Assignment 2: Convolutional Neural Network

Subrat Kumar Swain(2021QIZ8247)
Chandan(2021EEZ8527)

March 6, 2022

Run Instruction: 1a: *python mnist.py*
1b: *python mnist_reg.py*
2a: *python resnet_18.py*
2b: *python resnet_18_pre.py*
2c: *python resnet_tiny.py*
2d: *python visualize_resnet.py*

Trained models will be saved under models folder(once trained, same code can be used for inference). Data will be downloaded under data folder and plots will be saved to the plots folder. Trained models can be found from here: https://drive.google.com/drive/folders/1RfcxSD58yitjyNob9wnOaBIxJk3bw-p_?usp=sharing

1. (a) In this assignment, we have used PyTorch Framework for data training models and data validation images. In this training model, we are using Sequential classifier and Adam optimizer as it requires less memory and is efficient in large datasets and a lot of parameters present in the data sets. To run the program, simply run mnist.py python file. We have used batch size of 64, epochs size is 250. The following graphs are attached below.

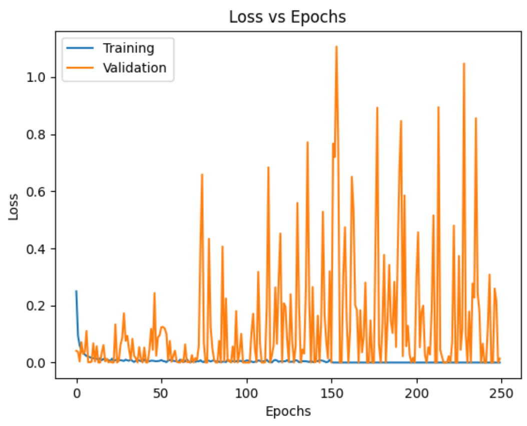

Figure 1: Loss vs Epochs

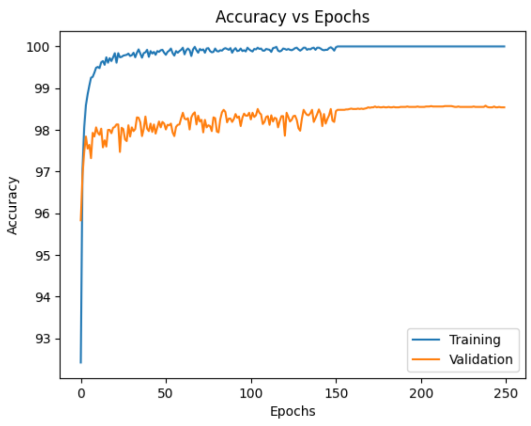

Figure 2: Accuracy vs Epochs

We found that the accuracy of our model on training images is 100% and losses is 0 as our model overfits the training datasets. The accuracy of our model on validation images is 98% and losses is 0.2.

(b) In this case, we have included all regularisations; Dropouts with (p =0.2) in our training model so that our model doesn't overfits on the training images. The script file name is mnist_reg.py file. We are using the cross entropy loss built-in function of the PyTorch module, as it is used where more than two classes have been used, like in our example, we have classification of digits (0-9).

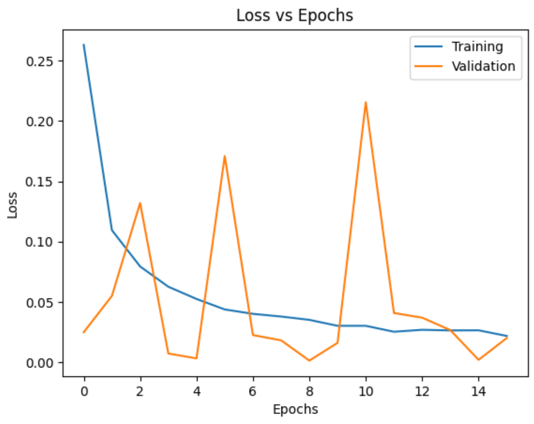

Figure 3: Loss vs Epochs

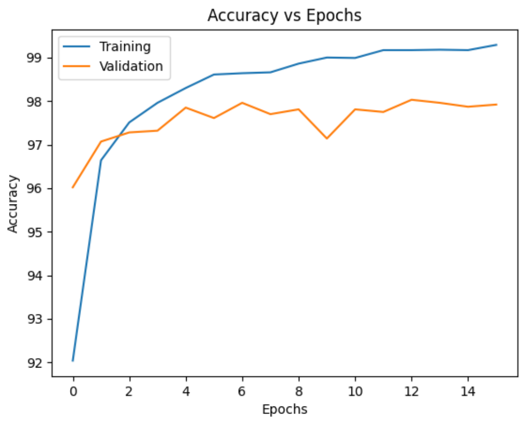

Figure 4: Accuracy vs Epochs

In this model, the accuracy of the model on the training images reaches to 99%. The accuracy of the model on test images is 98.23%. The losses on training images of the model is 0.02%. The losses on validation images of the model is lie between 0.20% to 0.05%. Even though the test performance is not significantly more than normal one, the performance is achieved with a very less number of epochs, 15 instead of 250. Also, it helped to avoid over-fitting. This is the advantage of the regularization.

2. (a) We trained a ResNet-18 model from scratch using hyper-parameters are listed below.

Epochs: 30	**Batch Size:** 128	**Initial Learning Rate:** 0.01
Gradient Clip: 0.1	**Weight Decay:** 1e-4	**Patience:** 6
Loss: Cross Entropy	**Optimizer:** SGD	**Scheduler:** Cosine Annealing

After training, we got the **Train Accuracy: 96.33** and **Test Accuracy: 77.29**(Varied between 75 to 78 in different training). We took the model with the lowest validation error. We've plotted the training curves to show the improvement through epochs Fig [5] and Fig [6].

(b) In this question, we used the pre-trained ResNet(trained on IMAGENET dataset) from torchvision models. We fine tune it using the CIFAR-10. We've documented the experiment setting and results below. Due to resource constrain, we couldn't experiment a lot.

Training last 4 layers, freezing the rest:

Epochs: 30	**Batch Size:** 128	**Initial Learning Rate:** 0.001
Gradient Clip: 0.1	**Weight Decay:** 1e-4	**Patience:** 5
Loss: Cross Entropy	**Optimizer:** Adam	**Scheduler:** None

Train Accuracy: 94.33 and **Test Accuracy:** 78.29

Training last 6 layers, freezing the rest:

Epochs: 30	**Batch Size:** 128	**Initial Learning Rate:** 0.001

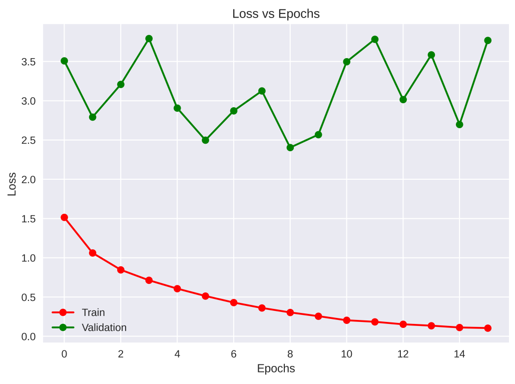

Figure 5: Loss vs Epochs

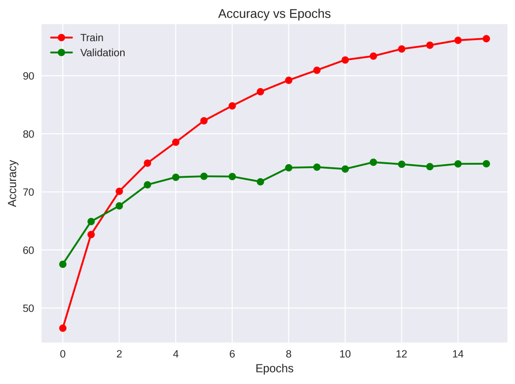

Figure 6: Accuracy vs Epochs

Gradient Clip: 0.1 **Weight Decay:** 1e-4 **Patience:** 5
Loss: Cross Entropy **Optimizer:** Adam **Scheduler:** None
Train Accuracy: 96.67 and **Test Accuracy:** 79.57

Training last 5 layers, freezing the rest:
Epochs: 30 **Batch Size:** 128 **Initial Learning Rate:** 0.01
Gradient Clip: 0.1 **Weight Decay:** 1e-4 **Patience:** 5
Loss: Cross Entropy **Optimizer:** SGD **Scheduler:** Cosine Annealing
Train Accuracy: 97.57 and **Test Accuracy:** 81.6

Training last 5 layers, freezing the rest:[Adam & Without Scheduler]
Epochs: 30 **Batch Size:** 128 **Initial Learning Rate:** 0.001
Gradient Clip: 0.1 **Weight Decay:** 1e-4 **Patience:** 5
Loss: Cross Entropy **Optimizer:** Adam **Scheduler:** None
Train Accuracy: 97.23 and **Test Accuracy:** 88.27

The highest accuracy we got from this 5 setting is **88.27**(test accuracy). This is achieved after 18 epochs by early stopping.

(c) We experiment with a different combination of data augmentation techniques. Finally, we kept these transformations: **RandomAffine, RandomAdjustSharpness, ColorJitter, RandomHorizontalFlip**. Others experimented but removed later are: **RandomCrop, RandomRotation, RandomVerticalFlip, RandomGrayscale**. Also, we added dropout after each ReLU of basic-blocks with a probability of **0.2**.

Due to the small length of training set, we couldn't get good results from this. Once we hit a test accuracy of 45% and once we crossed the 50% mark. But, overally the results of all the experiments are very less than standard setting with the full dataset.

(d) We plotted the output after each basic-block and for initial layers. We plotted some of the results below:

In the initial stage, the activations retain almost all of the information present in the initial picture. Although there are several filters that are not activated and are left blank. When we move deep, we can see the activations are capturing more abstract features rather than direct features which are less visually interpretable. Higher-level concepts such as single boundaries, corners, and angles begin to be encoded. Higher presentations provide less information on the image's visual

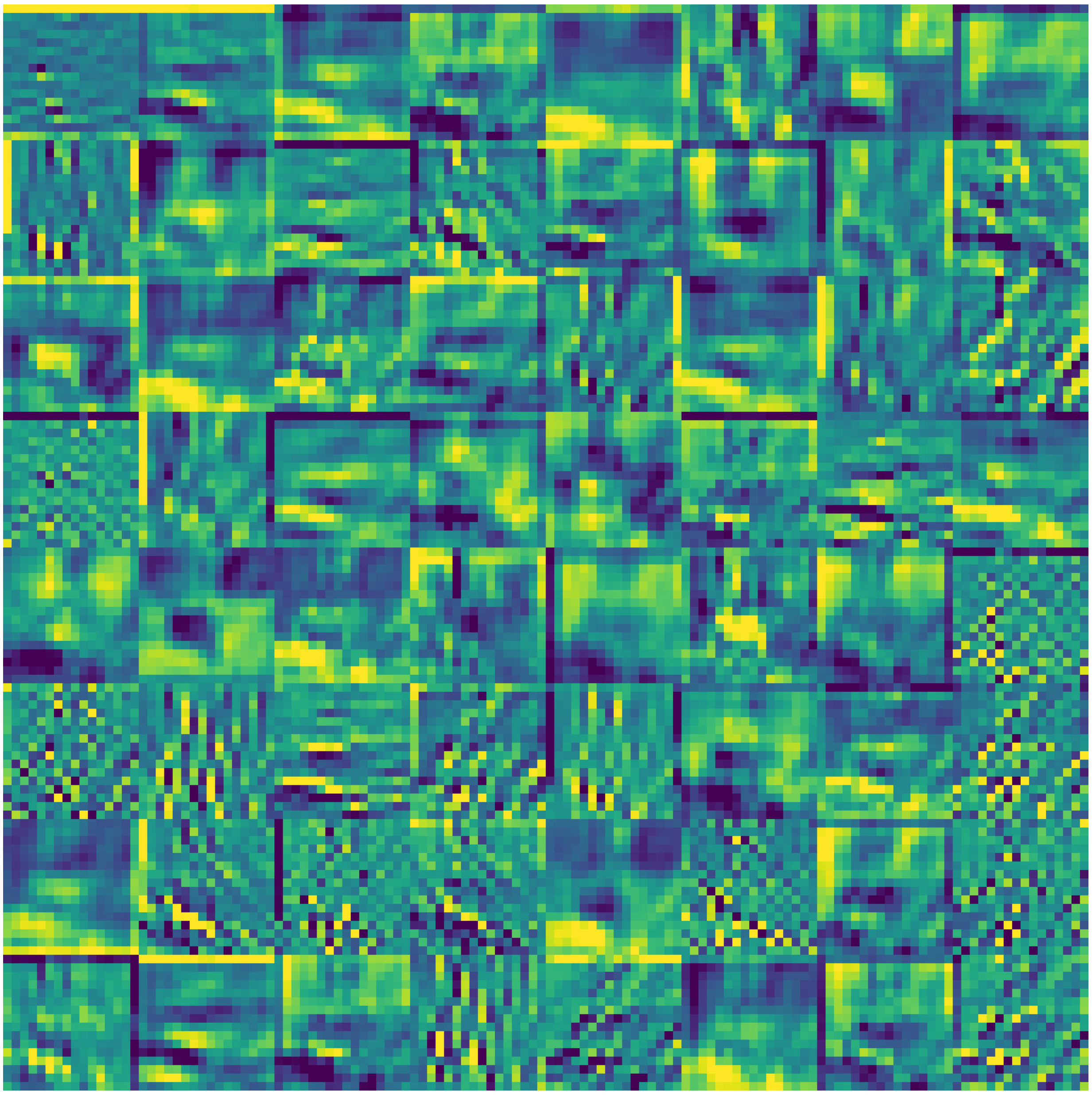

Figure 7: Intermediate Layer Viz

contents and more information about the image's class. The model structure is extremely complex to the point where our final levels are not activating at all, and there is nothing else to learn at that point. We've plotted the activations for the pre-trained one. Others can be obtained by running the visualize$_resnet.pycodewithdifferentmodels.Theactivationsaremoreorlesssameforthemall.But,thepre-trainedonegavefinerstructures,theaugmentedonegavevaguestructuresandtherawresnet(2a)gavesomethingin-between.$

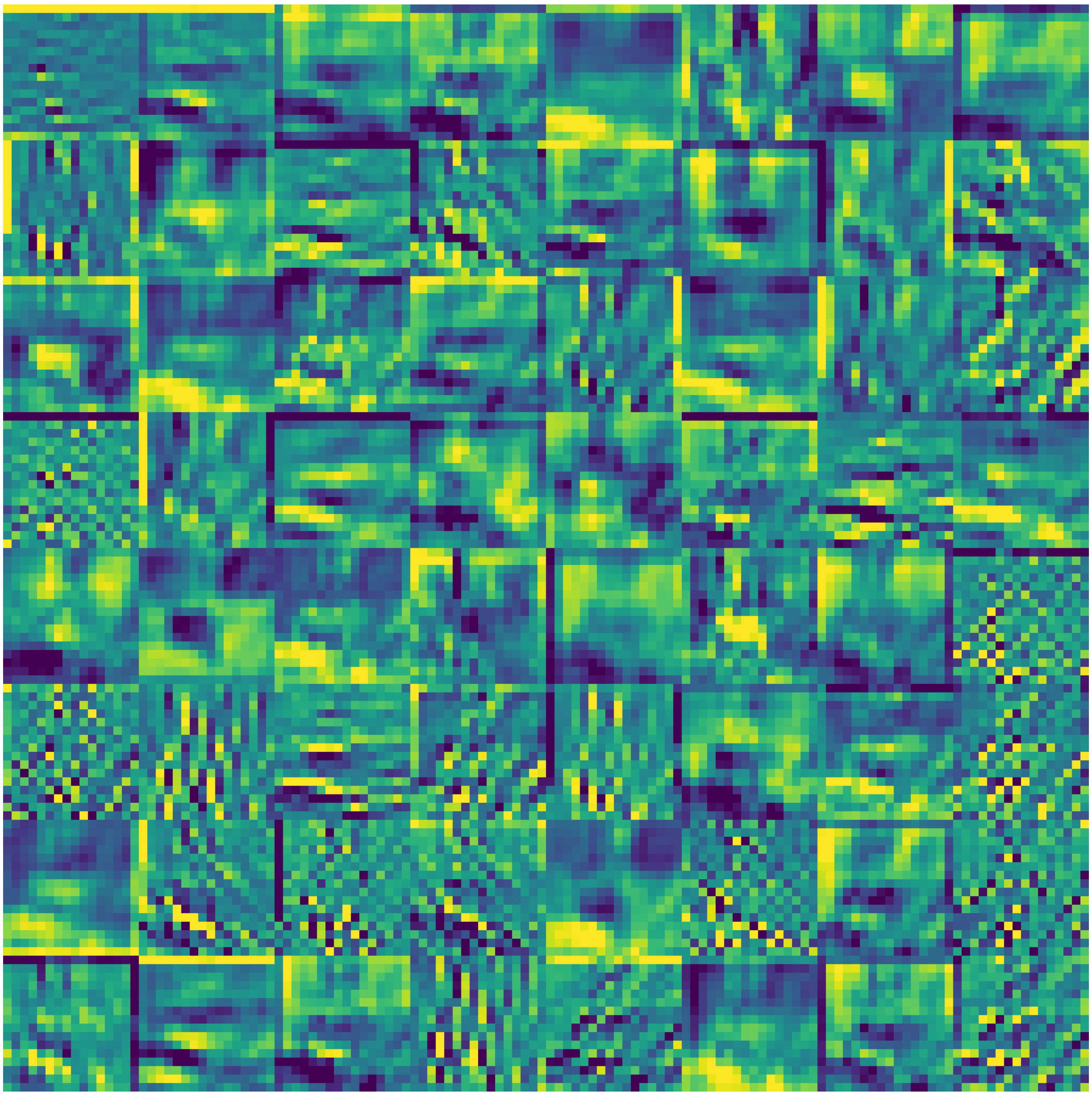

Figure 8: Intermediate Layer Viz

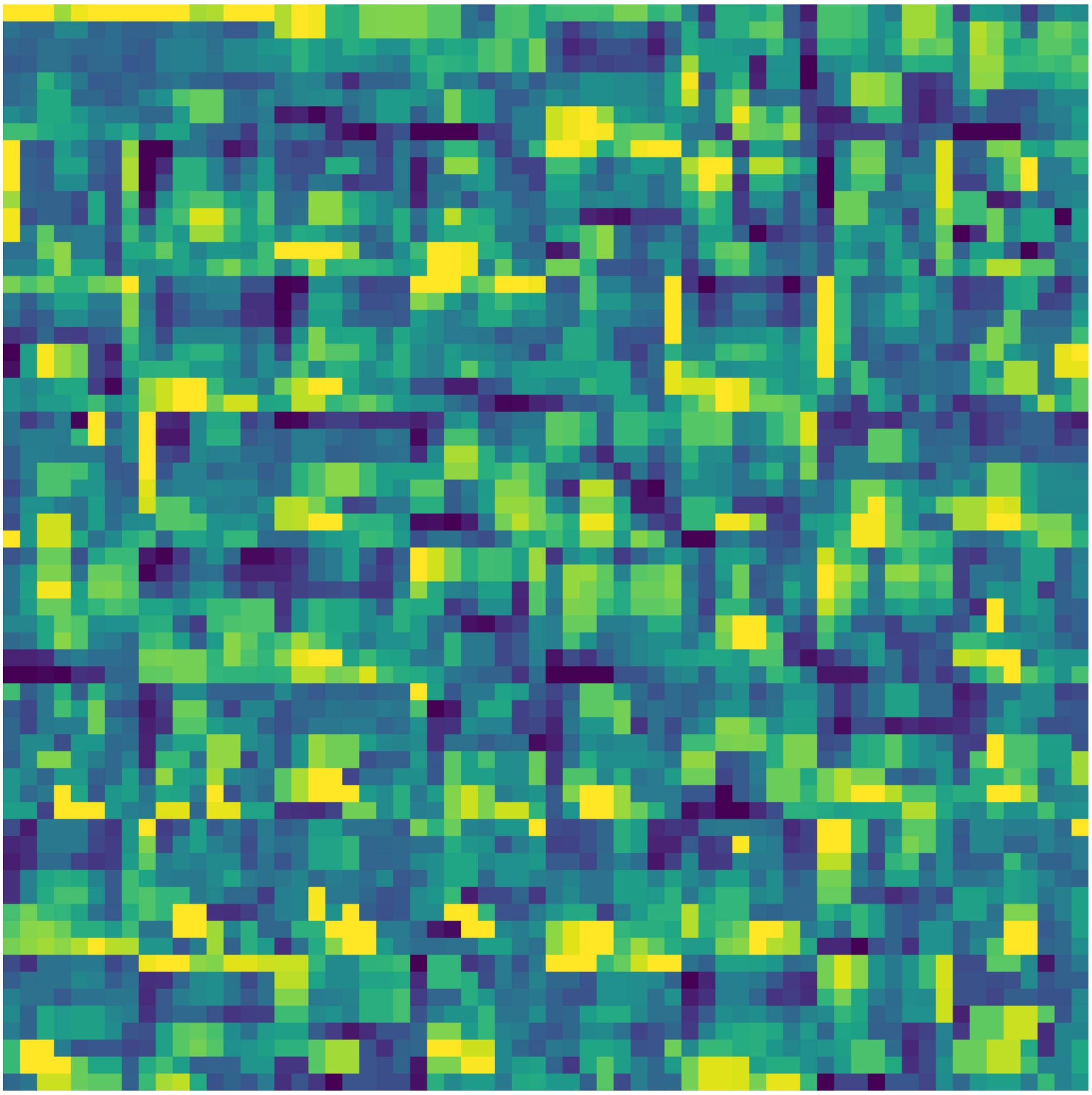
Figure 9: Intermediate Layer Viz

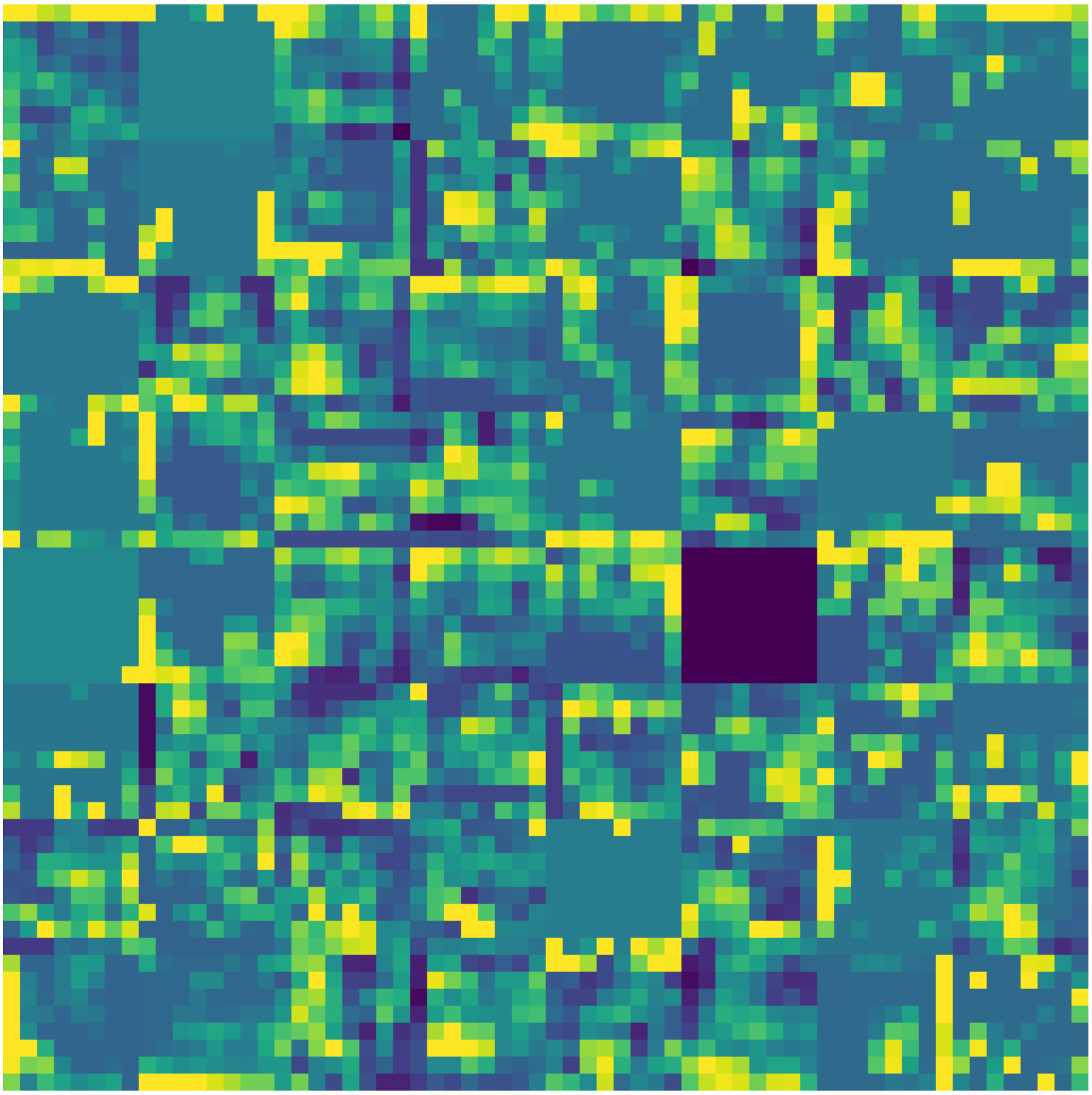

Figure 10: Intermediate Layer Viz