CS6910 : Deep Learning (for Computer Vision)

Irfaan Arif
ME18B048

Assignment 2

1 Part-A

Tuning the regularization parameter

This part was to take the best performing model from Assignment-1A and experiment with various regularization parameter (by changing the weight decay parameter of the optimizer) values and to find the best one and draw some conclusions.

All experiments and results below are on the Dataset 5, mini-ImageNet with 33 classes. The models were trained on a AWS instance (g4dn.xlarge) with **batch size 4** and for **80 epochs** with a learning rate**(lr) of 0.001** with SGD, and the remaining parameters same as the one in the boilerplate code with the weight decay parameter being changed. The model with the best validation accuracy among all the epochs was taken as the result for each experiment.

The following values were experimented with for the weight decay parameter: [1e-1, 1e-2, 5e-2, 1e-3, 5e-3, 1e-4, 5e-4, 1e-5, 5e-5, 1e-6, 5e-6]
The model used is the one in Figure: 1

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
           Conv2d-1          [-1, 64, 82, 82]           1,792
           Conv2d-2          [-1, 64, 80, 80]          36,928
         MaxPool2d-3          [-1, 64, 39, 39]               0
           Conv2d-4         [-1, 128, 37, 37]          73,856
           Conv2d-5         [-1, 128, 35, 35]         147,584
         MaxPool2d-6         [-1, 128, 17, 17]               0
           Conv2d-7         [-1, 256, 15, 15]         295,168
           Conv2d-8         [-1, 256, 13, 13]         590,080
         MaxPool2d-9           [-1, 256, 6, 6]               0
          Linear-10                 [-1, 256]          65,792
          Linear-11                 [-1, 256]          65,792
          Linear-12                  [-1, 33]           8,481
================================================================
Total params: 1,285,473
Trainable params: 1,285,473
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.08
Forward/backward pass size (MB): 10.81
Params size (MB): 4.90
Estimated Total Size (MB): 15.79
----------------------------------------------------------------
```

Figure 1: Best performing model (63 % test accuracy

- Weight decay of: **1e-1**

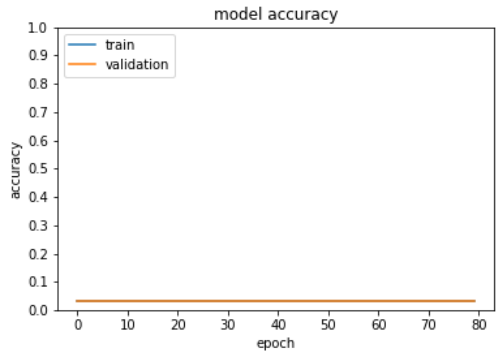

Figure 2: Accuracy curves

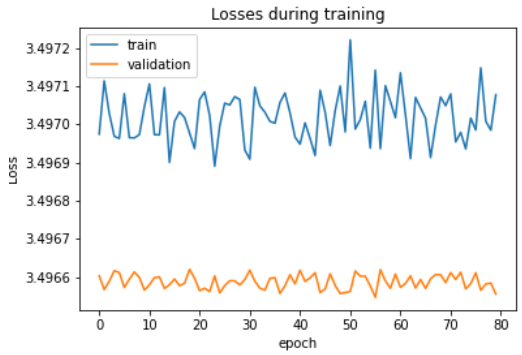

Figure 3: Loss curves

- Weight decay of: **5e-2**

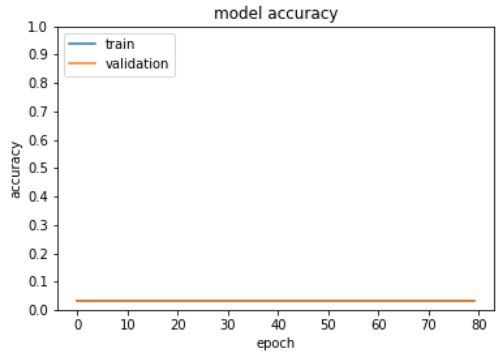

Figure 4: Accuracy curves

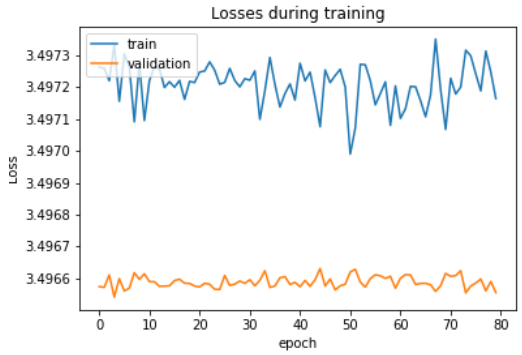

Figure 5: Loss curves

- Weight decay of: **1e-2**

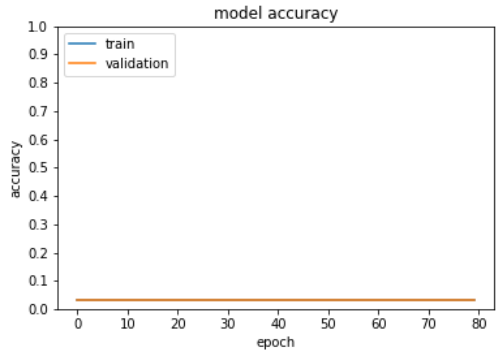

Figure 6: Accuracy curves

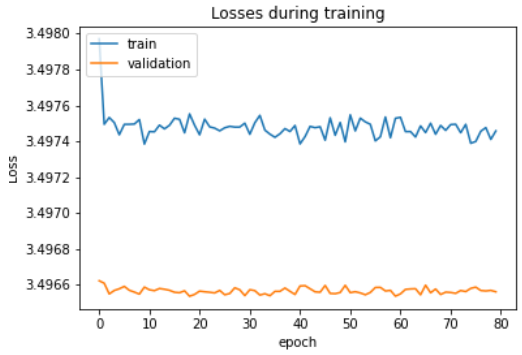

Figure 7: Loss curves

- Weight decay of: **5e-3**

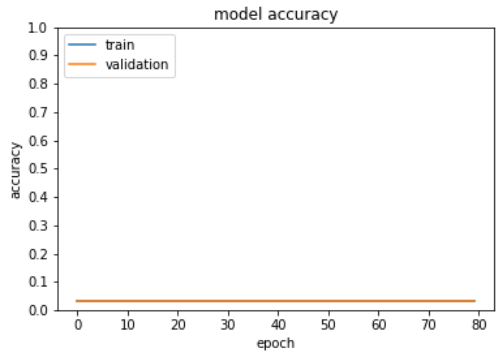

Figure 8: Accuracy curves

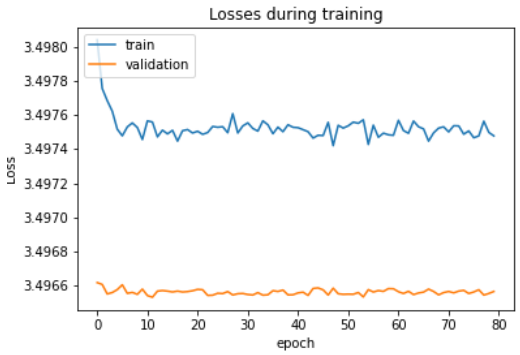

Figure 9: Loss curves

- Weight decay of: **1e-3**

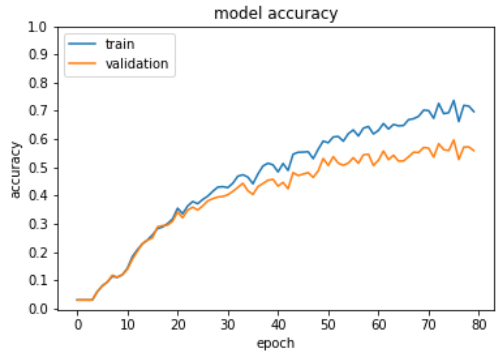

Figure 10: Accuracy curves

Figure 11: Loss curves

- Weight decay of: **5e-4**

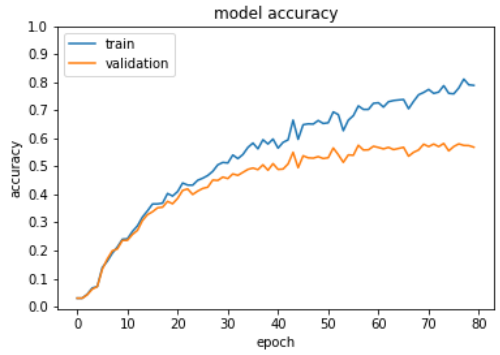

Figure 12: Accuracy curves

Figure 13: Loss curves

- Weight decay of: **1e-4**

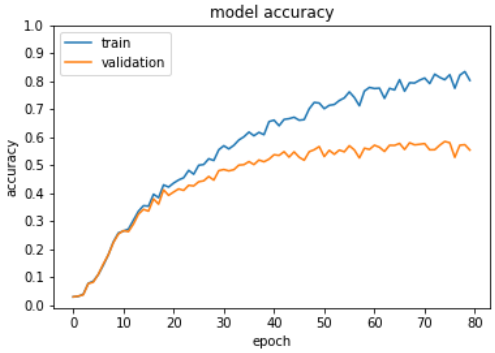

Figure 14: Accuracy curves

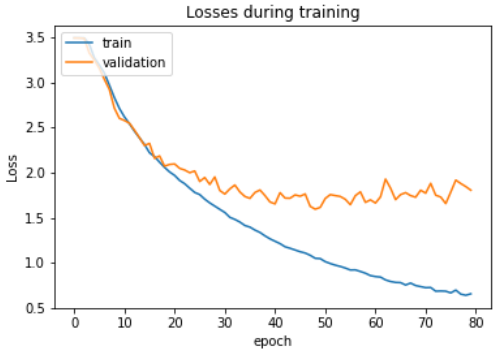

Figure 15: Loss curves

- Weight decay of: **5e-5**

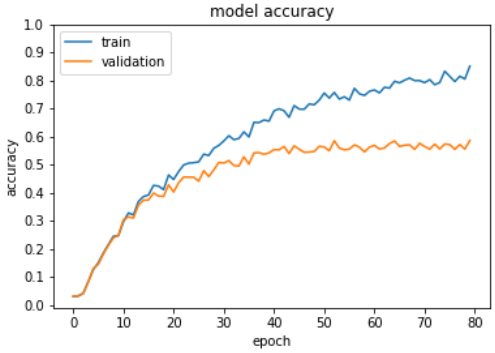

Figure 16: Accuracy curves

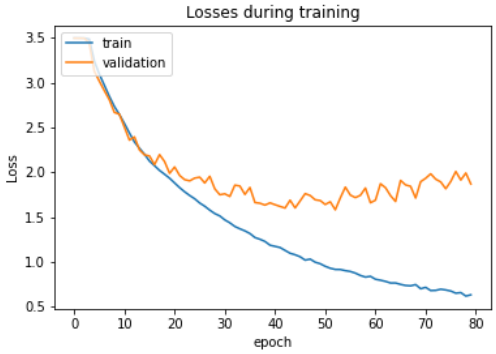

Figure 17: Loss curves

- Weight decay of: **1e-5**

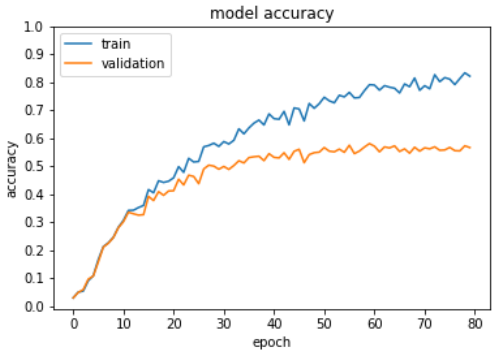

Figure 18: Accuracy curves

Figure 19: Loss curves

- Weight decay of: **5e-6**

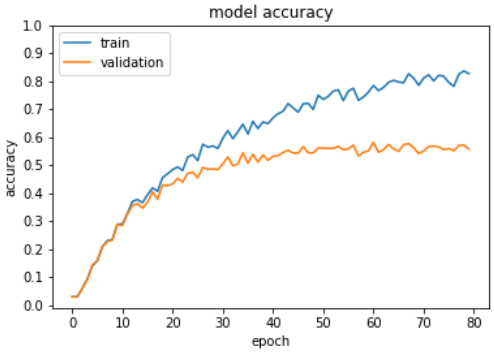

Figure 20: Accuracy curves

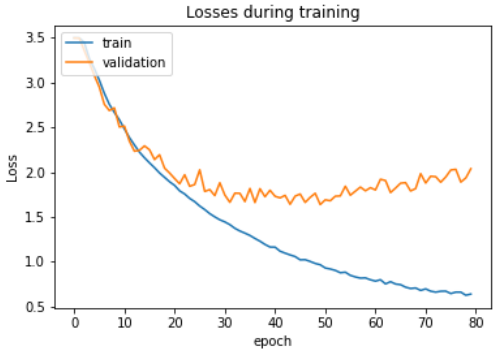

Figure 21: Loss curves

- Weight decay of: **1e-6**

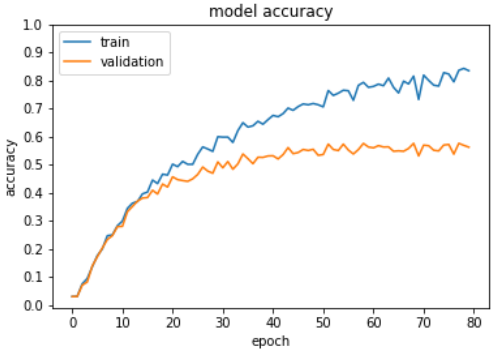

Figure 22: Accuracy curves

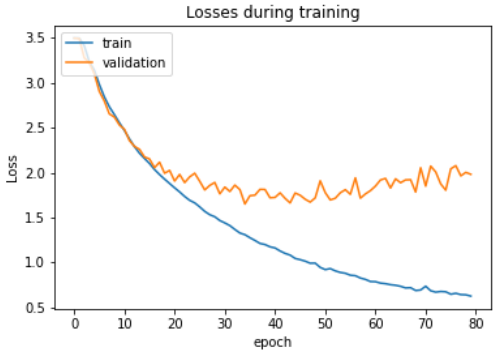

Figure 23: Loss curves

weight decay	% Training Accuracy	% Test accuracy
1e-1	3.03	3.03
5e-2	3.03	3.03
1e-2	3.03	3.03
5e-3	3.03	3.03
1e-3	73.7	59.7
5e-4	81.08	58.15
1e-4	83.39	**58.45**
5e-5	85.05	**58.58**
1e-5	83.31	58.06
5e-6	83.61	58.12
1e-6	84.26	57.61

Table 1: Final results by varying weight decay

1.0.1 Summary and Conclusions

One way for Regularization, which is a way to penalize complexity is to add all our parameters (weights) to our loss function. It usually doesn't quite work well because some parameters are positive and some are negative and even if we add the squares of all the parameters to our loss function, the loss would get so huge for large models that the best model would be to set all the parameters to 0.

To prevent this from happening, we multiply the sum of squares with another smaller number. This number is called the weight decay or *wd*. Our loss function for our network looks as follows:

$$Loss = CrossEntropyLoss(y_{pred}, y) + wd * sum(w^2) \tag{1}$$

We can understand for the results above that if we have too much weight decay, then no matter how much you train, the model will never quite fit well and converge whereas if we have too little weight decay, then we can still train well, we just need to monitor when the model starts over-fitting (ref Table 1).

This can especially be seen for the high weight decay values 1e-1, 5e-2, 1e-2, 5e-2 where the model is not training at all and the model is stuck, since the modal is being very heavily penalized and the model parameters just tend to zero. As the weight decay parameter decrease, we see its positive effects. However as it goes less than 1e-5, we see that the model tends to take more time to converge, hence some value around the range of (1e-4,5e-5) looks ideal.

2 Part-B

Since this part is a written assignment, the solutions were scanned and have been attached in Q2 folder.