ECGR 5106 – Real Time Machine Learning (Spring 2023)

Nahush D. Tambe – 801060297

https://github.com/Ntambe25

Homework # 2

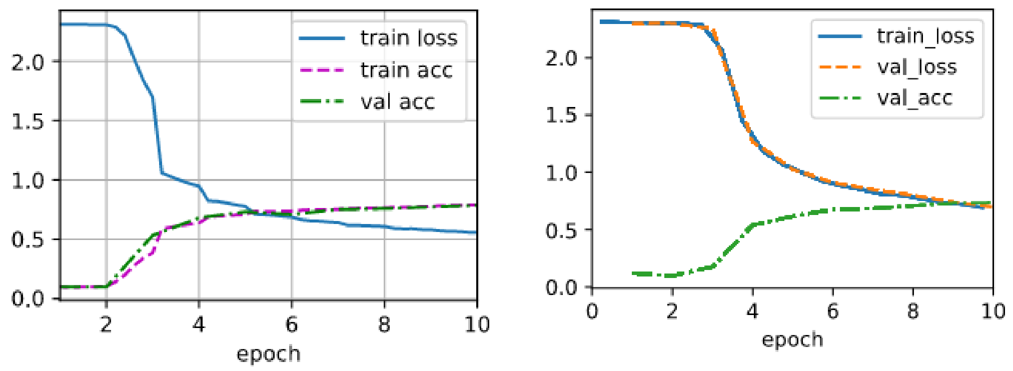


Figure 1: Results for Baseline LeNet Model

Figure 1, above, includes 2 plots with training loss, training accuracy, validation loss and validation accuracy. The reason for having 2 different plots for the baseline model as well as for all other models is the training accuracy function was used from the classic d2l book and the other plot was created using the trainer function of the regular d2l.

Problem 1:

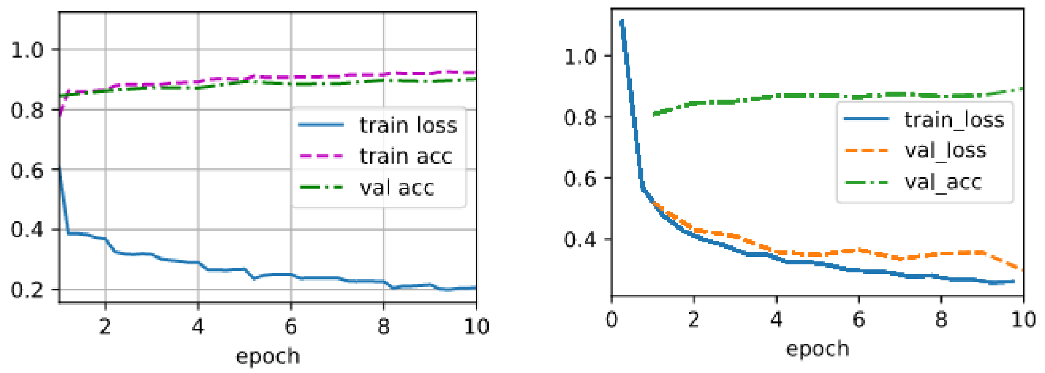


Figure 2: Results for Modernized LeNet Model

For Problem # 1, LeNet was modernized, and few changes were implemented and tested using the baseline model from d2l. First, the average pooling layer was replaced with max-pooling, and the SoftMax layer was replaced with ReLU. Figure 2, above, shows the two plots that were generated. To compare the modernized model with the baseline, it can be seen that the modernized model performs better in every aspect. The training as well as the validation accuracy is around 0.85. Training and validation loss shows a more decreasing trend than the baseline model and comes down up till 0.2, while the baseline's loss straightened out at around 0.5. The generalization gap in the modernized model is slightly bigger for both loss and accuracy.

Problem 2:

For Problem # 2, the size of the LeNet style network was changed in an attempt to improve its accuracy while keeping the max-pooling and ReLU layers.

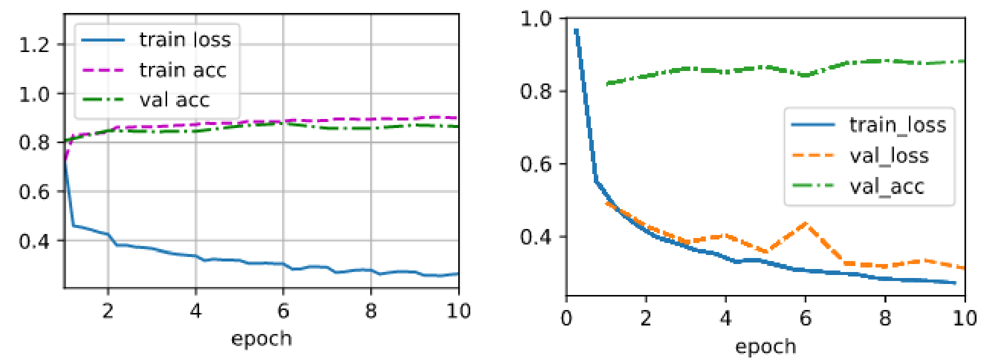
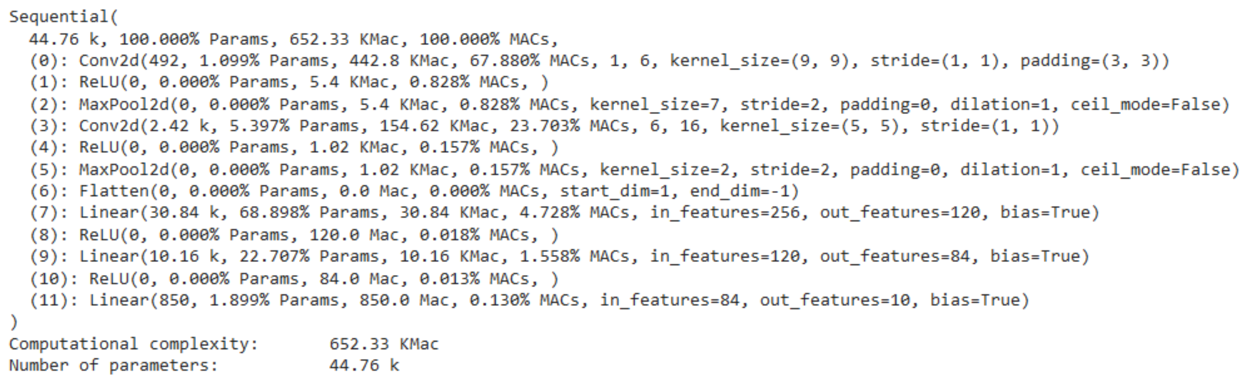


Figure 3: Results for LeNet Model with Changed Convolution Window Size

For part a, the convolution window size was experimented with. Figure 3 above shows the results for a changed convolution window size. For the 1st convolution layer, the convolution window size was changed from 5 to 7, and then to 9, and the 2nd convolution layer's window size to 7. The padding was also changed from 2 to 3 and 5 accordingly. Change in the plot was not as much visible when the window size was changed from 7 to 9, but significant change was seen in the model complexity, discussed further. This model is significantly better than the baseline model as well as the modernized model from problem 1 in all aspects. Also, point to be noted is that the generalization gap for this model is also slightly bigger than both the baseline and modernized models from problem 1.



```
Sequential(
  44.76 k, 100.000% Params, 652.33 KMac, 100.000% MACs,
  (0): Conv2d(492, 1.099% Params, 442.8 KMac, 67.880% MACs, 1, 6, kernel_size=(9, 9), stride=(1, 1), padding=(3, 3))
  (1): ReLU(0, 0.000% Params, 5.4 KMac, 0.828% MACs, )
  (2): MaxPool2d(0, 0.000% Params, 5.4 KMac, 0.828% MACs, kernel_size=7, stride=2, padding=0, dilation=1, ceil_mode=False)
  (3): Conv2d(2.42 k, 5.397% Params, 154.62 KMac, 23.703% MACs, 6, 16, kernel_size=(5, 5), stride=(1, 1))
  (4): ReLU(0, 0.000% Params, 1.02 KMac, 0.157% MACs, )
  (5): MaxPool2d(0, 0.000% Params, 1.02 KMac, 0.157% MACs, kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (6): Flatten(0, 0.000% Params, 0.0 Mac, 0.000% MACs, start_dim=1, end_dim=-1)
  (7): Linear(30.84 k, 68.898% Params, 30.84 KMac, 4.728% MACs, in_features=256, out_features=120, bias=True)
  (8): ReLU(0, 0.000% Params, 120.0 Mac, 0.018% MACs, )
  (9): Linear(10.16 k, 22.707% Params, 10.16 KMac, 1.558% MACs, in_features=120, out_features=84, bias=True)
  (10): ReLU(0, 0.000% Params, 84.0 Mac, 0.013% MACs, )
  (11): Linear(850, 1.899% Params, 850.0 Mac, 0.130% MACs, in_features=84, out_features=10, bias=True)
)
Computational complexity:       652.33 KMac
Number of parameters:           44.76 k
```

Figure 4: Model Complexity and Total # of Params. For LeNet Model with Changed Window Size

As stated earlier, the convolution window size was first changed to 7 from 5. The complexity for that model was around 20k less than this model with the window size of 9. The total number of params. decreased by around 20k. Figure 4 above shows the same.

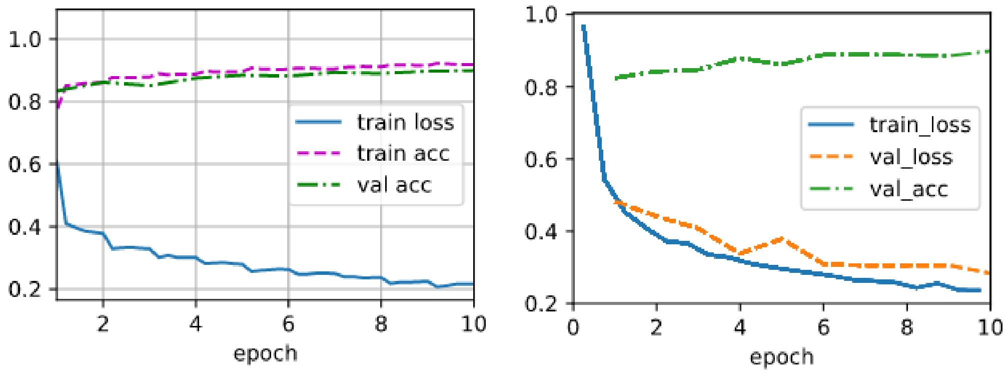


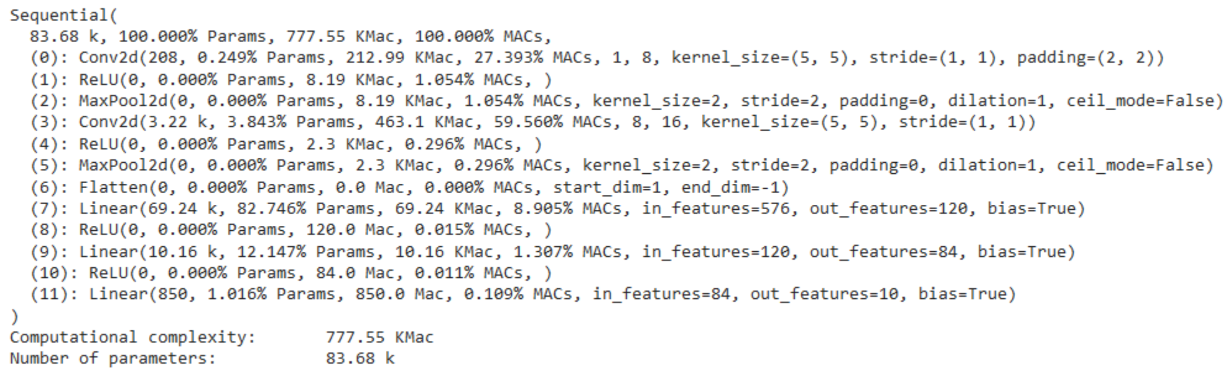
Figure 5: Results for LeNet Model with Changed Number of Output Channels

For part b, the number of output channels for only 1st convolution layer was changed from 6 to 8. Figure 5 above shows the results for a changed number of output channels. It was seen that the change in the plot was not visible when the number of output channels was increased more to numbers like 24, 32, and 64, but significant change was seen in the model complexity, discussed further. This model is significantly better than the baseline model, modernized model from problem 1 and the model in earlier part with the changed convolution window size.

To compare against the baseline and modernized model from problem 1, this model is better in every aspect.

To compare with the earlier model with changed convolution window size, the accuracy for both training and validation are around the same, but the loss for both training and validation is significantly less for this model with changed number of output channels.

Also, point to be noted is that the generalization gap for this model is also slightly smaller than the earlier model, but still slightly bigger than both the baseline and modernized models from problem 1.



```
Sequential(
  83.68 k, 100.000% Params, 777.55 KMac, 100.000% MACs,
  (0): Conv2d(208, 0.249% Params, 212.99 KMac, 27.393% MACs, 1, 8, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
  (1): ReLU(0, 0.000% Params, 8.19 KMac, 1.054% MACs, )
  (2): MaxPool2d(0, 0.000% Params, 8.19 KMac, 1.054% MACs, kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (3): Conv2d(3.22 k, 3.843% Params, 463.1 KMac, 59.560% MACs, 8, 16, kernel_size=(5, 5), stride=(1, 1))
  (4): ReLU(0, 0.000% Params, 2.3 KMac, 0.296% MACs, )
  (5): MaxPool2d(0, 0.000% Params, 2.3 KMac, 0.296% MACs, kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (6): Flatten(0, 0.000% Params, 0.0 Mac, 0.000% MACs, start_dim=1, end_dim=-1)
  (7): Linear(69.24 k, 82.746% Params, 69.24 KMac, 8.905% MACs, in_features=576, out_features=120, bias=True)
  (8): ReLU(0, 0.000% Params, 120.0 Mac, 0.015% MACs, )
  (9): Linear(10.16 k, 12.147% Params, 10.16 KMac, 1.307% MACs, in_features=120, out_features=84, bias=True)
  (10): ReLU(0, 0.000% Params, 84.0 Mac, 0.011% MACs, )
  (11): Linear(850, 1.016% Params, 850.0 Mac, 0.109% MACs, in_features=84, out_features=10, bias=True)
)
Computational complexity:       777.55 KMac
Number of parameters:           83.68 k
```

Figure 6: Model Complexity and Total # of Params. For LeNet Model with Changed Window Size

As stated earlier, the number of output channels for only 1st layer was changed to 8 from 6. For experimentation, the number of output channels was changed to gig numbers like 24, 32, and 64. The result for this was not a significant change in the loss and accuracy metrics but the complexity and the total number of parameters increased exponentially. To keep them around the same range as the earlier model with changed convolution number size, only a small change in the number of output channels was made in the 1st layer.

With this changed, the model complexity and the total number of parameters is slightly higher than the earlier model.

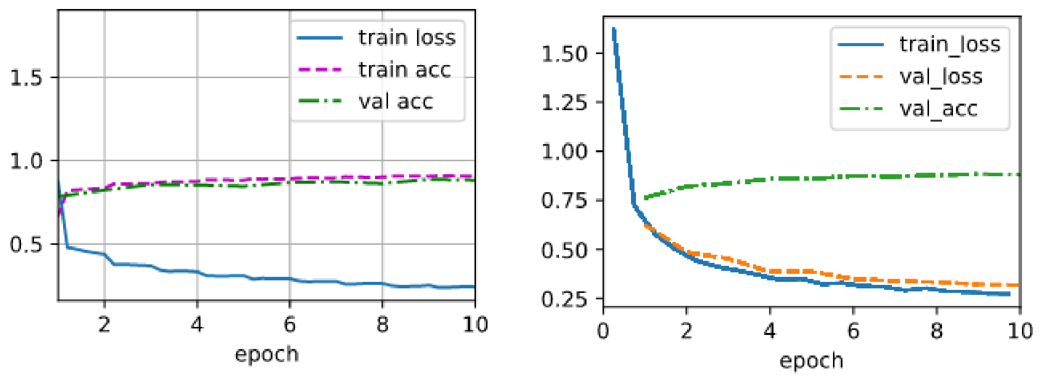
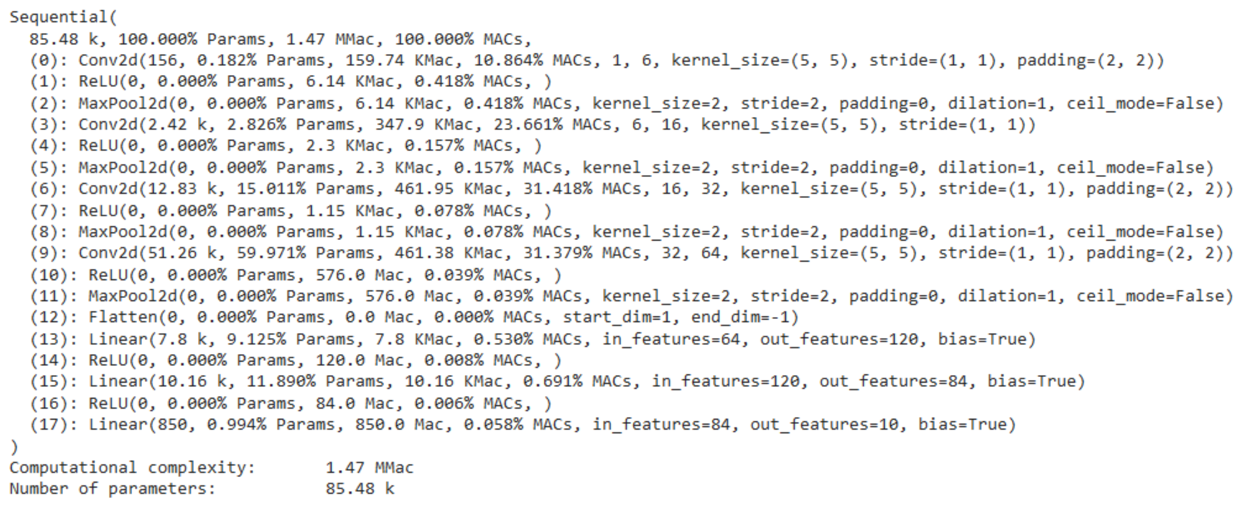


Figure 7: Results for LeNet Model with Changed Number of Convolution Layers

Por part c, the number of convolution layers was changed, and 2 more convolution layers were added. Figure 7 above shows the two plots for the changed model with training loss and accuracy as well as validation loss and accuracy. It can be safely concluded from the figure that the accuracy, for both training and validation, was around in the same range as the two previous models, but at the same time higher than the baseline and modernized model. The generalization gap is slightly less than all the 2 models discussed above.

When comparing this model against the 2 previous models, the training loss is much higher.



```
Sequential(
  85.48 k, 100.000% Params, 1.47 MMac, 100.000% MACs,
  (0): Conv2d(156, 0.182% Params, 159.74 KMac, 10.864% MACs, 1, 6, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
  (1): ReLU(0, 0.000% Params, 6.14 KMac, 0.418% MACs, )
  (2): MaxPool2d(0, 0.000% Params, 6.14 KMac, 0.418% MACs, kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (3): Conv2d(2.42 k, 2.826% Params, 347.9 KMac, 23.661% MACs, 6, 16, kernel_size=(5, 5), stride=(1, 1))
  (4): ReLU(0, 0.000% Params, 2.3 KMac, 0.157% MACs, )
  (5): MaxPool2d(0, 0.000% Params, 2.3 KMac, 0.157% MACs, kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (6): Conv2d(12.83 k, 15.011% Params, 461.95 KMac, 31.418% MACs, 16, 32, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
  (7): ReLU(0, 0.000% Params, 1.15 KMac, 0.078% MACs, )
  (8): MaxPool2d(0, 0.000% Params, 1.15 KMac, 0.078% MACs, kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (9): Conv2d(51.26 k, 59.971% Params, 461.38 KMac, 31.379% MACs, 32, 64, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
  (10): ReLU(0, 0.000% Params, 576.0 Mac, 0.039% MACs, )
  (11): MaxPool2d(0, 0.000% Params, 576.0 Mac, 0.039% MACs, kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (12): Flatten(0, 0.000% Params, 0.0 Mac, 0.000% MACs, start_dim=1, end_dim=-1)
  (13): Linear(7.8 k, 9.125% Params, 7.8 KMac, 0.530% MACs, in_features=64, out_features=120, bias=True)
  (14): ReLU(0, 0.000% Params, 120.0 Mac, 0.008% MACs, )
  (15): Linear(10.16 k, 11.890% Params, 10.16 KMac, 0.691% MACs, in_features=120, out_features=84, bias=True)
  (16): ReLU(0, 0.000% Params, 84.0 Mac, 0.006% MACs, )
  (17): Linear(850, 0.994% Params, 850.0 Mac, 0.058% MACs, in_features=84, out_features=10, bias=True)
)
Computational complexity:       1.47 MMac
Number of parameters:           85.48 k
```

Figure 8: Model Complexity and Total # of Params. For LeNet Model with Changed Number of Convolution Layers

As stated earlier, for this model, the number of convolution layers was changed, and 2 more convolution layers were added. The result of this was a rise significant rise in the model complexity and a small increase in the number of total parameters in comparison to the earlier two models with changed window size and number of output channels. Figure 8 above shows the same.

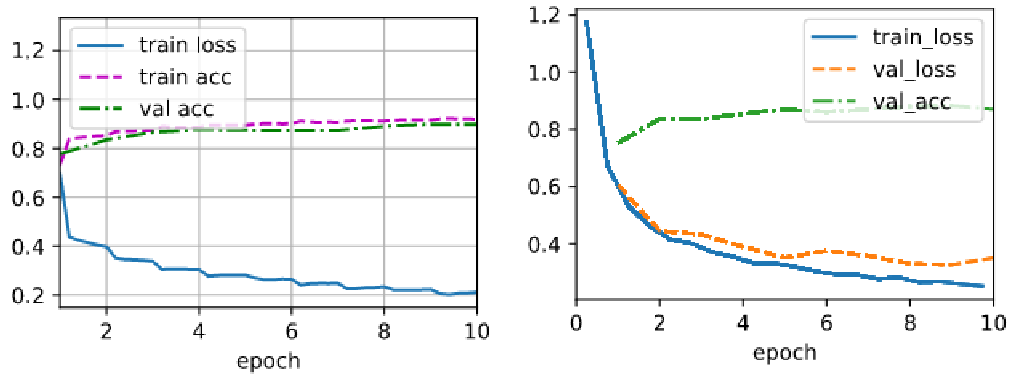
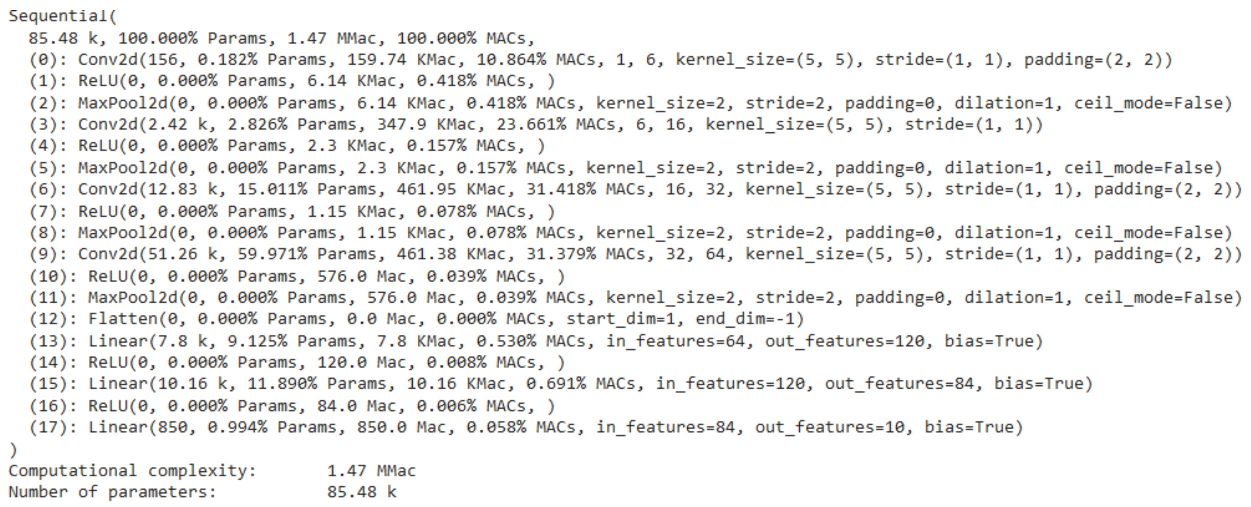


Figure 9: Results for LeNet Model with Changed Number of Fully Connected Layers

For part d, the number of fully connected layers was changed and two more fully connected layers were added. Figure 9 above shows the same. The training and validation accuracy was higher than the baseline and modernized model from problem 1, but almost same as the 3 models discussed above. The training loss is significantly smaller than the model with the changed number of convolution layers, around the same as the model with changed convolution window size, and slightly higher than the model with changed number of output channels.



```
Sequential(
  85.48 k, 100.000% Params, 1.47 MMac, 100.000% MACs,
  (0): Conv2d(156, 0.182% Params, 159.74 KMac, 10.864% MACs, 1, 6, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
  (1): ReLU(0, 0.000% Params, 6.14 KMac, 0.418% MACs, )
  (2): MaxPool2d(0, 0.000% Params, 6.14 KMac, 0.418% MACs, kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (3): Conv2d(2.42 k, 2.826% Params, 347.9 KMac, 23.661% MACs, 6, 16, kernel_size=(5, 5), stride=(1, 1))
  (4): ReLU(0, 0.000% Params, 2.3 KMac, 0.157% MACs, )
  (5): MaxPool2d(0, 0.000% Params, 2.3 KMac, 0.157% MACs, kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (6): Conv2d(12.83 k, 15.011% Params, 461.95 KMac, 31.418% MACs, 16, 32, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
  (7): ReLU(0, 0.000% Params, 1.15 KMac, 0.078% MACs, )
  (8): MaxPool2d(0, 0.000% Params, 1.15 KMac, 0.078% MACs, kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (9): Conv2d(51.26 k, 59.971% Params, 461.38 KMac, 31.379% MACs, 32, 64, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
  (10): ReLU(0, 0.000% Params, 576.0 Mac, 0.039% MACs, )
  (11): MaxPool2d(0, 0.000% Params, 576.0 Mac, 0.039% MACs, kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (12): Flatten(0, 0.000% Params, 0.0 Mac, 0.000% MACs, start_dim=1, end_dim=-1)
  (13): Linear(7.8 k, 9.125% Params, 7.8 KMac, 0.530% MACs, in_features=64, out_features=120, bias=True)
  (14): ReLU(0, 0.000% Params, 120.0 Mac, 0.008% MACs, )
  (15): Linear(10.16 k, 11.890% Params, 10.16 KMac, 0.691% MACs, in_features=120, out_features=84, bias=True)
  (16): ReLU(0, 0.000% Params, 84.0 Mac, 0.006% MACs, )
  (17): Linear(850, 0.994% Params, 850.0 Mac, 0.058% MACs, in_features=84, out_features=10, bias=True)
)
Computational complexity:       1.47 MMac
Number of parameters:           85.48 k
```

Figure 10: Model Complexity and Total # of Params. For LeNet Model with Changed Number of Fully Connected Layers

As stated earlier, for this part of the problem the number of fully connected layers was changed and two more additional fully connected layers were added. Figure 10 above shows the same. The result of this change, as expected was a significant rise in the model complexity, same as the model with changed number of convolution layers. While the model complexity increased, the total number of parameters stayed in the same range, just slightly higher than the 3 models discussed above.

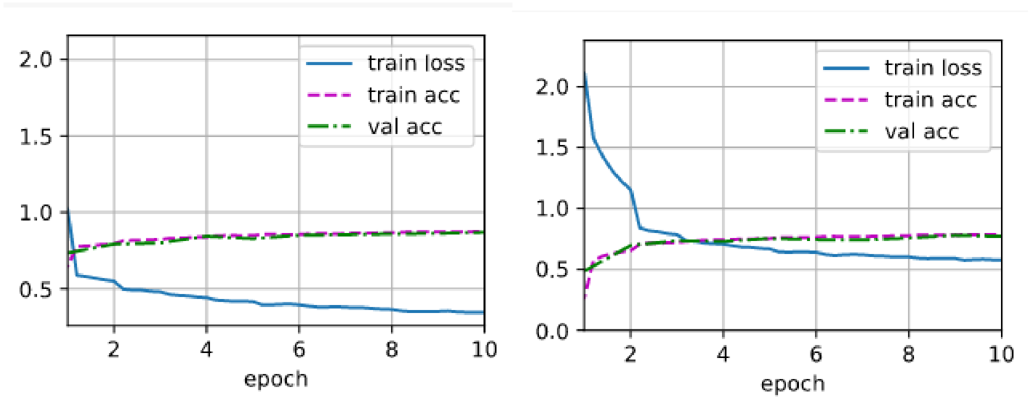
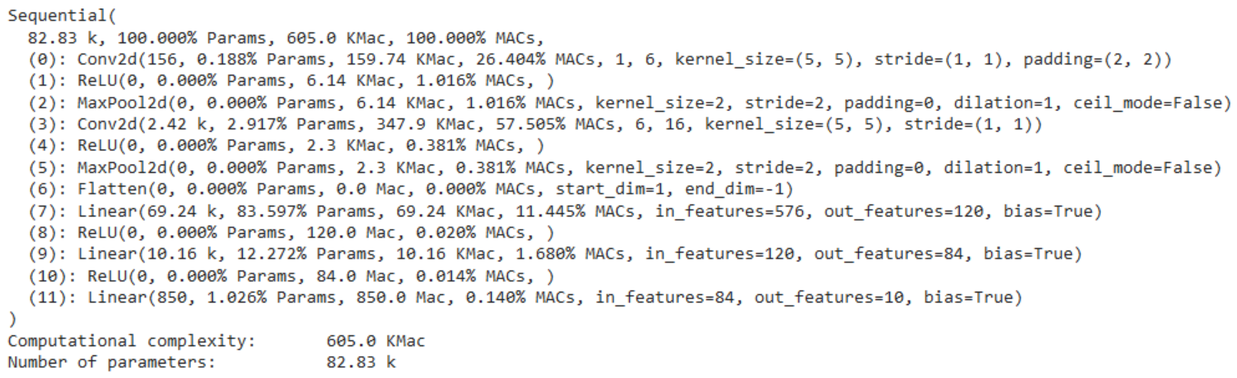


Figure 11: Results for LeNet Model with Different Learning Rates (0.01) on the Left and (0.001) on the Right

For part e, the same modernized model from problem 1 was used and different learning rates were explored. First, the learning rate of 1.0 was tried and it turned out that the result was the worst of all the experiments. Next, a learning rate of 0.01 was tried and lastly, learning rate of 0.001 was experimented with. Figure 11 above shows the results for the model with a learning rate of 0.01, on the left, and the right plot with is of the model with a learning rate off 0.001. It can be clearly seen that the learning rate of 0.01 gives the best results in terms of the training loss. The accuracy for both training and validation was definitely higher than the baseline and modernized models but at the same time, almost same as the 4 different models discussed earlier. The generalization gap is almost none for this model. The training loss is almost higher for this model, with all learning rates, than the earlier models.



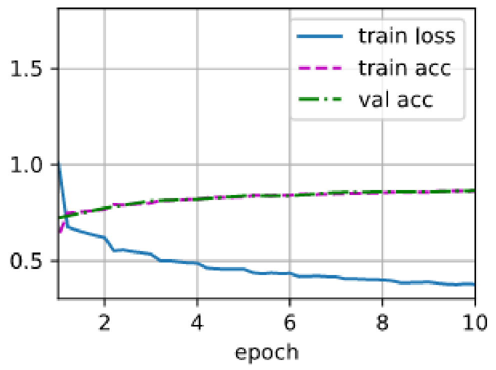
```
Sequential(
  82.83 k, 100.000% Params, 605.0 KMac, 100.000% MACs,
  (0): Conv2d(156, 0.188% Params, 159.74 KMac, 26.404% MACs, 1, 6, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
  (1): ReLU(0, 0.000% Params, 6.14 KMac, 1.016% MACs, )
  (2): MaxPool2d(0, 0.000% Params, 6.14 KMac, 1.016% MACs, kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (3): Conv2d(2.42 k, 2.917% Params, 347.9 KMac, 57.505% MACs, 6, 16, kernel_size=(5, 5), stride=(1, 1))
  (4): ReLU(0, 0.000% Params, 2.3 KMac, 0.381% MACs, )
  (5): MaxPool2d(0, 0.000% Params, 2.3 KMac, 0.381% MACs, kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (6): Flatten(0, 0.000% Params, 0.0 Mac, 0.000% MACs, start_dim=1, end_dim=-1)
  (7): Linear(69.24 k, 83.597% Params, 69.24 KMac, 11.445% MACs, in_features=576, out_features=120, bias=True)
  (8): ReLU(0, 0.000% Params, 120.0 Mac, 0.020% MACs, )
  (9): Linear(10.16 k, 12.272% Params, 10.16 KMac, 1.680% MACs, in_features=120, out_features=84, bias=True)
  (10): ReLU(0, 0.000% Params, 84.0 Mac, 0.014% MACs, )
  (11): Linear(850, 1.026% Params, 850.0 Mac, 0.140% MACs, in_features=84, out_features=10, bias=True)
)
Computational complexity:       605.0 KMac
Number of parameters:           82.83 k
```

Figure 12: Model Complexity and Total # of Params. For LeNet Model with a Learning Rate of 0.01

As stated earlier, the learning rate of 0.01 gave the best results in terms of different learning rates. Figure 12 above shows the model complexity as well as the total number of parameters for the model with a learning rate of 0.01. It can be noted that the model complexity is the lowest when compared with the 4 previous models and total number of parameters is around the same as other model.

IN CONCLUSION, the model with changed number of output channels has been the most proficient in terms of the training loss, which for the model is lower than 0.2, which is definitely lower than baseline, modernized and other 4 models with changed parameters. In terms of accuracy, this model has a higher accuracy than the baseline and modernized models, but around the same for the other 4. In terms of the model complexity, it is in the lower range with only 777.55 kMac and the total number of parameters are also in the lower range with only 83.68 k.

Problem 3:



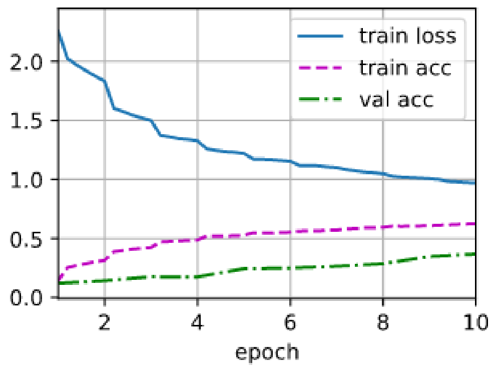


Figure 13: Results for Best Model with Dropout of (0.1) Left, and Dropout of (0.7) Right Added

As stated earlier in problem 2, the model that was most proficient among the 5 was the model with the changed number of output channels. Using that model, another additional implementation was performed for this problem. The addition was a Dropout layer that as added after all the Conv2d and LazyLinear layers. Different numbers like 0.1, 0.3, 0.5, and 0.7 for Dropout were tested with and the Dropout with 0.1 slightly improved the results for the best model from problem 2. Figure 13 above shows the amount of different that was seen with two different Dropout values.

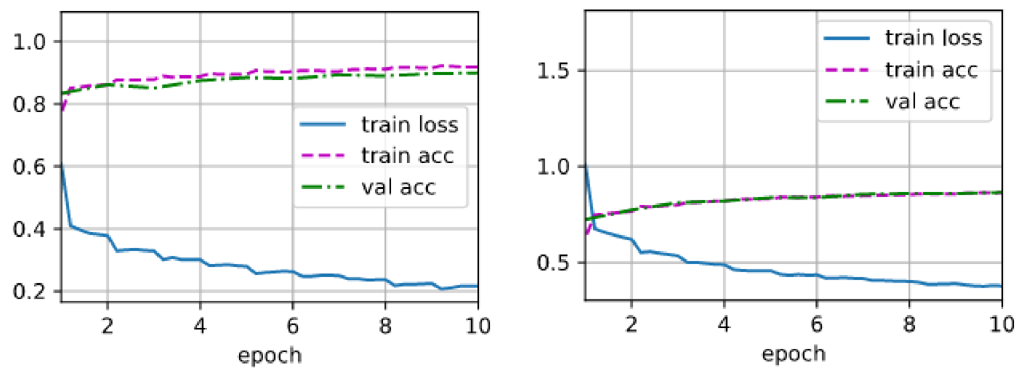


Figure 14: Best Model from Problem 2 (Left) Vs. Same Model with Dropout of 0.1 Added (Right)

Figure 14 above is a comparison between the best model from problem 2, on the left, and the same model with a Dropout of 0.1 added to it. It can clearly be seen that the training loss is showing the same trend and the lowest loss value for both the plots is around 0.2 to 0.3. The improvement is that with the addition of Dropout of 0.1, the generalization gap has almost been eliminated. This proves that the model with Dropout added is more fit for unseen data or is more generalized. In terms of the accuracy, it is around same for both the models.

Problem 4:

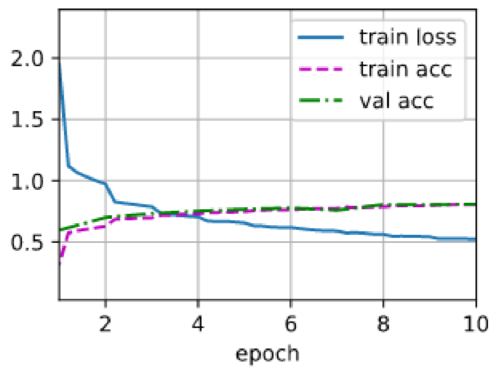


Figure 15: Results for Simplified AlexNet Model

For problem 4, the AlexNet model was used from d2l as it is without making any change and keeping its complexity same. After training it for 10 epochs, it was realized that the training time was very long. To reduce the training time, the model was simplified down by taking out one Conv2d layer and 2 linear layers. Next the kernel size was also reduced, and lastly the image size was reduced from 224x224 to 128x128. While doing so, the model was trained in between to see how the training time is changing and how the results are changing with the reduction in the model complexity.

To compare the model against the model in problem 2 and problem 3, the training loss was slightly high for this model, but the accuracy was maintained for this model. It was around the same as the two previous models. The generalization gap was seen slightly at the start but after around 3rd epoch, the generalization gap was eliminated.

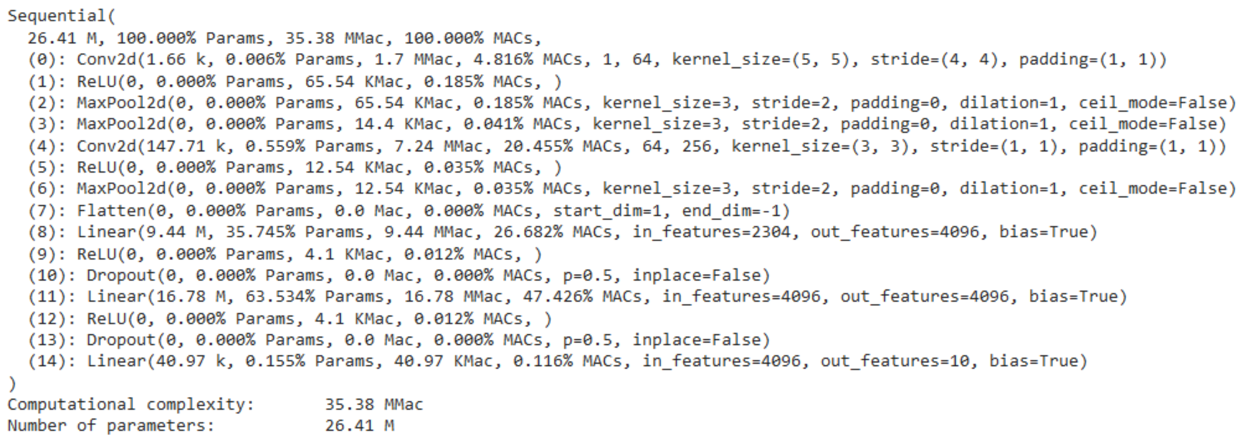

```
Sequential(
  26.41 M, 100.000% Params, 35.38 MMac, 100.000% MACs,
  (0): Conv2d(1.66 k, 0.006% Params, 1.7 MMac, 4.816% MACs, 1, 64, kernel_size=(5, 5), stride=(4, 4), padding=(1, 1))
  (1): ReLU(0, 0.000% Params, 65.54 KMac, 0.185% MACs, )
  (2): MaxPool2d(0, 0.000% Params, 65.54 KMac, 0.185% MACs, kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
  (3): MaxPool2d(0, 0.000% Params, 14.4 KMac, 0.041% MACs, kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
  (4): Conv2d(147.71 k, 0.559% Params, 7.24 MMac, 20.455% MACs, 64, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (5): ReLU(0, 0.000% Params, 12.54 KMac, 0.035% MACs, )
  (6): MaxPool2d(0, 0.000% Params, 12.54 KMac, 0.035% MACs, kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
  (7): Flatten(0, 0.000% Params, 0.0 Mac, 0.000% MACs, start_dim=1, end_dim=-1)
  (8): Linear(9.44 M, 35.745% Params, 9.44 MMac, 26.682% MACs, in_features=2304, out_features=4096, bias=True)
  (9): ReLU(0, 0.000% Params, 4.1 KMac, 0.012% MACs, )
  (10): Dropout(0, 0.000% Params, 0.0 Mac, 0.000% MACs, p=0.5, inplace=False)
  (11): Linear(16.78 M, 63.534% Params, 16.78 MMac, 47.426% MACs, in_features=4096, out_features=4096, bias=True)
  (12): ReLU(0, 0.000% Params, 4.1 KMac, 0.012% MACs, )
  (13): Dropout(0, 0.000% Params, 0.0 Mac, 0.000% MACs, p=0.5, inplace=False)
  (14): Linear(40.97 k, 0.155% Params, 40.97 KMac, 0.116% MACs, in_features=4096, out_features=10, bias=True)
)
Computational complexity:       35.38 MMac
Number of parameters:           26.41 M
```

Figure 16: Model Complexity and Total # of Params. For Simplified AlexNet Model

As stated earlier, in order to reduce the training time, the model was simplified down but it was made sure that the model's performance was not highly affected. After training the model, the model's complexity came out to be around 35 MMac and the total number of parameters were around 26.5 M.

Problem # 5:

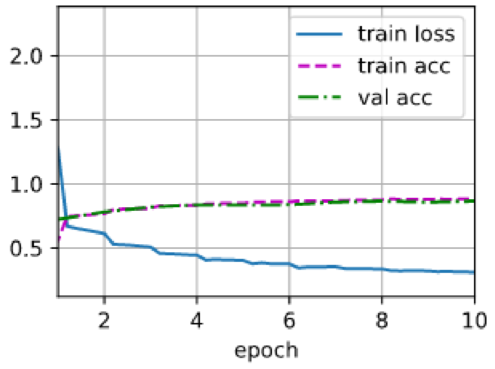
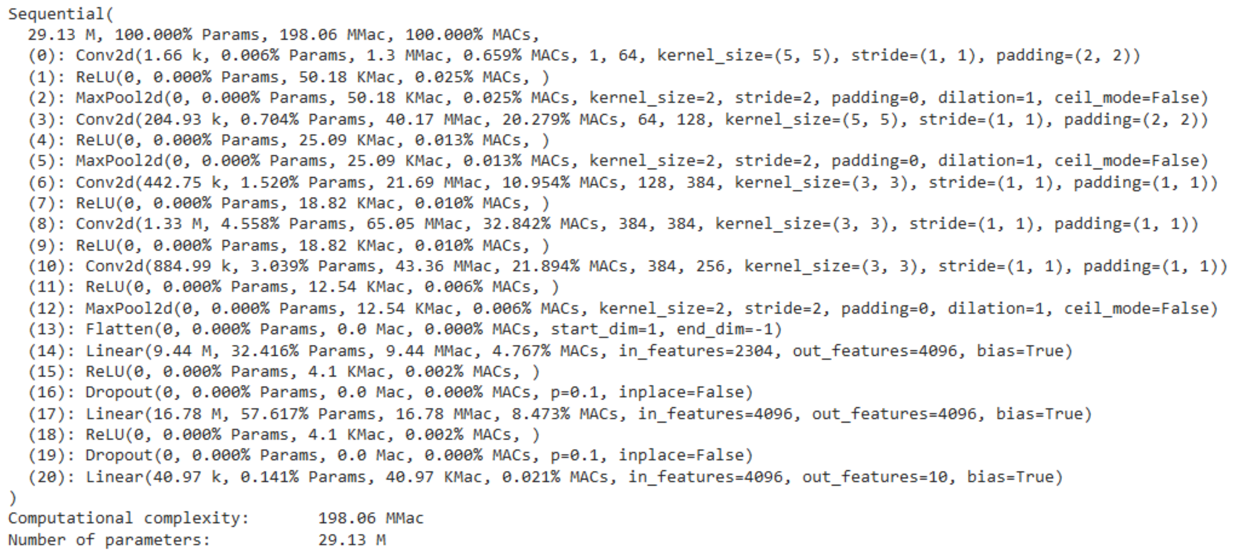


Figure 17: Results for Original AlexNet Model for 28x28 Images

For problem 5, the goal was to make changes to the AlexNet model so it can take images of size 28x28 as input. For that the original AlexNet model was edited and some changes like the kernel size, strides, and padding sizes were change to accommodate a 28x28 image. To check its initial properties like the model complexity and number of total parameters, the model was trained with the FasionMNIST dataset and the results were obtained. Figure 17 above shows the results through a plot of training loss, accuracy, and validation loss. It can be seen that for the original model the accuracy goes to around 0.8 for both training and validation. The loss goes down till 0.25.



```
Sequential(
  29.13 M, 100.000% Params, 198.06 MMac, 100.000% MACs,
  (0): Conv2d(1.66 k, 0.006% Params, 1.3 MMac, 0.659% MACs, 1, 64, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
  (1): ReLU(0, 0.000% Params, 50.18 KMac, 0.025% MACs, )
  (2): MaxPool2d(0, 0.000% Params, 50.18 KMac, 0.025% MACs, kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (3): Conv2d(204.93 k, 0.704% Params, 40.17 MMac, 20.279% MACs, 64, 128, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
  (4): ReLU(0, 0.000% Params, 25.09 KMac, 0.013% MACs, )
  (5): MaxPool2d(0, 0.000% Params, 25.09 KMac, 0.013% MACs, kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (6): Conv2d(442.75 k, 1.520% Params, 21.69 MMac, 10.954% MACs, 128, 384, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (7): ReLU(0, 0.000% Params, 18.82 KMac, 0.010% MACs, )
  (8): Conv2d(1.33 M, 4.558% Params, 65.05 MMac, 32.842% MACs, 384, 384, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (9): ReLU(0, 0.000% Params, 18.82 KMac, 0.010% MACs, )
  (10): Conv2d(884.99 k, 3.039% Params, 43.36 MMac, 21.894% MACs, 384, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (11): ReLU(0, 0.000% Params, 12.54 KMac, 0.006% MACs, )
  (12): MaxPool2d(0, 0.000% Params, 12.54 KMac, 0.006% MACs, kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (13): Flatten(0, 0.000% Params, 0.0 Mac, 0.000% MACs, start_dim=1, end_dim=-1)
  (14): Linear(9.44 M, 32.416% Params, 9.44 MMac, 4.767% MACs, in_features=2304, out_features=4096, bias=True)
  (15): ReLU(0, 0.000% Params, 4.1 KMac, 0.002% MACs, )
  (16): Dropout(0, 0.000% Params, 0.0 Mac, 0.000% MACs, p=0.1, inplace=False)
  (17): Linear(16.78 M, 57.617% Params, 16.78 MMac, 8.473% MACs, in_features=4096, out_features=4096, bias=True)
  (18): ReLU(0, 0.000% Params, 4.1 KMac, 0.002% MACs, )
  (19): Dropout(0, 0.000% Params, 0.0 Mac, 0.000% MACs, p=0.1, inplace=False)
  (20): Linear(40.97 k, 0.141% Params, 40.97 KMac, 0.021% MACs, in_features=4096, out_features=10, bias=True)
)
Computational complexity:       198.06 MMac
Number of parameters:           29.13 M
```

Figure 18: Model Complexity and Total # of Parameters for the Original AlexNet Model for 28x28 Images

As stated above, Figure 18 above shows the model complexity of the original AlexNet model designed to handle images of size 28x28. The computational complexity of the model is around

200 MMacs, which is a lost, but having large model complexity is a feature of AlexNet. The total # of parameters were also around 30 M.

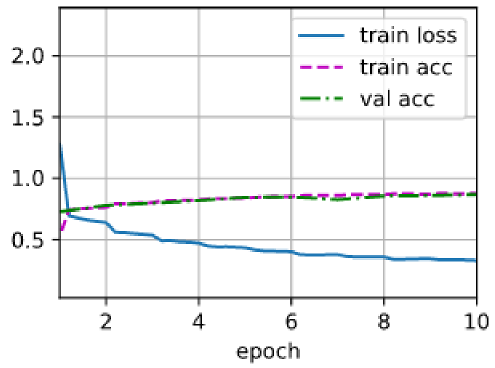
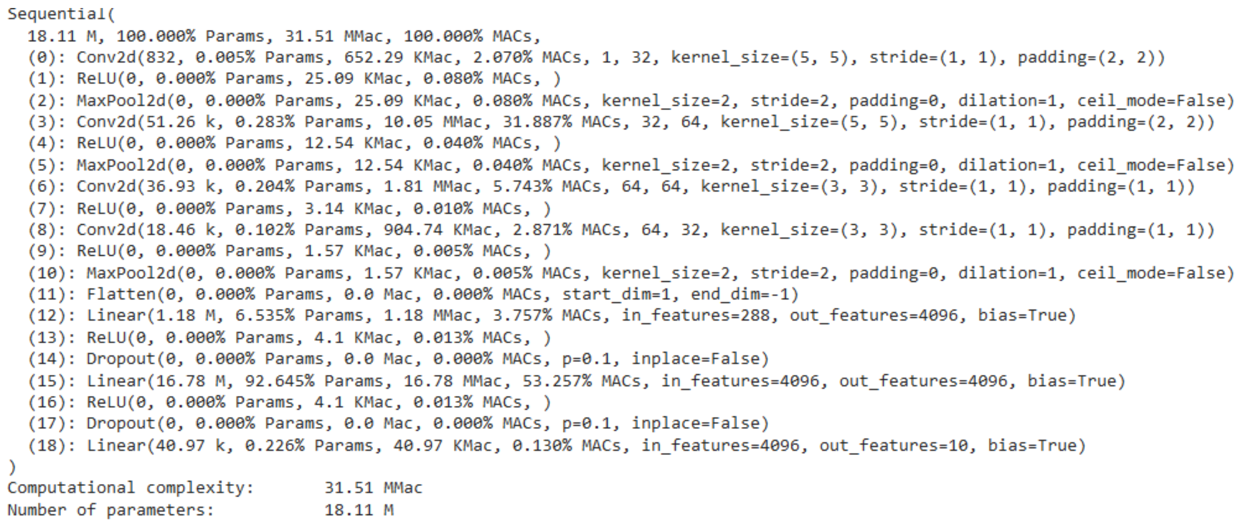


Figure 18: Results for Modified AlexNet Model for 28x28 Images with Better Results

To improve the models results like accuracy, loss, model complexity, and total # of parameters, few changed were implemented like the number of output channels was significantly lowered, and according to that the kernel_size, strides, and padding sizes were also adjusted. To further lower the model complexity and reduce the total number of parameters, one Conv2d layer was removed. The result of this change was significant reduction in model complexity and total number of parameters. Figure 18 above shows the plot for the modified AlexNet model that includes training loss, accuracy, and validation accuracy.

In comparison to the original model, the accuracy is slightly improved, the generalization gap is not present, and looks like the training loss does decrease but it is around the same as the original AlexNet model.



```
Sequential(
  18.11 M, 100.000% Params, 31.51 MMac, 100.000% MACs,
  (0): Conv2d(832, 0.005% Params, 652.29 KMac, 2.070% MACs, 1, 32, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
  (1): ReLU(0, 0.000% Params, 25.09 KMac, 0.080% MACs, )
  (2): MaxPool2d(0, 0.000% Params, 25.09 KMac, 0.080% MACs, kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (3): Conv2d(51.26 k, 0.283% Params, 10.05 MMac, 31.887% MACs, 32, 64, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
  (4): ReLU(0, 0.000% Params, 12.54 KMac, 0.040% MACs, )
  (5): MaxPool2d(0, 0.000% Params, 12.54 KMac, 0.040% MACs, kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (6): Conv2d(36.93 k, 0.204% Params, 1.81 MMac, 5.743% MACs, 64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (7): ReLU(0, 0.000% Params, 3.14 KMac, 0.010% MACs, )
  (8): Conv2d(18.46 k, 0.102% Params, 904.74 KMac, 2.871% MACs, 64, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (9): ReLU(0, 0.000% Params, 1.57 KMac, 0.005% MACs, )
  (10): MaxPool2d(0, 0.000% Params, 1.57 KMac, 0.005% MACs, kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (11): Flatten(0, 0.000% Params, 0.0 Mac, 0.000% MACs, start_dim=1, end_dim=-1)
  (12): Linear(1.18 M, 6.535% Params, 1.18 MMac, 3.757% MACs, in_features=288, out_features=4096, bias=True)
  (13): ReLU(0, 0.000% Params, 4.1 KMac, 0.013% MACs, )
  (14): Dropout(0, 0.000% Params, 0.0 Mac, 0.000% MACs, p=0.1, inplace=False)
  (15): Linear(16.78 M, 92.645% Params, 16.78 MMac, 53.257% MACs, in_features=4096, out_features=4096, bias=True)
  (16): ReLU(0, 0.000% Params, 4.1 KMac, 0.013% MACs, )
  (17): Dropout(0, 0.000% Params, 0.0 Mac, 0.000% MACs, p=0.1, inplace=False)
  (18): Linear(40.97 k, 0.226% Params, 40.97 KMac, 0.130% MACs, in_features=4096, out_features=10, bias=True)
)
Computational complexity:       31.51 MMac
Number of parameters:           18.11 M
```

Figure 19: Model Complexity and Total # of Parameters for the Modified AlexNet Model for 28x28 Images

With a goal of improving the model results, the original AlexNet model was modified, and changes discussed earlier were implemented. It can be seen that the model complexity was reduced significantly from around 200 MMacs to around 30 MMacs. This significant reduction was achieved without harming the loss and accuracy. The total number of parameters were also reduced down to around 18 M from 30 M. Figure 19 above shows the same.
