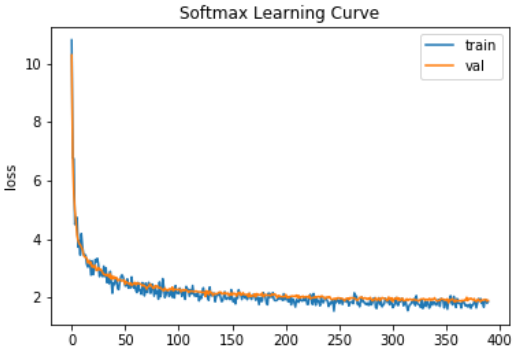
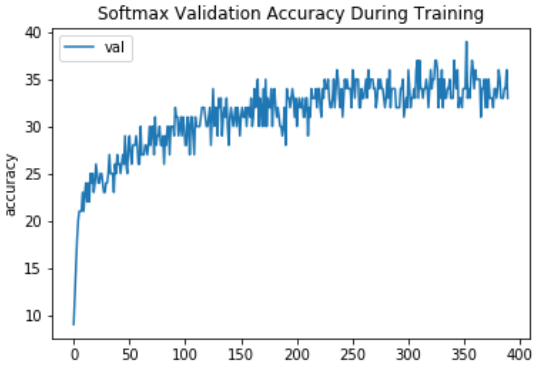
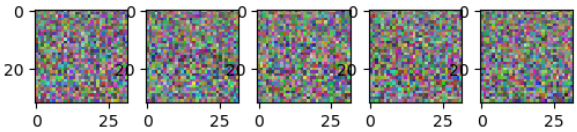
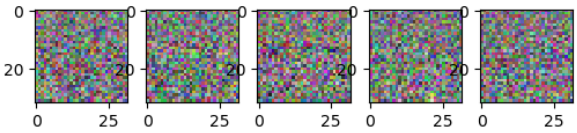
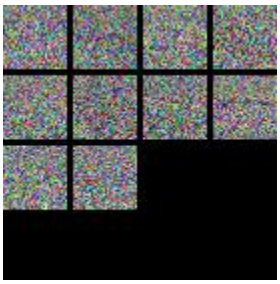
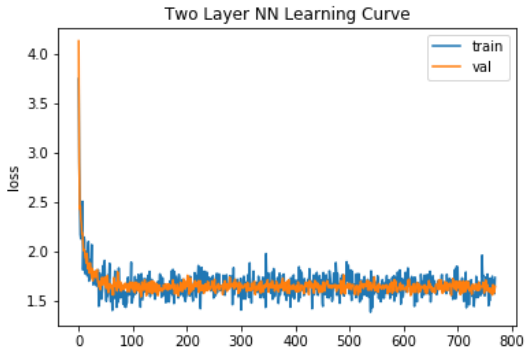

Softmax Learning Curve
train
val
loss
10
8
6
4
2
0
50
100
150
200
250
300
350
400

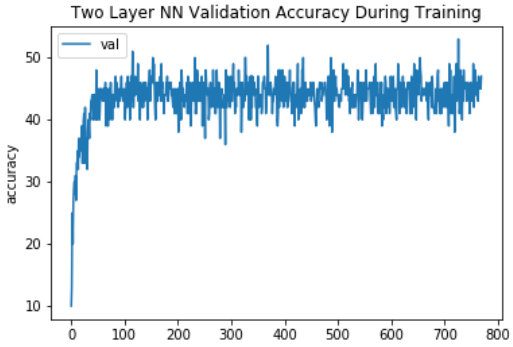

Softmax Validation Accuracy During Training
val
accuracy
40
35
30
25
20
15
10
0
50
100
150
200
250
300
350
400

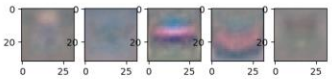

0
20
0
25

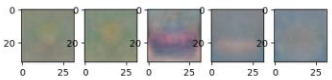

0
20
0
25

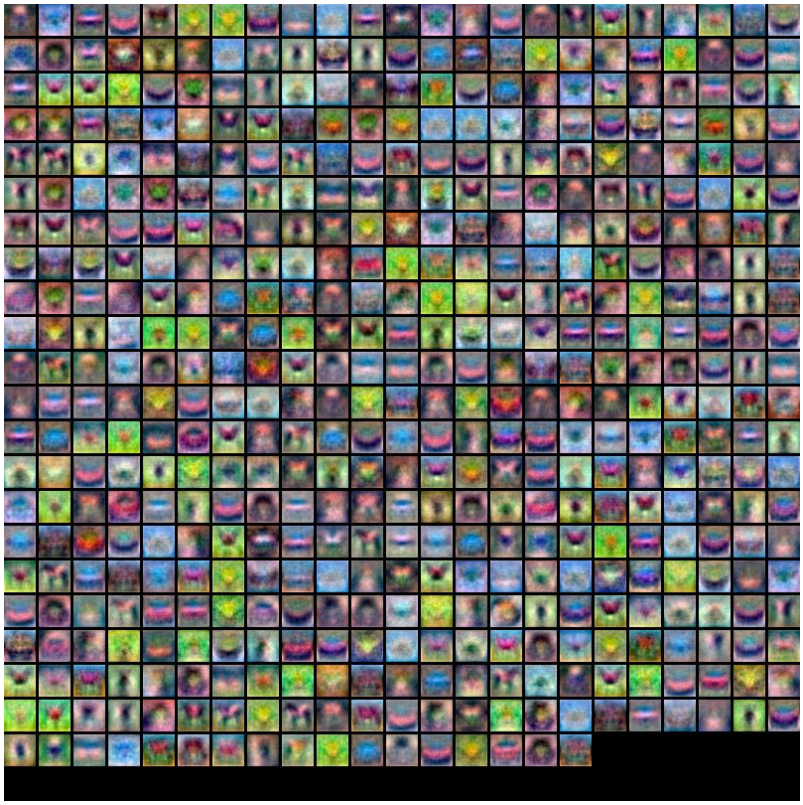


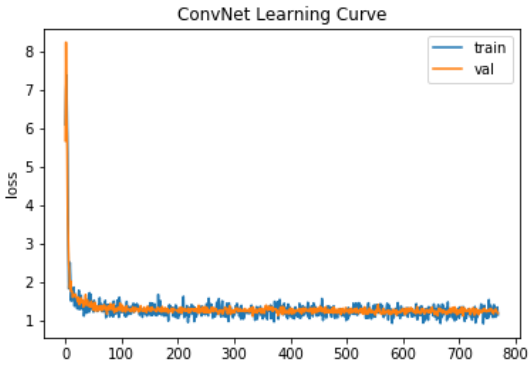

Two Layer NN Learning Curve
train
val
loss
4.0
3.5
3.0
2.5
2.0
1.5
0
100
200
300
400
500
600
700
800

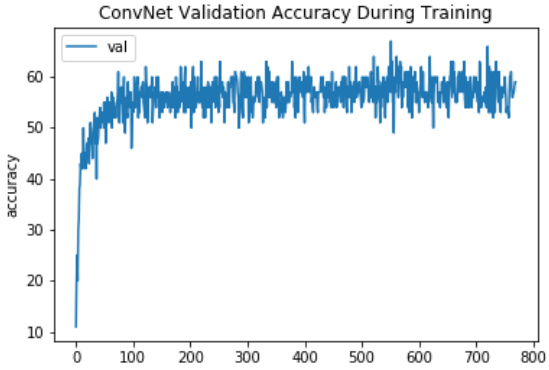

Two Layer NN Validation Accuracy During Training
val
accuracy
50
40
30
20
10
0
100
200
300
400
500
600
700
800

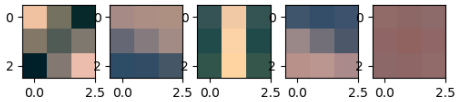


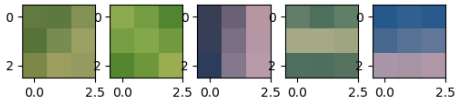


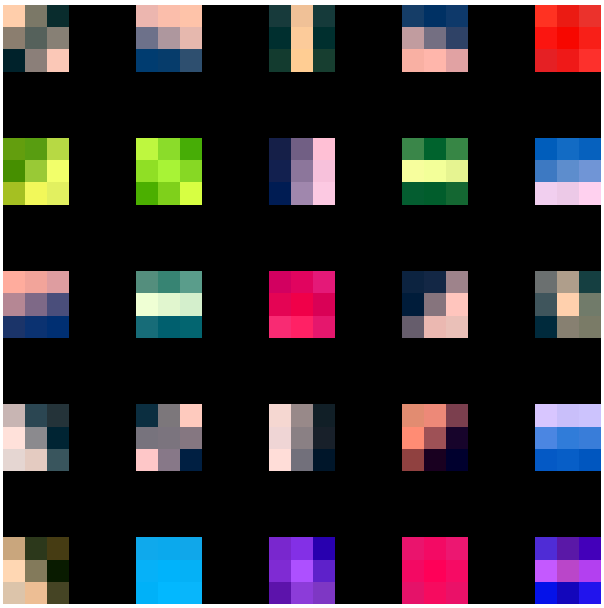


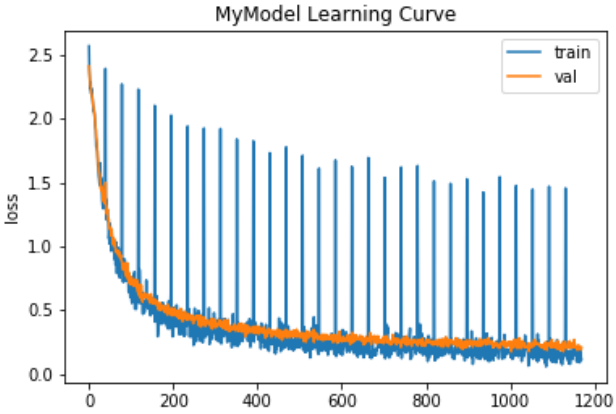

ConvNet Learning Curve
train
val
loss

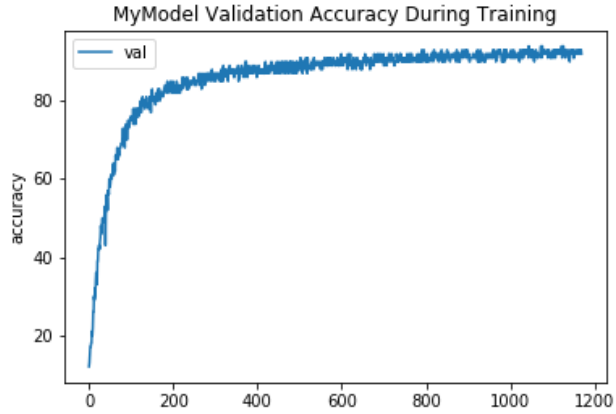

ConvNet Validation Accuracy During Training
val
accuracy








MyModel Learning Curve
train
val
loss
2.5
2.0
1.5
1.0
0.5
0.0
0
200
400
600
800
1000
1200


MyModel Validation Accuracy During Training
val
accuracy
80
60
40
20
0
200
400
600
800
1000
1200

Name: James Hahn
Email ID: jhahn37@gatech.edu
Best Accuracy: 91%

Procedure:
When I first saw this task, I didn't think twice. I knew this was a perfect task for transfer learning, since the ImageNet task is similar in nature, just with larger images and more classes. The benefit CIFAR10 has over ImageNet is that CIFAR10 has more samples per class. So, I went to PyTorch's website and found the pre-trained models (https://pytorch.org/docs/stable/torchvision/models.html). Lucky for me, there is a table with all the top-1 and top-5 accuracies on ImageNet. I basically just went down the list, tried ResNet18, VGG16, AlexNet, DenseNet161, DenseNet121. Initially, I was training this on my desktop and it was extremely slow, so I was training on either 1, 5, or 10 epochs. My accuracies were pretty poor, but I reached 76% accuracy on DenseNet161. It's important to note in this initial phase, I froze all the pre-trained layers and replaced the last FC layer with my own layer to output 10 scores for the CIFAR10 dataset; also, I resized the images from 32x32 to 224x224 for the pre-trained models. I didn't really know how to get up to a higher accuracy, most of my models were stuck around 76%. As such, I did a lot of searches online, specifically for CIFAR10 papers and Github repositories. I found a lot of articles using either their own network or pre-trained networks, but a lot of the articles were pretty annoying to read because they discussed resizing images and data augmentation and I was too lazy to do any of that. Eventually, I went to one of my other machine learning classes, and found a student from the deep learning class that had already beaten the TA's benchmark. I asked him what his approach was, and to my shock, he used pre-trained networks as well, but he was training on a significant higher number of epochs. I was kind of confused, because my computer wasn't even using all of my GPU VRAM, but eventually found out he was training on Google Colab. I went back to my apartment, plugged my code into Colab, used their GPU, and the training was at least 5x faster. I was able to train for 30 or 50 epochs and run through the pre-trained models once again. I eventually found some with 81%, 81%, 82%, and 83% accuracy. The DenseNet161 from earlier was the one with 83% accuracy, so I just stuck with that one. Initially, I was just going to stop here since it accomplished the goal of beating the TA, but I wanted to be as safe as possible when it was evaluated on the full test set. After a lot of debugging, I tested a few more models and hyperparameters and eventually found ResNet18 to work best, accomplishing 91% on the EvalAI set. Additionally, I added data augmentation by randomly performing a horizontal flip of the images with a probability of 0.5. Finally, from before, I froze all the weights, but this time around, I brought in the pre-trained weights and continued to train the entire network without any frozen layers. I think it's important to note I tried three really different approaches to finding my final solution: prior knowledge and experience in DL/transfer learning, online searches of previous results on CIFAR and similar datasets, and person-to-person communication to hear their struggles/experiences. Overall, pretty well-rounded, but ended up with data augmentation + pre-trained model.