, an architecture similar to ResNet-50 but with twice the number of channels in each layer, effectively making the network "wider" instead of deeper. The idea was that this would help with learning the patterns better. To give it time to train, i set the batch size to 16 and the epochs to 100 for the first run of this type of model. To my surprise it did not do that well and had an even lower accuracy than mobilnet of 33%. Thinking I may have overtrained, I reverted the optimizer back to sdg, the batch size to 8, and the epochs to 50, the same parameters that gave me my best result for ResNet-18. This final run had an accuracy similar to the ResNet-18 run, boasting an accuracy of 43% as well. It was behind by around .0004%, so very very close.

Train Loss and Train Accuracy Graphs;





From the above graphs, we see a wide array of different loss functions. The runs associated with worldly-bird and lunar-cloud refer to our ResNet-18 runs, the best of the bunch. Ur ResNet-50 wide model that came a close second is related with worthy-flower. All 3 of these graphs share a similarity in that they steadily decrease, have a spike decrease after 10 epochs, then slightly decrease again. For train accuracy this pattern in also present.

Takeaways:

Through this midterm, I was able to implement and test out a wide variety of models, learning how to implement and test hyperparameters along the way. The winner here was ResNet-18. Despite trying deeper and wider models, ResNet-18 consistently gave the best balance of performance and stability, achieving an accuracy of around 43-44% without any data augmentation. Increasing model depth or width beyond ResNet-18 did not significantly improve accuracy, and in some cases, even worsened it due to probabl;e overfitting with the deeper models.