Output layer

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

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

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

activation from previous layer

$$\delta_j = \frac{\partial C}{\partial z_j}, \quad \frac{\partial C}{\partial w_{ji}^l} = a_i^{l-1} \delta_j^l, \quad \delta_j^l = \frac{\partial C}{\partial z_j^l} = -(t_k - y_k)$$

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

gradient in output

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

$f(z_j)$ hidden layer activation fun

hidden layer activation function

$$z_j = \sum w_{ij} x_i$$

$f'(z)$

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

Task 1b

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

$$= \begin{bmatrix} w_{11} & \cdots & w_{1j} \\ \vdots & \ddots & \vdots \\ w_{k1} & \cdots & w_{kj} \end{bmatrix} - \alpha \begin{bmatrix} \delta_1 \\ \vdots \\ \delta_k \end{bmatrix} \begin{bmatrix} a_1 & \cdots & a_j \end{bmatrix}$$

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

$$= \begin{bmatrix} w_{11} & \cdots & w_{1i} \\ \vdots & \ddots & \vdots \\ w_{j1} & \cdots & w_{ji} \end{bmatrix} - \alpha \begin{bmatrix} \delta_1 \\ \vdots \\ \delta_j \end{bmatrix} \begin{bmatrix} x_1 & \cdots & x_j \end{bmatrix}$$

Task 36

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

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

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

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

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

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

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). 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)

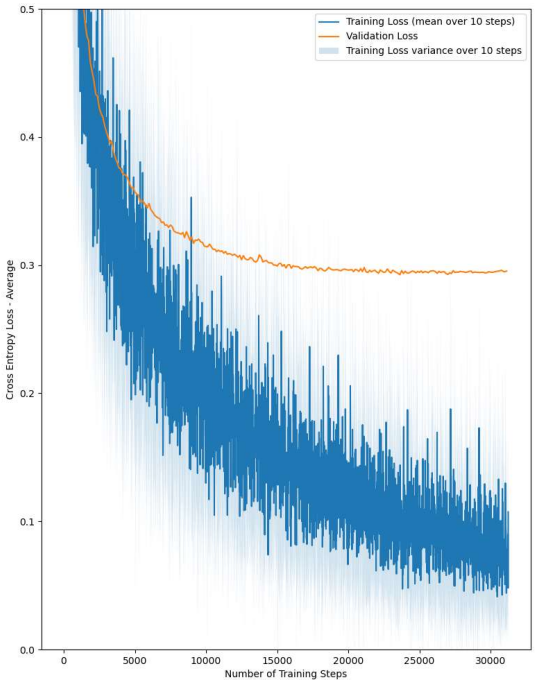
Fill in task 1a image of hand-written notes which are easy to read, or latex equations here

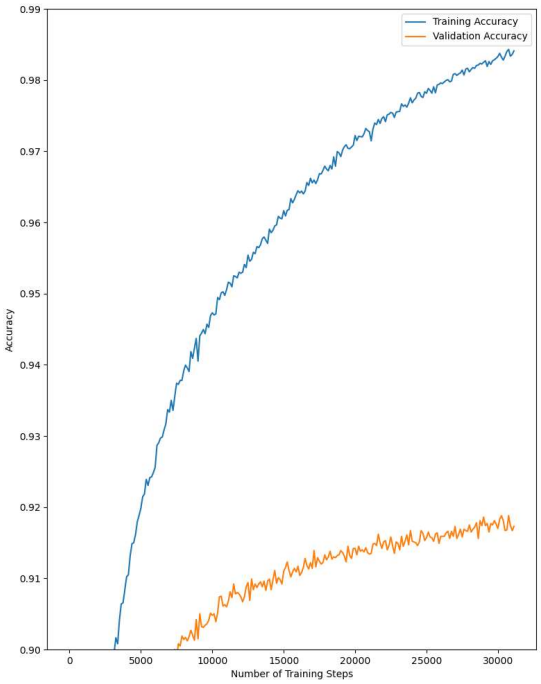
task 1a)

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

Task 2

Task 2c)



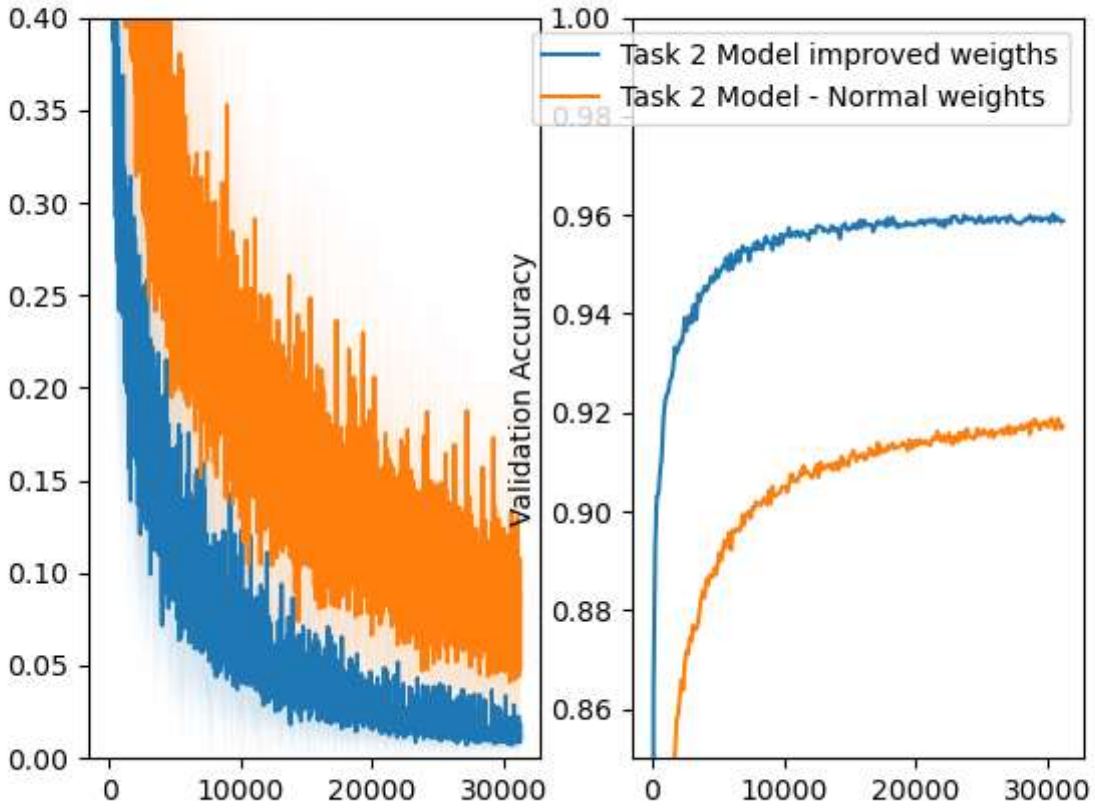


Task 2d)

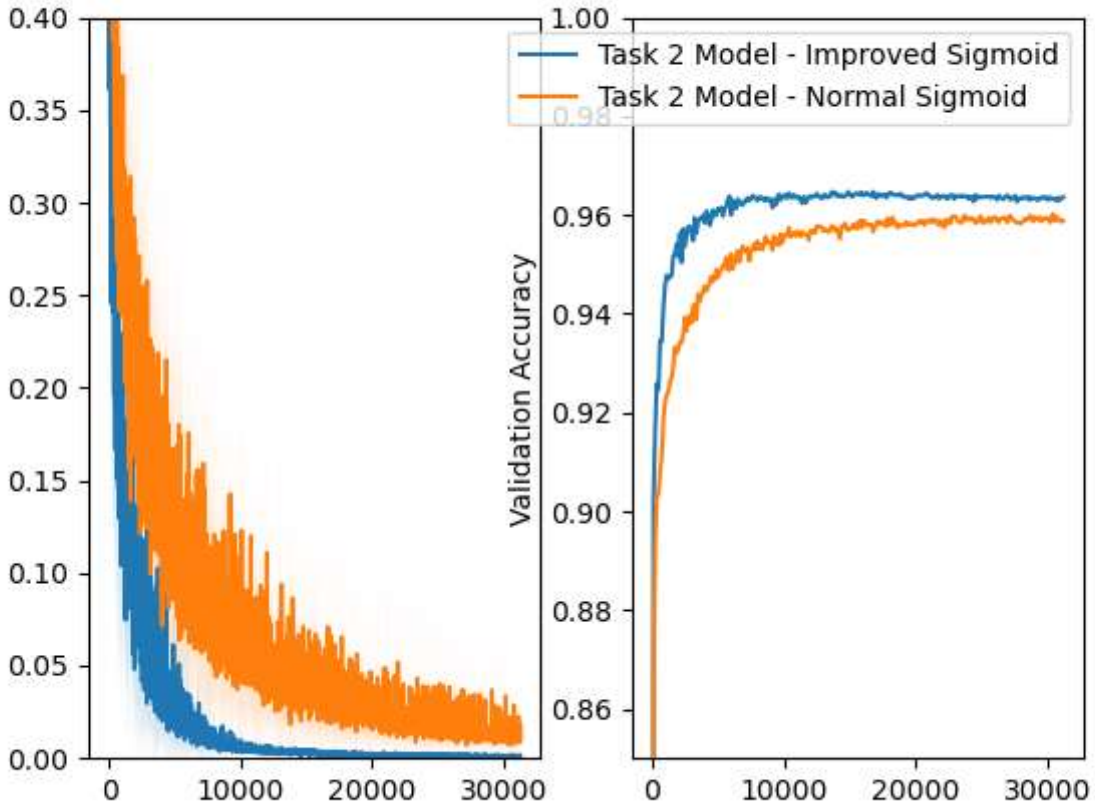
Parameters = num weights + biases = (50240 + 640) = 50880 Inspected from debugger

Task 3

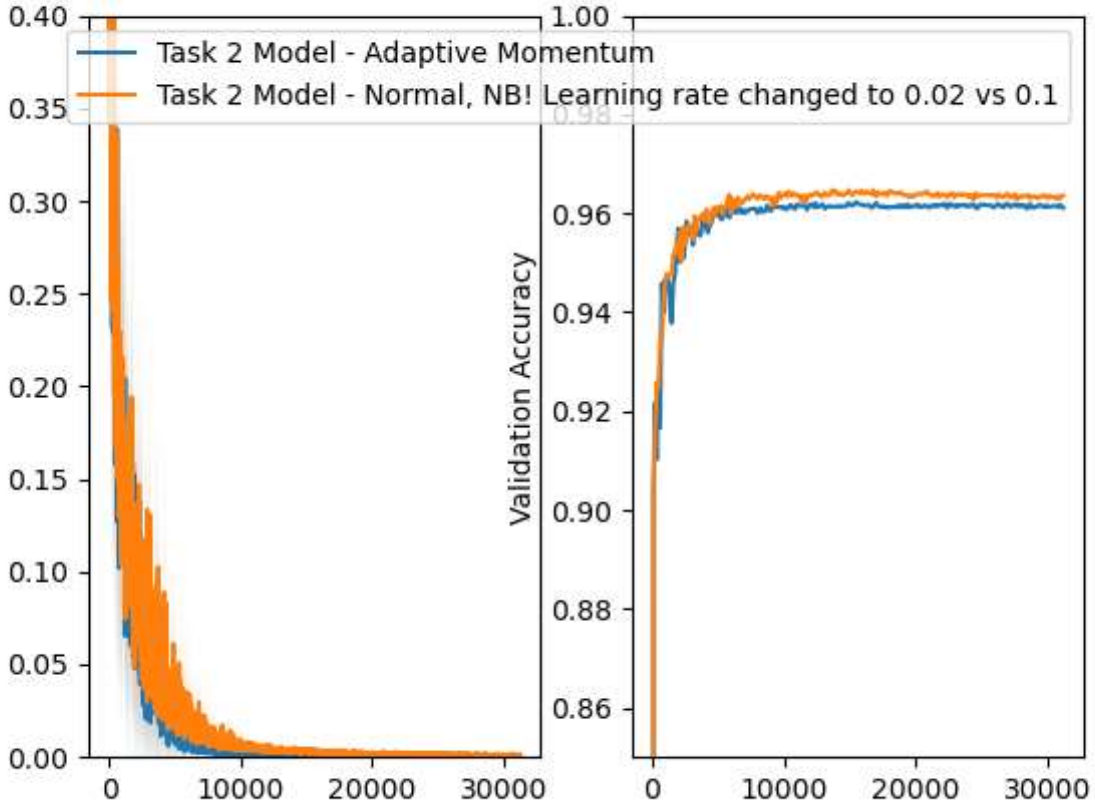
Taask 3a) First improved weights. This results in inmproved convergance rate, and increased accuracy.



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 valdidation accuracy also suggest that we have less overfitting/memorizing of the learning data. Convergance rate is also further improved! Derivation of the derivative is shown in the handwritten notes!



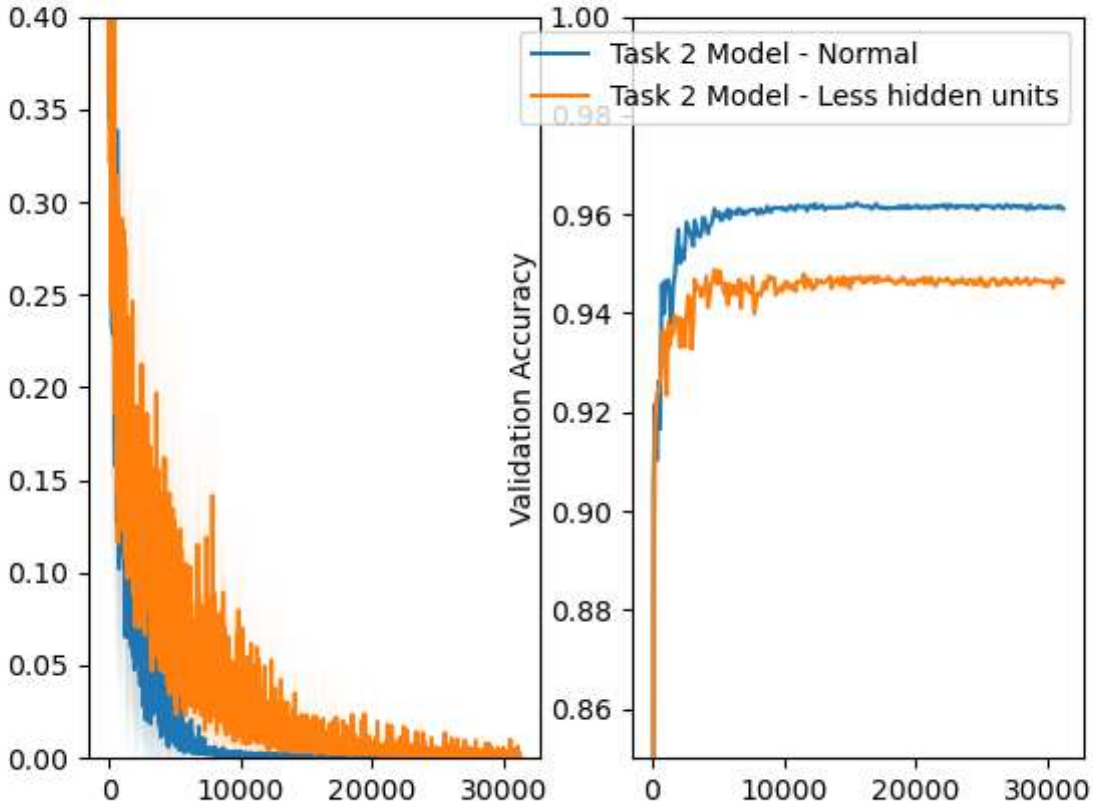
Task 3c) Both useses improvements from a and b.



Task 4

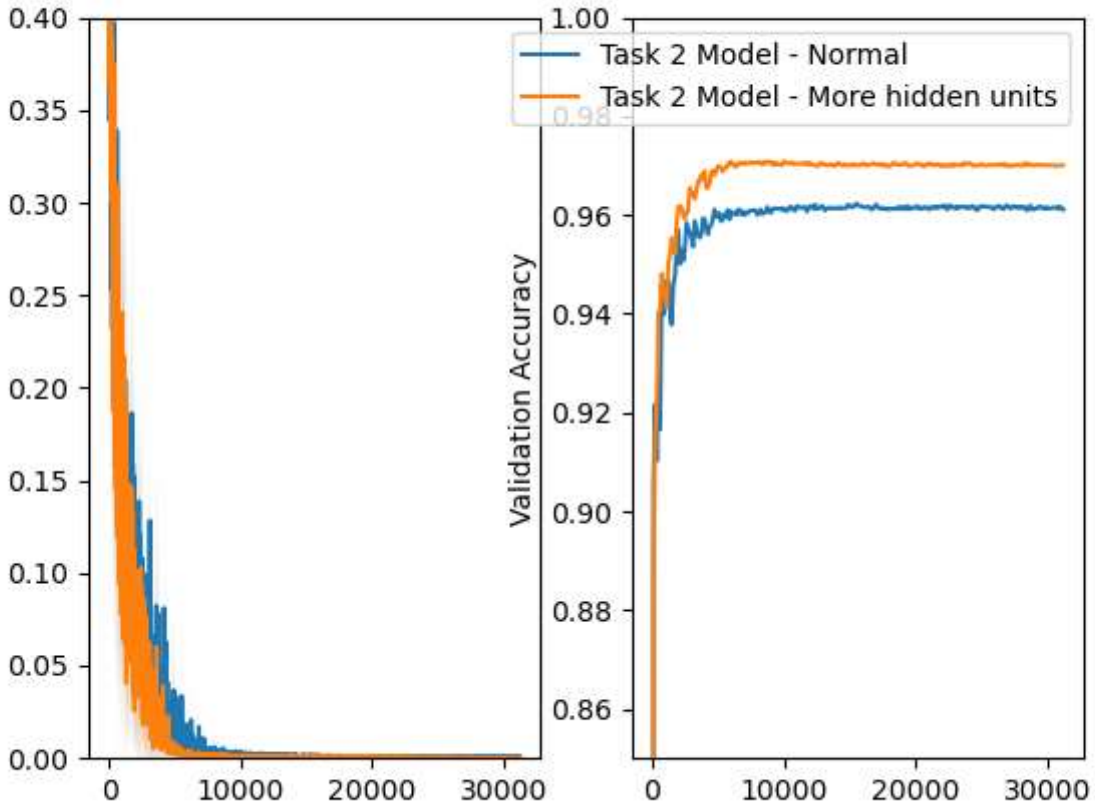
Task 4a)

The acurracy gets worse even tho the loss stil convergece to zero



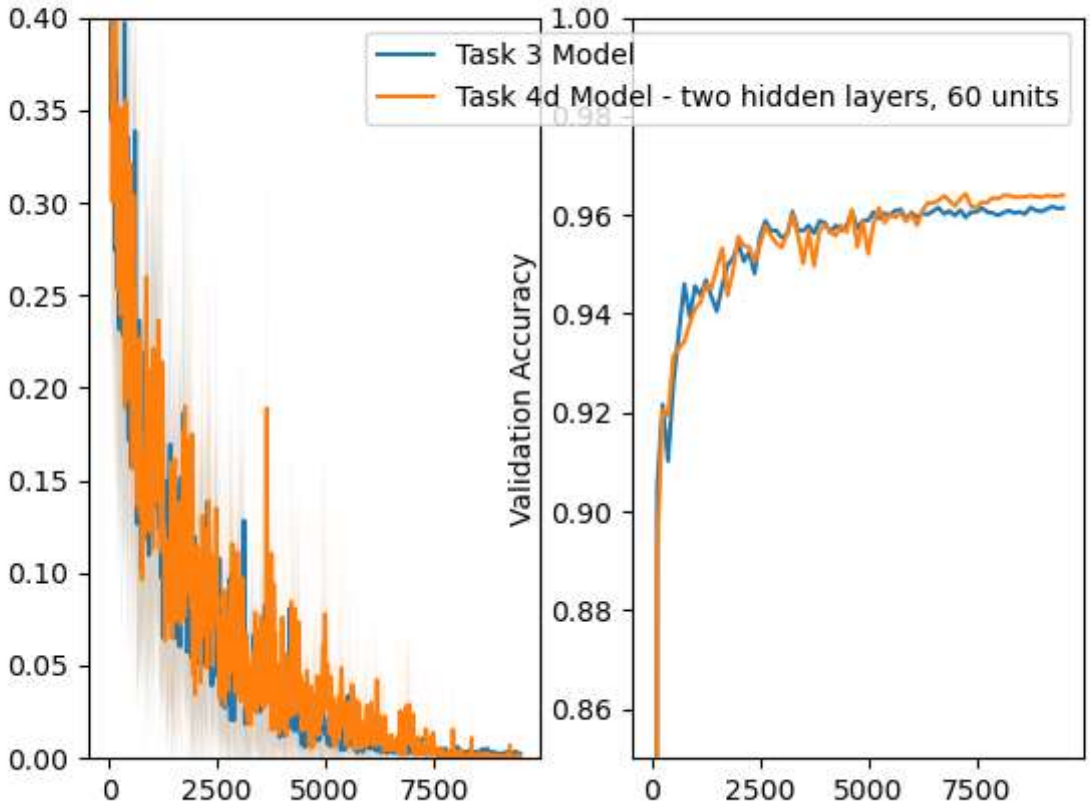
Task 4b)

The acurracy gets worse even tho the loss stil convergece to zero. The running time also increse.



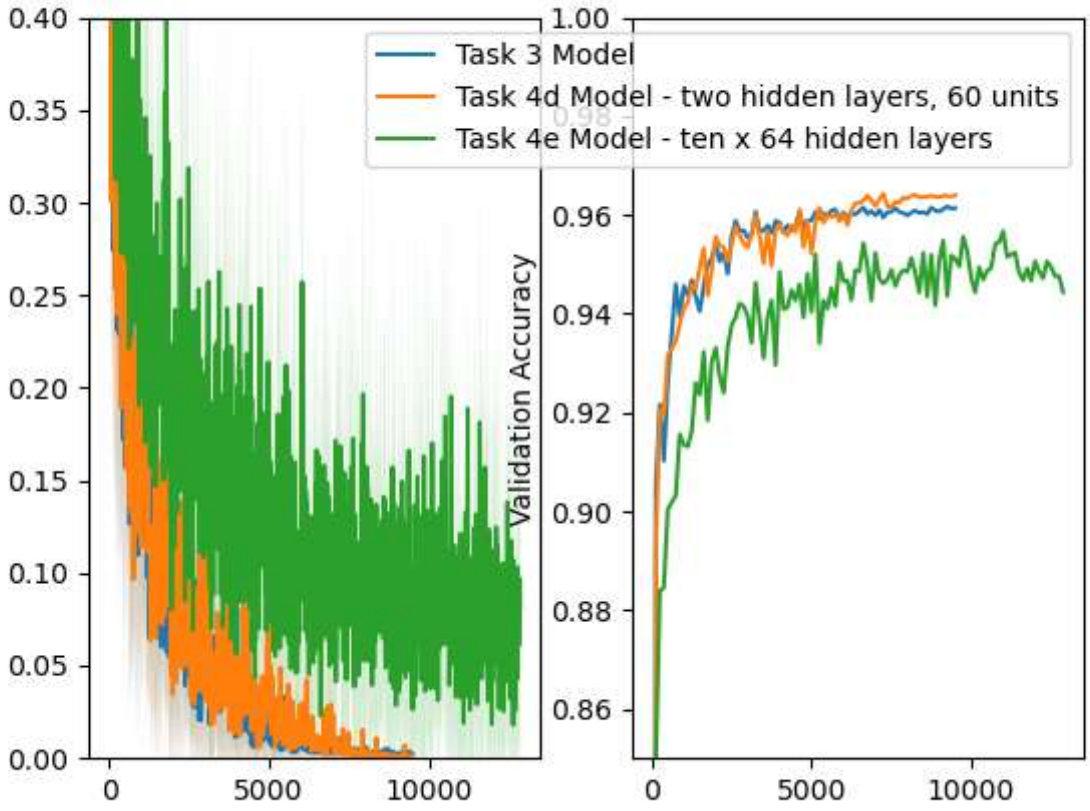
Task 4d)

This plot compares one hidden layers with 64 units to one with two hidden layers with 60 units. These shouls have aproximatly the same number of parameters.Parameters = num weights + biases = 785*60* + *60*60 + 60*10 = 51300 witch is close to the previous number of parameters! The one with two hidden layers is a little better! The difference is small and may be due to a little more parameters then the otehr!



Task 4e)

Model from task 3 vs 4d vs 4e. Model from 4e has alot more paramters and the running time was really large. More parameters alows 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, and the one with two outperformes ten! And i requires alot less computing! Early stopping was implemented to cut in computation time. Run time without was 342.85s Runtime with was 132.17s.


Task 3 Model
Task 4d Model - two hidden layers, 60 units
Task 4e Model - ten x 64 hidden layers
Validation Accuracy
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
1.00
0.96
0.94
0.92
0.90
0.88
0.86
0
5000
10000
0
5000
10000

In [ ]: