

Thing 7

$$w_{kj} := w_{kj} - \alpha \frac{\partial C}{\partial w_{kj}} = w_{kj} - \alpha \delta_k a_j \quad (1)$$

$$\delta_k = \frac{\partial C}{\partial z_k} \approx \frac{\partial C}{\partial z_k} = -(t_k - y_k)$$

$$w_{ji} := w_{ji} - \alpha \frac{\partial C}{\partial w_{ji}} \quad (2)$$

$$\delta_j = \frac{\partial C}{\partial z_j}$$

activation from previous layer

$$\frac{\partial C}{\partial w_{ji}^L} = a_i^{L-1} \delta_j^L, \quad \delta_j^L = \frac{\partial C}{\partial z_j^L} = -(t_j - y_j)$$

$$w_{ji} = w_{ji} - \alpha a_i \delta_j, \quad a_i = x_i$$

gradient in output layer

$$\frac{\partial C}{\partial z_j} = \sum_k \frac{\partial C}{\partial z_k} \frac{\partial z_k}{\partial a_j} \frac{\partial a_j}{\partial z_j} = \delta_j$$

hidden layer activation function

$$z_j = \sum_i w_{ji} x_i$$

hidden layer activation function $f(z)$

$$\frac{\partial C}{\partial z_j} = \sum_k \frac{\partial C}{\partial z_k} \frac{\partial z_k}{\partial z_j} \Rightarrow f'(z) \sum_k \delta_k x_i = \delta_j$$

Task 1b

$$w_{kj} := w_{kj} - \alpha \delta_k a_j$$

$$= \begin{bmatrix} w_{k1} & \dots & w_{kj} \\ \vdots & & \vdots \\ w_{k,1} & \dots & w_{k,j} \end{bmatrix} - \alpha \begin{bmatrix} \delta_k \\ \vdots \\ \delta_k \end{bmatrix} \begin{bmatrix} a_1 & \dots & a_j \end{bmatrix}$$

$$w_{ji} := w_{ji} - \alpha \delta_j x_i$$

$$= \begin{bmatrix} w_{j1} & \dots & w_{ji} \\ \vdots & & \vdots \\ w_{j,1} & \dots & w_{j,i} \end{bmatrix} - \alpha \begin{bmatrix} \delta_j \\ \vdots \\ \delta_j \end{bmatrix} \begin{bmatrix} x_1 & \dots & x_j \end{bmatrix}$$

Task 3b

$$f(x) = 1,7159 \tanh\left(\frac{2}{3}x\right)$$

$$\frac{\partial f(x)}{\partial x} = 1,7159 \frac{\partial \sinh\left(\frac{2}{3}x\right)}{\partial x \cosh\left(\frac{2}{3}x\right)} =$$

$$= 1,7159 \frac{\cosh x (\cosh x - \sinh x \sinh x)}{\cosh^2 x} \cdot \left(\frac{2}{3} + \frac{2}{3}\right)$$

$$= 1,7159 \cdot \frac{4}{3} \cdot \frac{\cosh^2 x - \sinh^2 x}{\cosh^2 x}$$

$$= \frac{1}{\cosh^2 x} \cdot \frac{4}{3} \cdot 1,7159 \quad \Big|_{x=\frac{2}{3}z}$$

$$\Rightarrow \frac{\partial f}{\partial x} = \frac{4/3 \cdot 1,7159}{\cosh^2\left(\frac{2}{3}x\right)} = \frac{4/3 \cdot 1,7159}{\cosh\left(\frac{4}{3}x\right)} + 1$$

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 \(https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet\)](https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet). 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 somewhat 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)

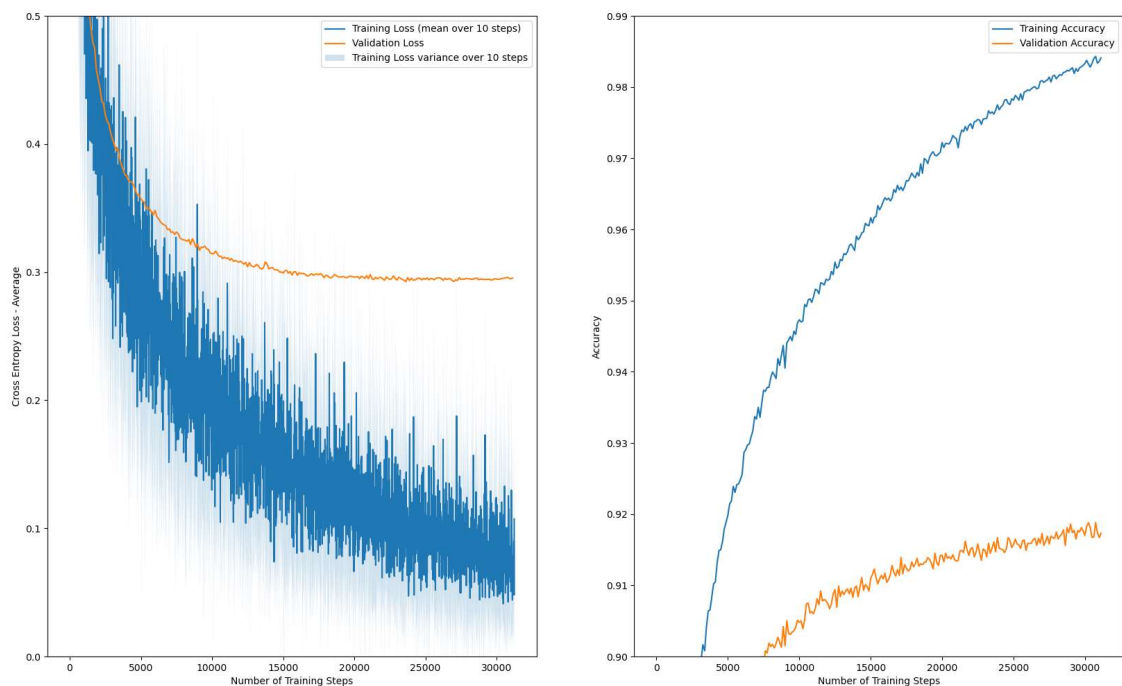
Fill in task 1a image of hand-written notes which are easy to read, or latex equations here

task 1a)

Fill in task 1a image of hand-written notes which are easy to read, or latex equations here

Task 2

Task 2c)

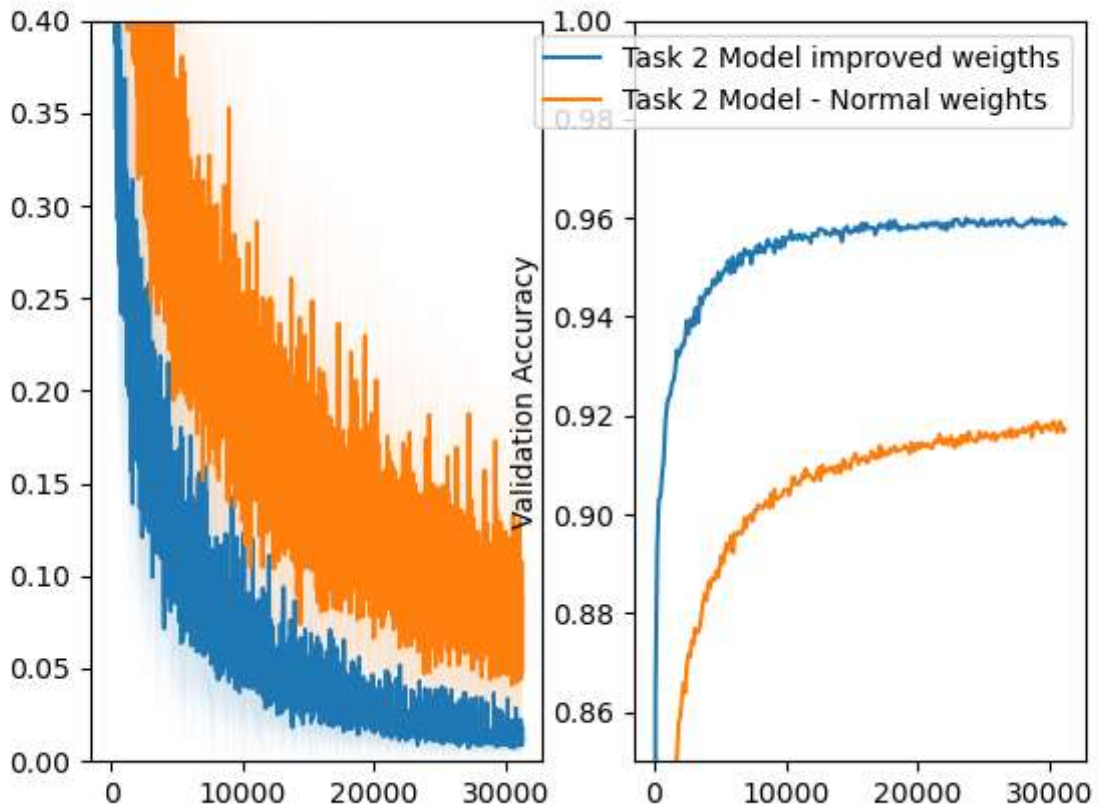


Task 2d)

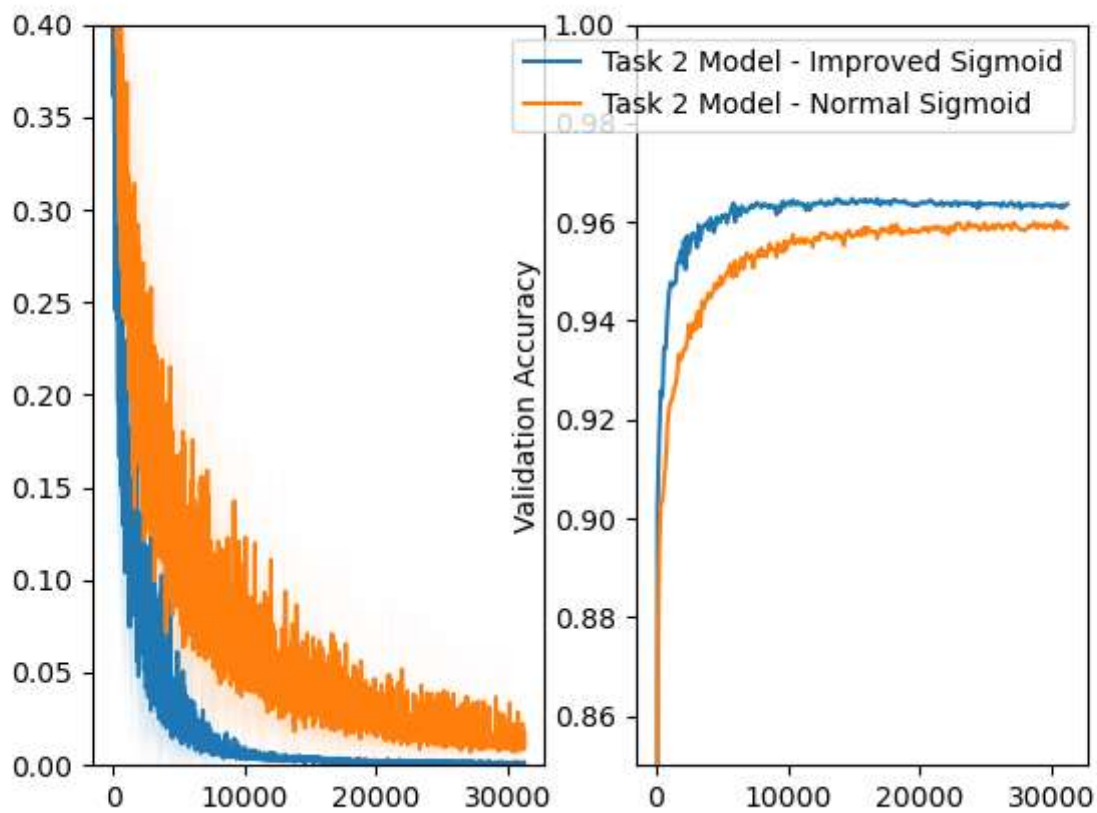
Parameters = num weights + biases = $(50240 + 640) + 2000 = 52880$

Task 3

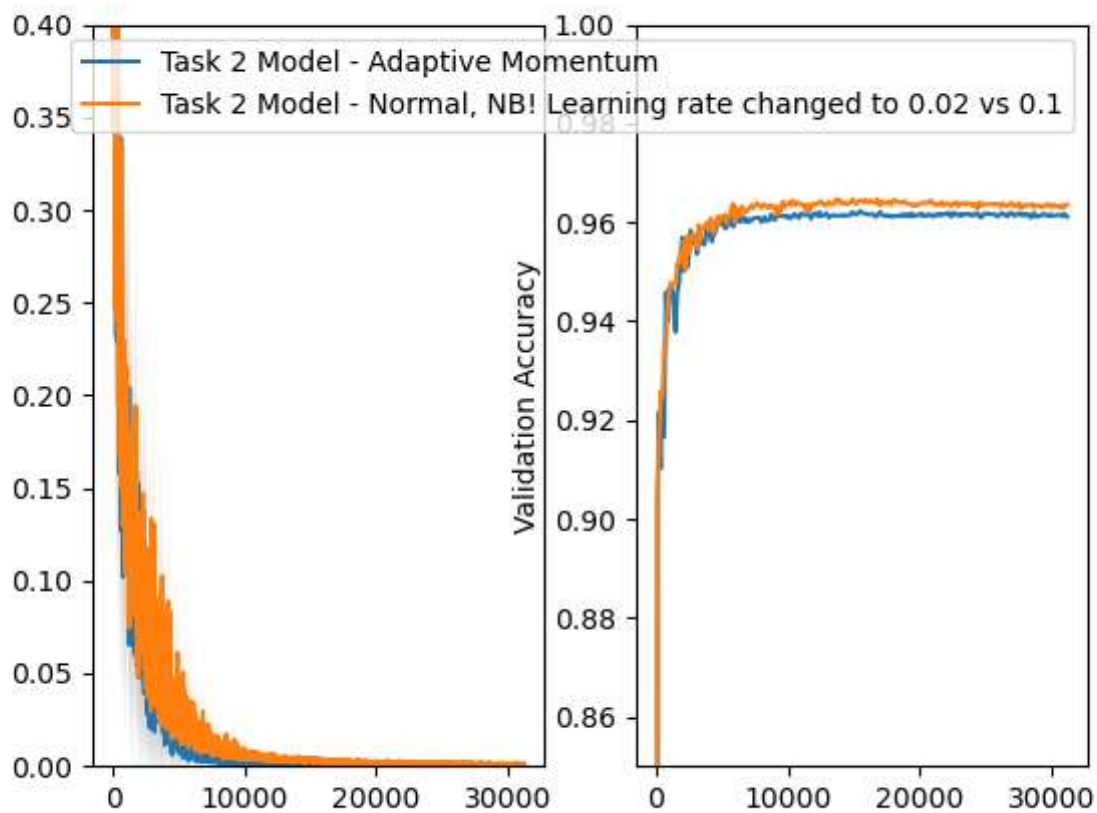
Task 3a) First improved weights. This results in improved convergence rate, and increased accuracy.



Task 3b) Both uses improved weights from task a. We can clearly se that both accuracy and loss is further improved by using the improved sigmoid funciton! Increased validation accuracy also suggest that we have less overfitting/memorizing of the learning data. Convergence rate is also further improved! Derivation of the derivative is shown in the handwritten notes!



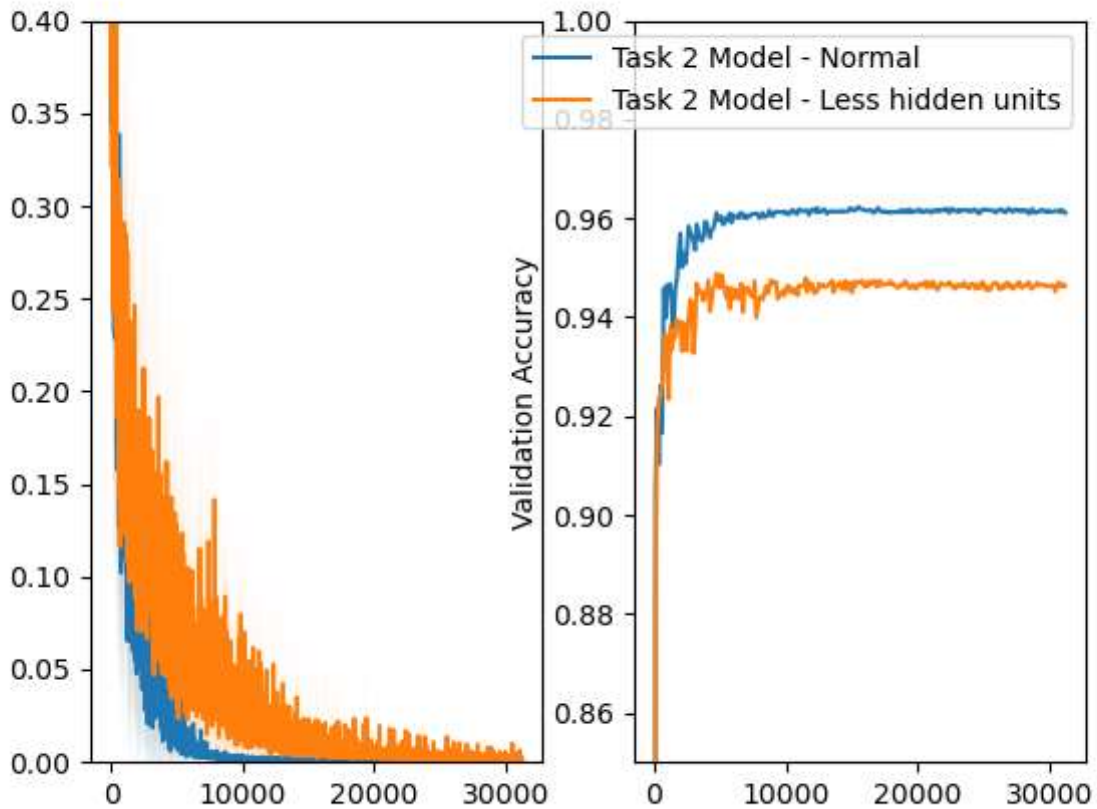
Task 3c) Both useses improvements from a and b.



Task 4

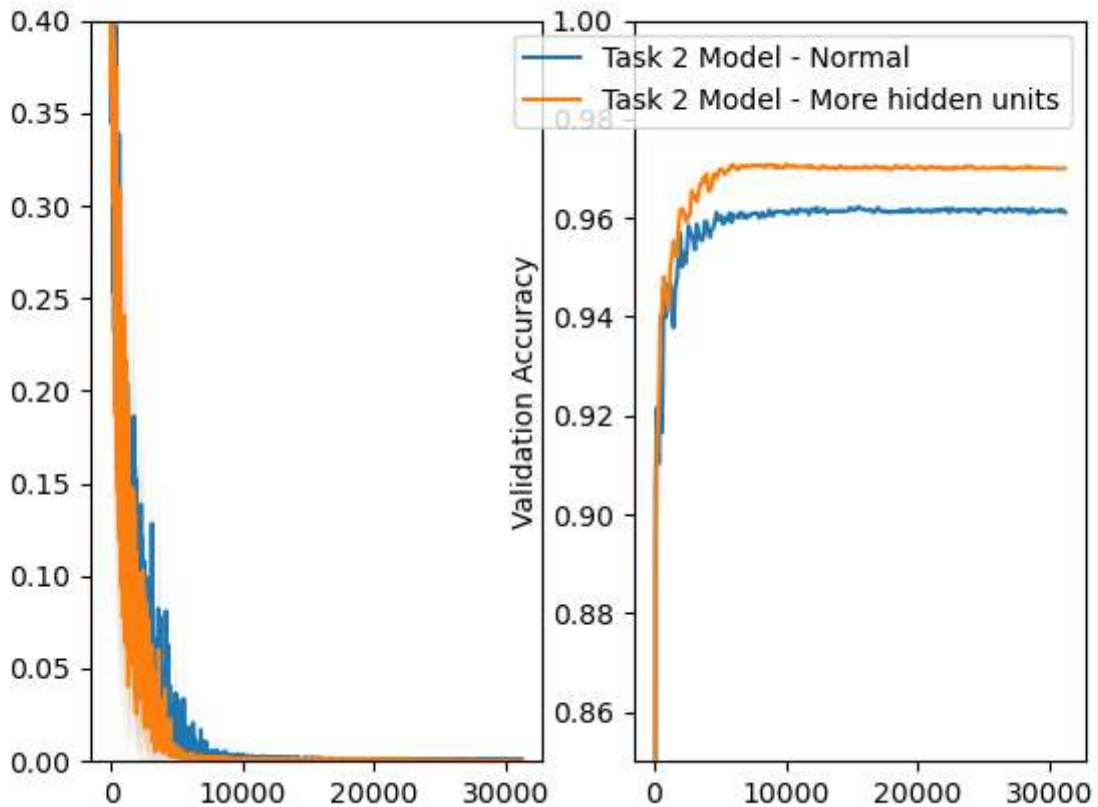
Task 4a)

The accuracy gets worse even tho the loss still converge to zero



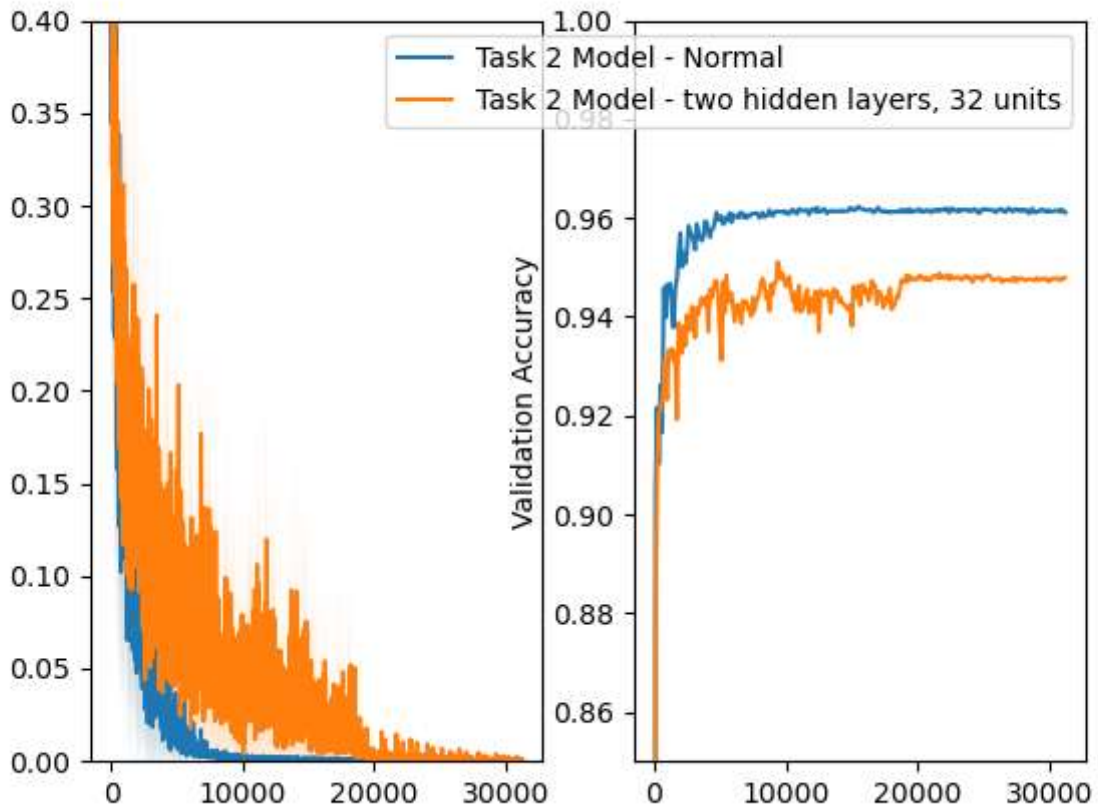
Task 4b)

The accuracy gets worse even tho the loss still converge to zero. The running time also increase.



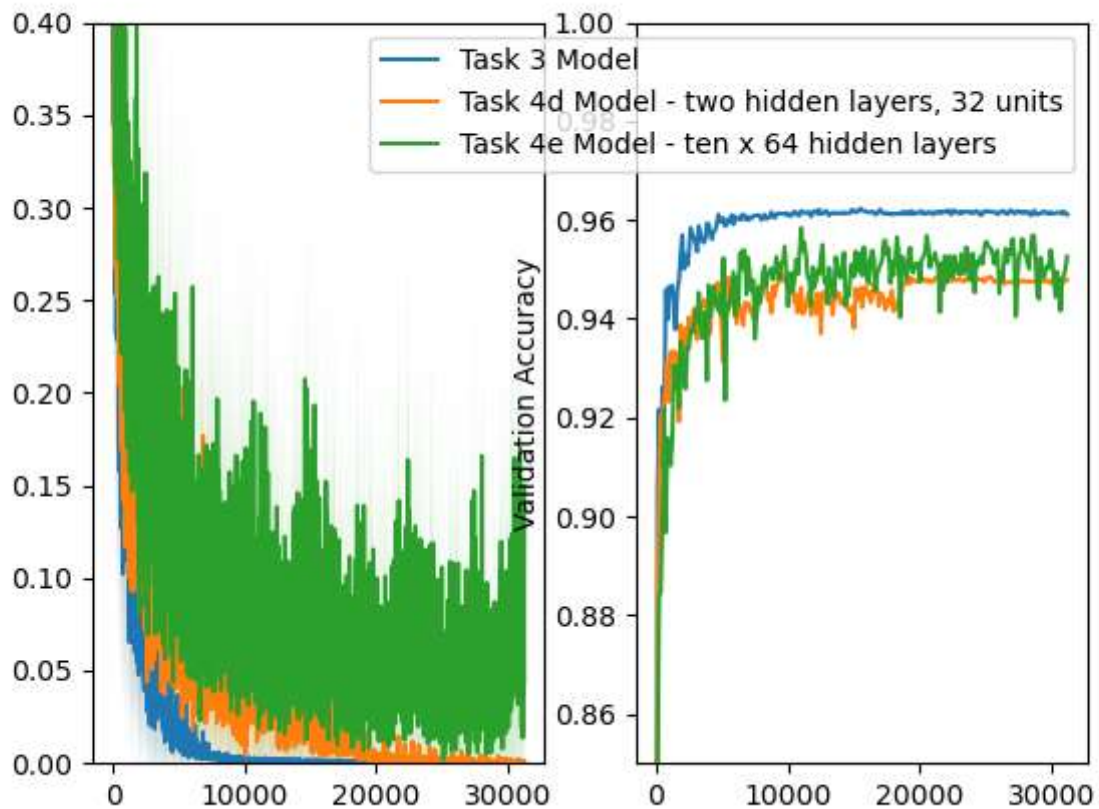
Task 4d)

This plot compares one hidden layers with 64 units to one with two hidden layers with 32 units. These should have approximately the same number of parameters. $\text{Parameters} = \text{num weights} + \text{biases} = (50240 + 640) + 2000 = 52880$. The one with two hidden layers is a lot more noisy, and simply performs more poorly. However, this could probably be fixed by having more units, and thus parameters!



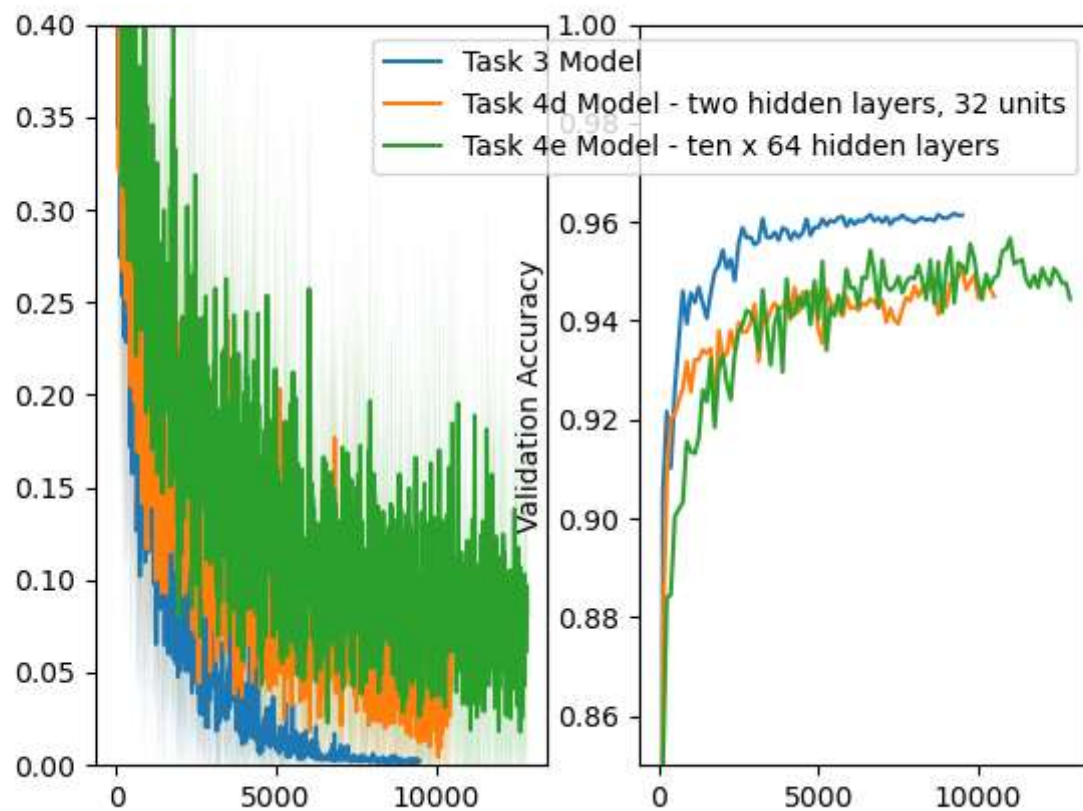
Task 4e)

Model from task 3 vs 4d vs 4e. Model from 4e has alot more paramters and the running time was really large. I should probarly include early stopping hehe. More parameters allows the model to learn training data better, however it may increase the chance for overfitting. It can be seen that model 3 with just one hidden layer outperformes all the other! And i requires alot less computing!



Early stopping

Had to do this Run time without was 342.85s Runtime with was 132.17s



In []: