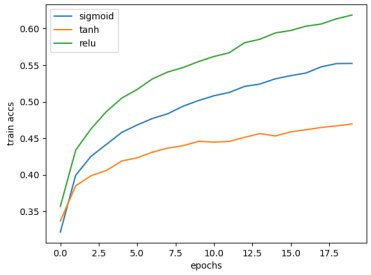
6135B-Assignment 1-Report

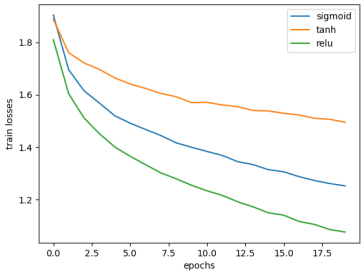
Ethan Kreuzer

September 2024

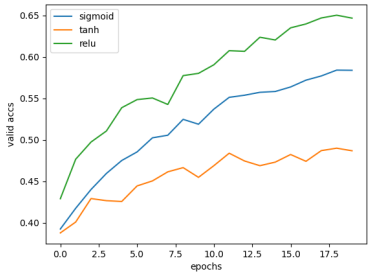
Question 4.2



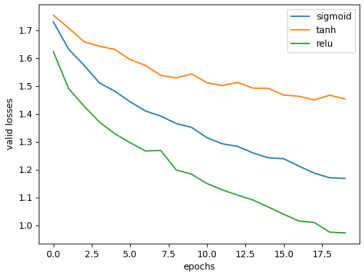
(a) Training Accuracy



(b) Training Loss



(c) Validation Accuracy

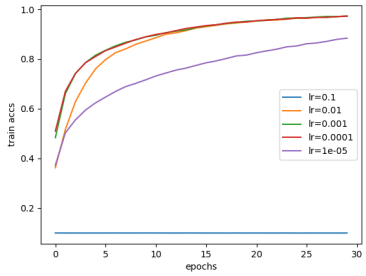


(d) Validation Loss

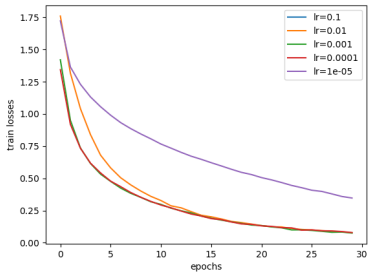
Figure 1: Training and Validation Metrics for Default MLP configuration

It is pretty definitive from the validation metrics that ReLu is the best non-linear activation, followed by sigmoid and then tahn. ReLU avoids saturation for positive inputs, which helps prevent the vanishing gradient problem that often occurs with sigmoid or tanh, due to their "flatter" regions in the function domain.

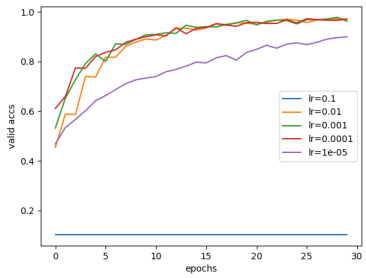
Question 4.3



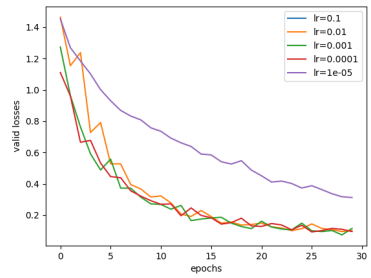
(a) Training Accuracy



(b) Training Loss



(c) Validation Accuracy



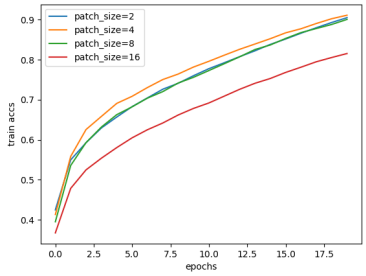
(d) Validation Loss

Figure 2: Training and Validation Metrics for varying learning rates with ResNet18

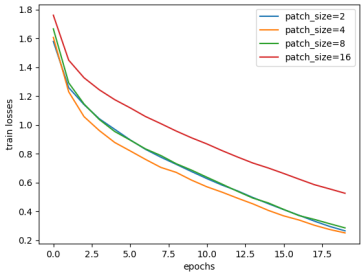
The learning rate of 0.1 is too large and training diverges. When we look at the validation metrics, it is clear that training is extremely smooth at 1e-5, but very slow and the learning rates of 0.01, 0.001, and 0.0001 strike a better balance of being smooth in training but not taking too long to train as epochs progress. The best learning rates appear to be 0.001 and 0.0001, as they are much smoother in the beginning of training compared to 0.01.

Question 4.4

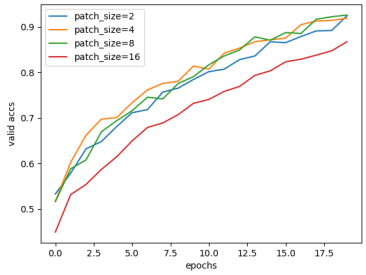
We ran the experiment with patch sizes 2, 4, 8 and 16. Smaller patches yields more parameters. The model with patch size 2 has 2376458 parameters, the model with patch size 4 has 2188298 total parameters and the model with patch size 8 2175818 parameters. We can see that across all metrics the model with the least amount of parameters, patch size 4 and 8, seem to perform the best. This may convey that patch sizes 4 and 8 provide a balance of capturing local and global structures, whereas a patch size of 2 doesn't capture enough global



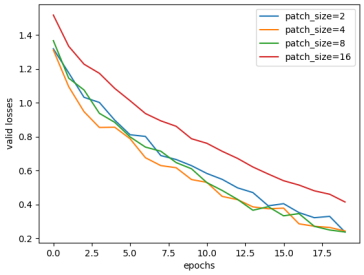
(a) Training Accuracy



(b) Training Loss



(c) Validation Accuracy



(d) Validation Loss

Figure 3: Training and Validation Metrics for varying patch sizes with MLP-Mixer

information and 16 not enough local information. Also the model with smaller patch sizes took longer to train because they have many more parameters to learn.

Question 4.5

From question 3, we saw that training the ResNet18 model with a learning rate of 0.0001 struck the best balance of smooth and fast training, so we reused this value. Furthermore, in order to avoid tuning the epoch values individually, we trained the models with early stopping so they would continue training until the best validation performance was achieved. The remaining hyper-parameter tuned was weight decay. Below is a table of results.

The best performance was achieved with no weight decay and 98.6% validation accuracy, but this did take by far the most amount of epochs to finish training.

These plots were made by taking the weights of the kernels in the first convolution layer, standardizing them so the values are in range 0 to 1 and then plotted in an 8x8 grid for all 64 kernels. In the case of the grey-scale, the values

Table 1: ResNet18 Model Performance with Different Weight Decay Values

Weight Decay	Epochs	Best Validation Accuracy
0	56	98.6%
1×10^{-3}	25	96.4%
5×10^{-5}	33	97.4%
1×10^{-4}	43	98.2%
1×10^{-5}	24	96.8%

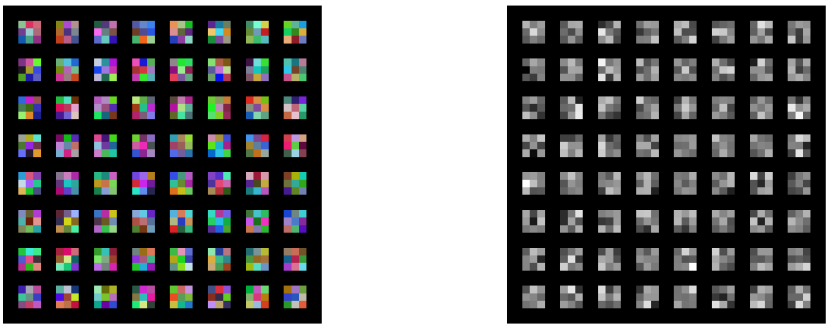


Figure 4: First Layer Kernel Visualization in RGB and Greyscale

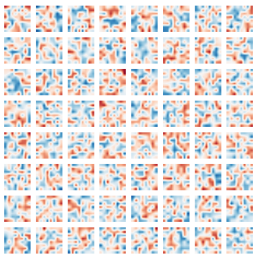
across the R, G and B channels are averaged to get a single dimension which gives a grey-scale value between 0 and 1.

Question 4.6

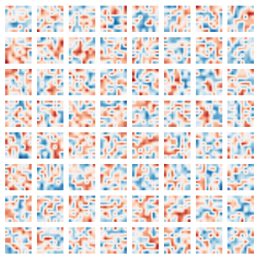
For hyper-parameter tuning the MLPMixer model, we employed a similar strategy as with the ResNet18 model. We fixed a learning rate of 0.0001, as this seemed small enough, and an embedding dimension of 256, as this yielded arguably the best results in question 7. We then tuned the number of blocks as a hyper-parameter and employed early stopping to not need to directly tune the number of epochs. The results are below

Table 2: MLPMixer Model Performance with Different number of blocks

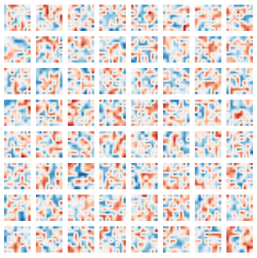
Blocks	Epochs	Best Validation Accuracy
2	70	95.8%
4	45	96.6%
6	62	97.4%
8	46	97.2%



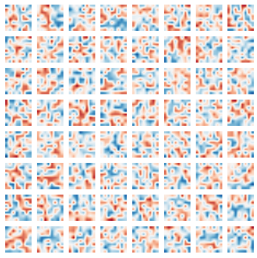
(a) Nodes 1 through 64



(b) Nodes 64 through 128



(c) Nodes 128 through 192



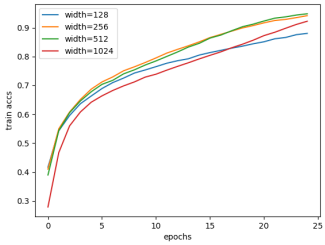
(d) Nodes 192 through 256

Figure 5: Node learned weights in first MLPMixer layer

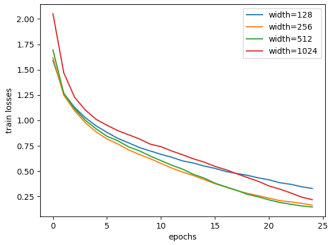
The weights of the token mixing MLP captures interactions between patches, while ResNet18 detects local edges and textures using convolutional filters. Both extract meaningful features, but MLP-Mixer focuses on patch relationships, whereas ResNet18 builds local feature hierarchies. MLPMixer performs much better than a regular MLP because its token-mixing and channel-mixing MLPs efficiently capture global and local information, unlike a standard MLP that lacks this type of spatial awareness.

Question 4.7

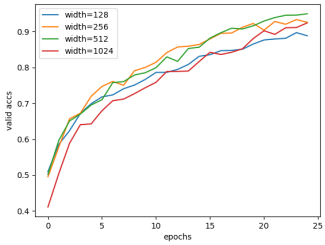
We see that when the Mixer model is trained with a hidden dimension of 256 and 512 it performs better than 1024 and 128. Mixing layers of dimension 128 may be underpowered, lacking the capacity to capture more nuanced patterns in the data, which can lead to lower accuracy. However, 1024 dimensions can introduce excessive complexity, which could make it more difficult for the model to learn important features. In contrast, 256 and 512 dimensions seem to strike a better balance.



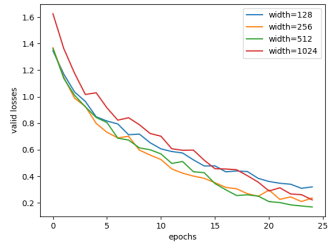
(a) Training Accuracy



(b) Training Loss



(c) Validation Accuracy



(d) Validation Loss

Figure 6: Training and Validation metrics for MLPMixer Layer Dimensions

Question 4.8

For the MLP model, we can see that the gradient norms clearly decrease as we progress through the layers, with the last layer having much smaller norms than the first layer. Furthermore, we see that the gradient norms for the layers increase in the early epochs but taper off as training progresses.

For ResNet18, we see again that later layers overall have smaller gradient norms than the earlier layers. Furthermore, in ResNet18, we see that gradient norms grow quickly and then stabilize at around 10 epochs.

A similar trend repeats itself for MLPMixer. Later blocks (layers) have smaller gradient norms than earlier layers in the model. Also, we notice again that the gradient norms grow at the beginning of training and then start to taper off. The stabilization of the gradient norms happens slower than ResNet18 and MLP and we notice that the norms are overall smaller than the both much smaller than the MLP and ResNet18 models norms. Moreover, we see that the token-mixing MLP has smaller gradients than the channel-mixing MLP.

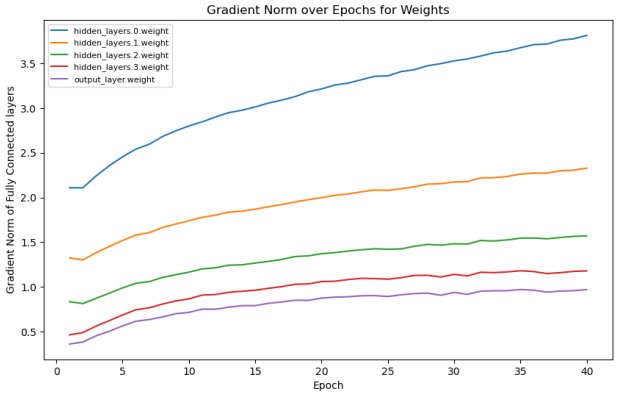
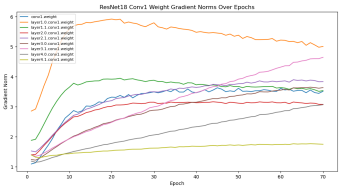
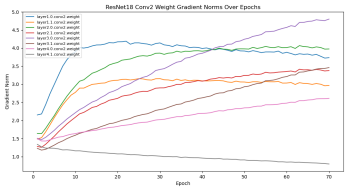


Figure 7: Gradient norms for the MLP model over epochs.

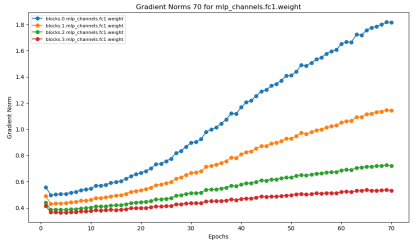


(a) ResNet18 conv1

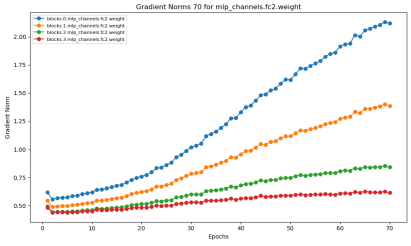


(b) ResNet18 conv2

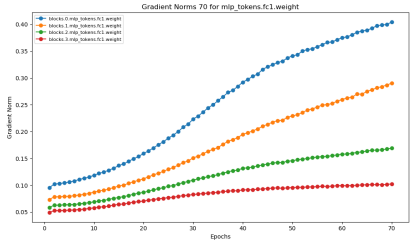
Figure 8: ResNet18 Layer Gradient Norms



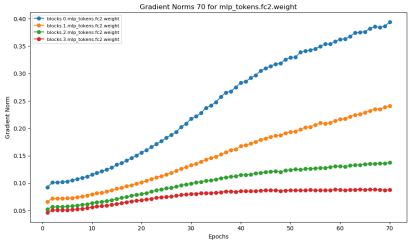
(a) Channel-Mixing first layer



(b) Channel-Mixing second layer



(c) Token-Mixing first layer



(d) Token-Mixing second layer

Figure 9: MLPMixer Layer Gradient Norms