Assignment 1 Report

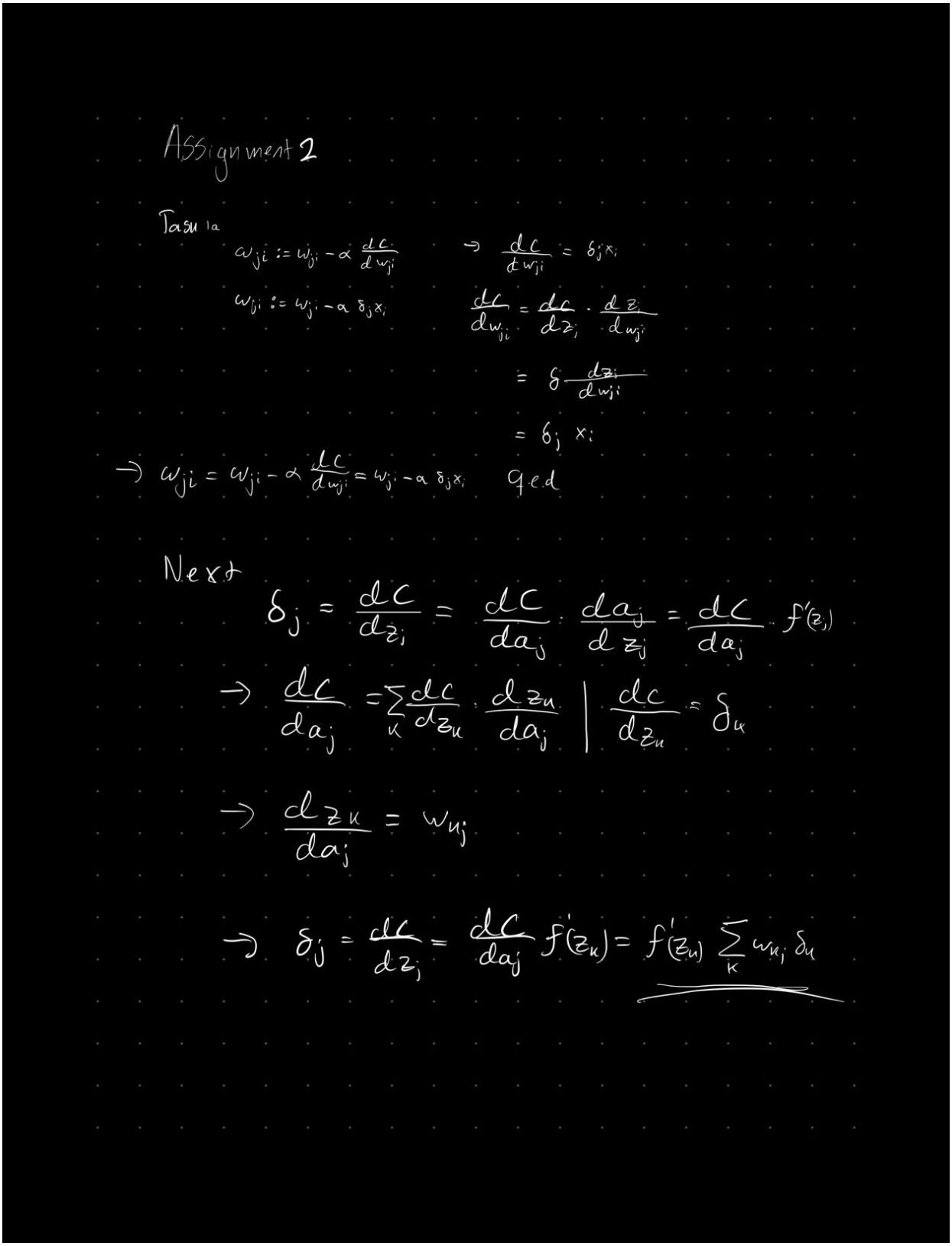
This is an outline for your report to ease the amount of work required to create your report. Jupyter notebook supports markdown, and I recommend you to check out this cheat sheet. If you are not familiar with markdown.

Before delivery, **remember to convert this file to PDF**. You can do it in two ways:

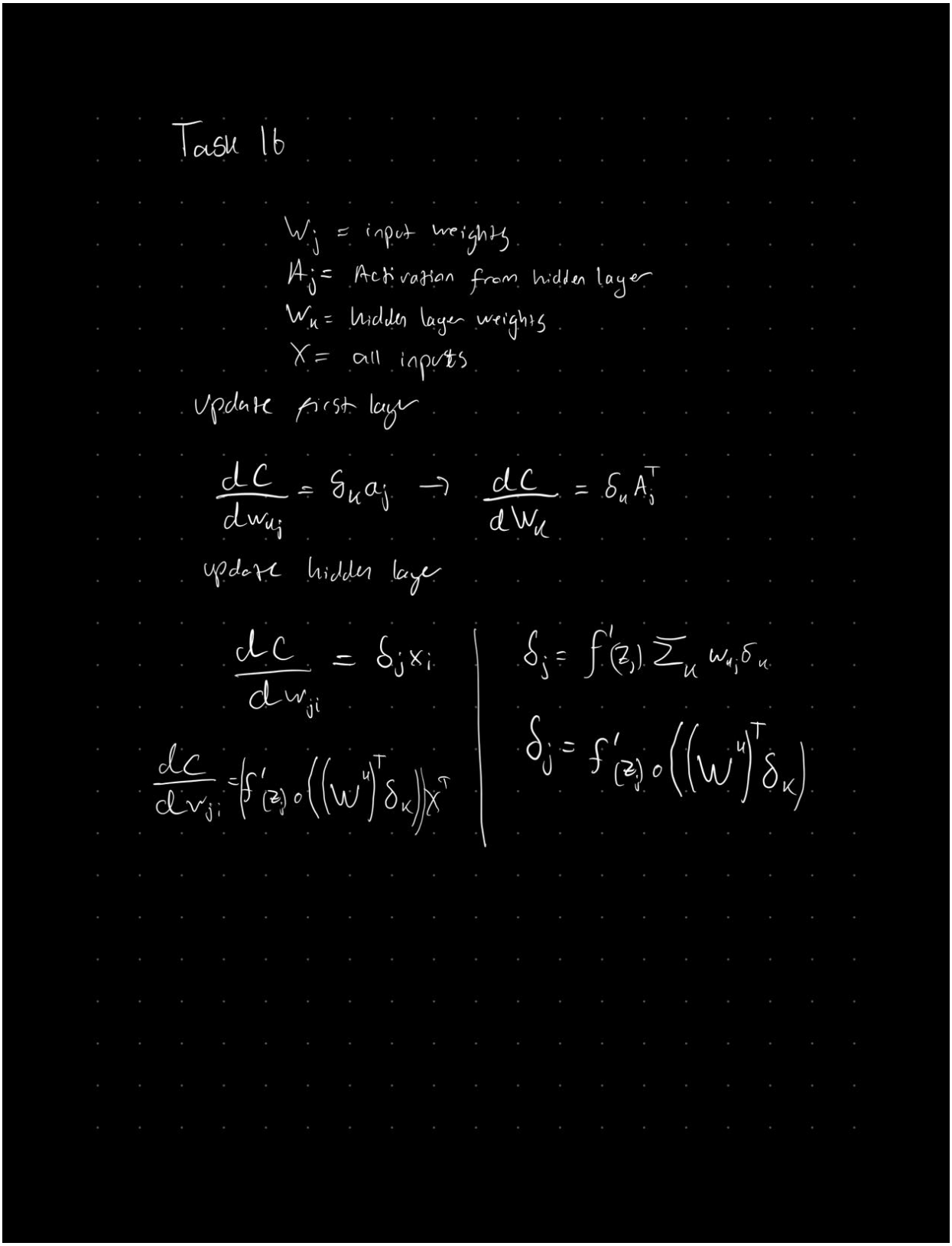
1. Print the webpage (ctrl+P or cmd+P)
2. Export with latex. This is somewhat more difficult, but you'll get somehwat of a "prettier" PDF. Go to File -> Download as -> PDF via LaTeX. You might have to install nbconvert and pandoc through conda; `conda install nbconvert pandoc` .

Task 1

task 1a)


Assignment 2
Task 1a
$w_{ji} := w_{ji} - \alpha \frac{dC}{dw_{ji}}$
$\rightarrow \frac{dC}{dw_{ji}} = \delta_j x_i$
$w_{ji} := w_{ji} - \alpha \delta_j x_i$
$\frac{dC}{dw_{ji}} = \frac{dC}{dz_j} \cdot \frac{dz_j}{dw_{ji}}$
$= \delta_j \frac{dz_j}{dw_{ji}}$
$= \delta_j x_i$
$\rightarrow w_{ji} = w_{ji} - \alpha \frac{dC}{dw_{ji}} = w_{ji} - \alpha \delta_j x_i$ q.e.d
Next
$\delta_j = \frac{dC}{dz_j} = \frac{dC}{da_j} \cdot \frac{da_j}{dz_j} = \frac{dC}{da_j} f'(z_j)$
$\rightarrow \frac{dC}{da_j} = \sum_k \frac{dC}{dz_k} \cdot \frac{dz_k}{da_j}$ | $\frac{dC}{dz_k} = \delta_k$
$\rightarrow \frac{dz_k}{da_j} = w_{kj}$
$\rightarrow \delta_j = \frac{dC}{dz_j} = \frac{dC}{da_j} f'(z_k) = f'(z_k) \sum_k w_{kj} \delta_k$

task 1b)



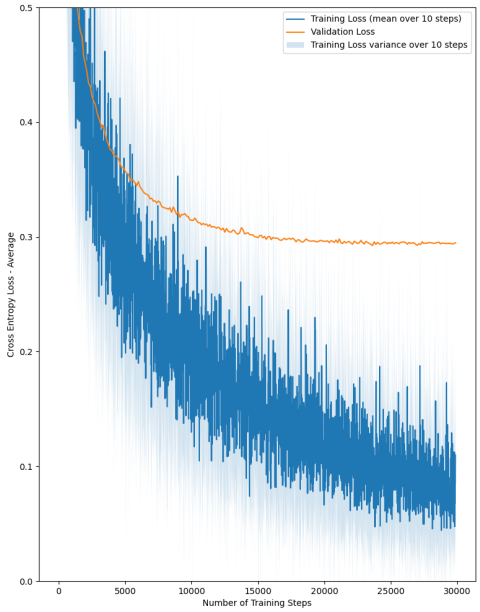
Task 2

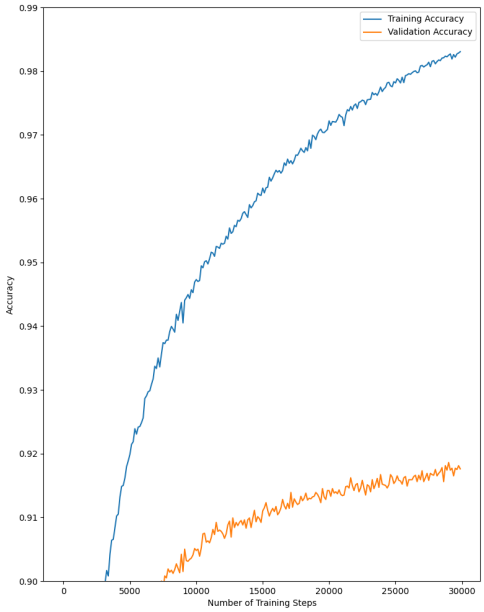
Task 2a)

mean: 33.55274553571429

standard deviation: 78.87550070784701

Task 2c)

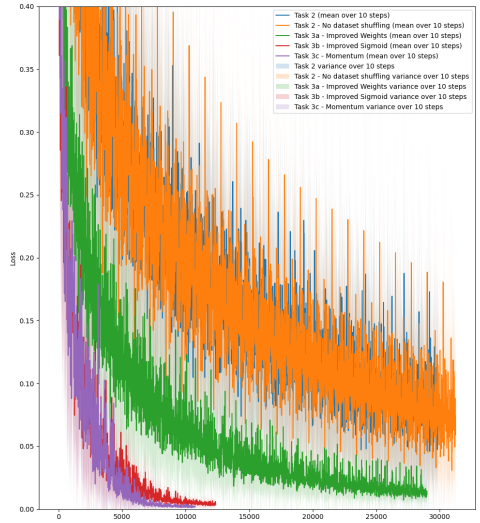


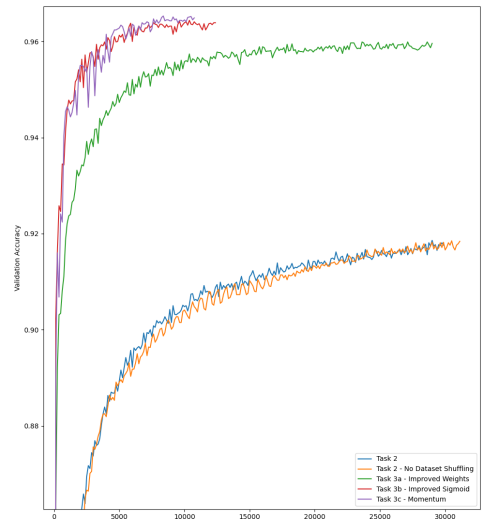


Task 2d)

(28 * 28 + 1) * 64 + 64 * 10 = 50880

Task 3



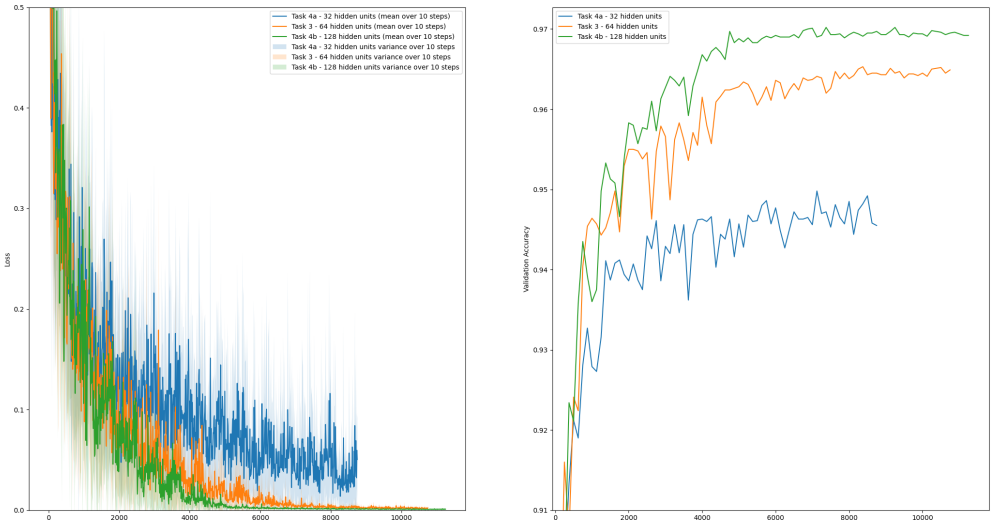


We can clearly see that the loss is improved greatly from task 2 without any enhancements. Task 3c

finishes a lot faster and has a lower loss then any other graph which shows that the improved sigmoid, improved weights and momentum really boosted the model. The same can be said for the accuracy as the momentum and improved sigmoid are almost as equal in accuracy but the momentum has fewer training steps

Task 4

Task 4a)



As we can see32 hidden units has a higer loss with more noice and a lot less accuracy. However we can also observe that is is a lot faster model to train.

Task 4b)

On the other hand 128 has almost the same loss (almost 0) but better accuracy. Unfortunetly it also requries a bit longer to train but not a drastic change. We can also see almost no noise compared to 64 when approacing minimal loss.

Task 4d)

(28 * 28 + 1) 64 + 64 * 10 = 50880

Find two hidden layer size:

(28 * 28 + 1) x + x * x + x 10 = 50880

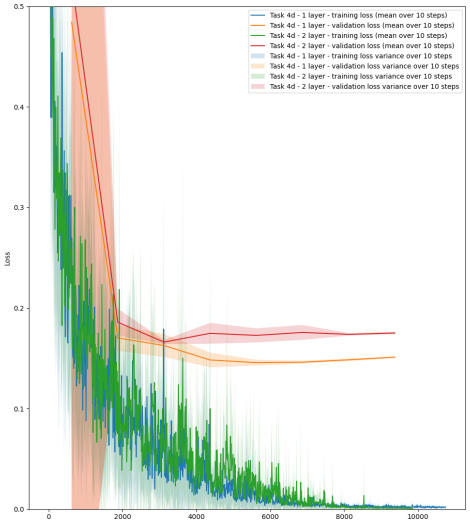
x^2 + 795x - 50880 = 0

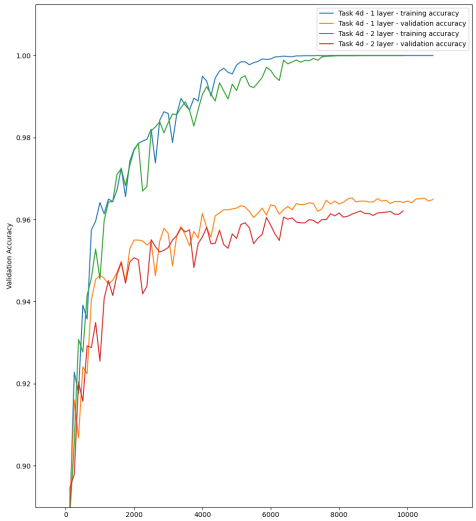
x = 59.5 ≈ 60

I end up using 60 units in each of the layers.

```
neurons_per_layer = [60,60,10]
```

(28 * 28 + 1) 60 + 60 * 60 + 60 10 = 50770



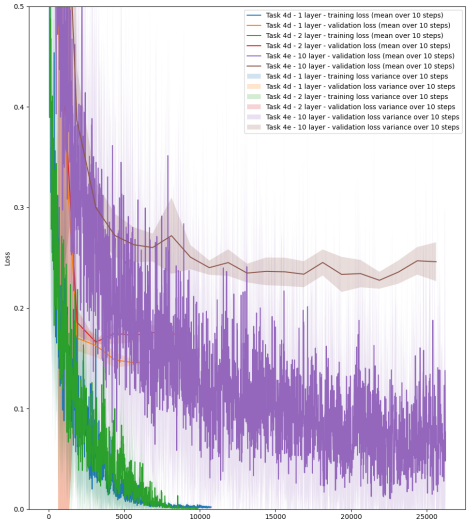


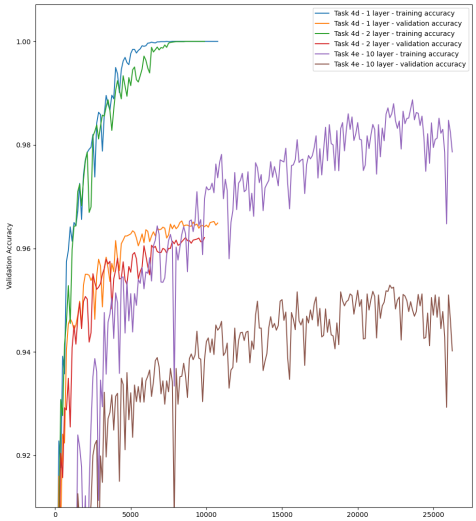
Looks like both models are similar but 2 layers is faster to train compared to 1 layer.

This might come from the fact that two layers has fewer parameters then 1 layer.

We can also see that two layers accuracy has more spikes when and performs a little worse

Task 4e)





On the other hand 10 layer model does accieve a better accuracy both takes a long time to achive this and has a lot more noise (spikes). The loss is a lot worse as well and this might come from that early stopping is not kicking in, maybe because of the amount of the small increases in loss over time.