AssignmentReport-Group101

February 23, 2024

1 Assignment 2 Report

Marijan Soric and Zan Stanonik

2 Task 1

2.1 task 1a)

Backpropagation:

$$w_{ji} := w_{ji}\alpha\frac{\partial C}{\partial w_{ji}}$$

With the chain rule:

$$\frac{\partial C}{\partial w_{ji}} = \frac{\partial C}{\partial z_j}\frac{\partial z_j}{\partial w_{ji}} = \delta_j \frac{\partial}{\partial w_{ji}}\left(\sum_i w_{ij}x_i\right) = \delta_j x_i$$

With

$$\delta_j = \frac{\partial C}{\partial z_j} = (y_j\hat{y}_j) = (y_j f(z_j))$$

But also:

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

$$\frac{\partial C}{\partial a_j} = \sum_k \underbrace{\frac{\partial C}{\partial z_k}}_{\delta_k}\underbrace{\frac{\partial z_k}{\partial a_j}}_{w_{kj}} = \sum_k \delta_k w_{kj}$$

And, as activation function the sigmoid,

$$\frac{\partial a_j}{\partial z_j} = \frac{\partial f(z_j)}{\partial z_j} = f'(z_j)$$

Thus we obtain:

$$\delta_j = f'(z_j)\sum_k \delta_k w_{kj}$$

2.2 task 1b)

Vectorize computation: Input into hidden layer:

$$W^{(0)} \leftarrow W^{(0)}\alpha f'(Z) \odot (W^{(0)T}(\hat{Y} - Y)) \cdot X^T$$

Where $W^{(0)} \in \mathbb{R}^{J\times I}, Z \in \mathbb{R}^{J\times N}, Y \in \mathbb{R}^{N\times J}, X \in \mathbb{R}^{N\times I}$

Hidden layer into output layer:

$$W^{(1)} \leftarrow W^{(1)} \alpha(\hat{Y} - Y) \cdot H^T$$

Where $W^{(1)} \in \mathbb{R}^{K \times J}, Y \in \mathbb{R}^{K \times N}, H \in \mathbb{R}^{N \times J}$

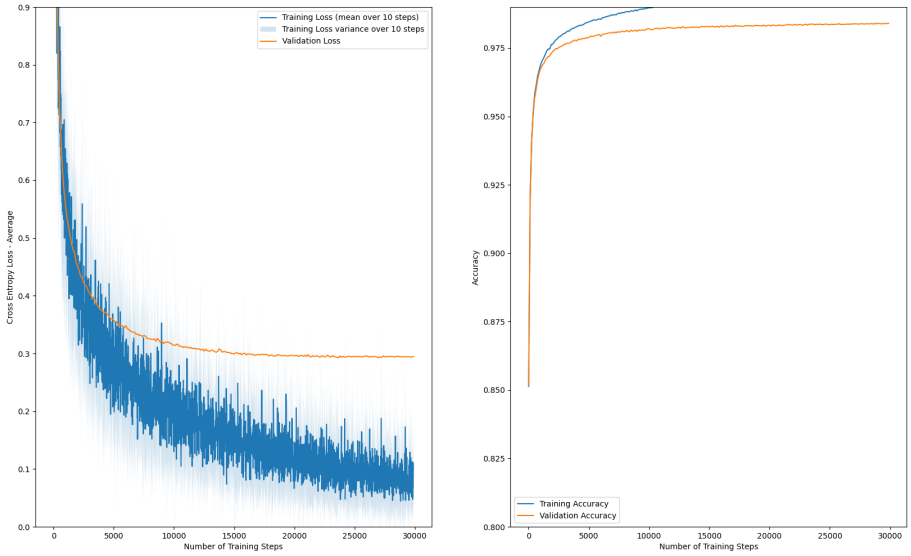
Where $\cdot$ is the matrix multiplication and $\odot$ the Hadamard product (pointwise multiplication). And $

3 Task 2

[1]:
```
import numpy as np
import utils
X, Y_train, *_ = utils.load_full_mnist()
print(f"Mean: {np.mean(X):.2f}\nStandard deviation: {np.var(X)**.5:.2f}")
```

```
Train shape: X: (20000, 784), Y: (20000, 1)
Validation shape: X: (10000, 784), Y: (10000, 1)
Mean: 33.55
Standard deviation: 78.88
```

3.1 Task 2c)



3.2 Task 2d)

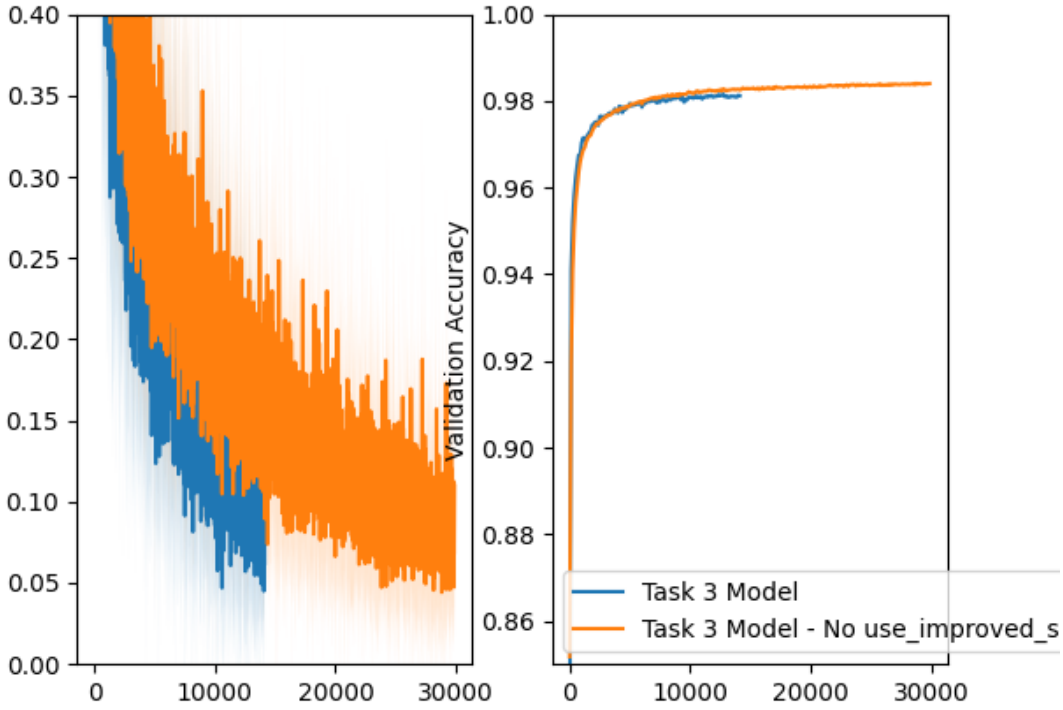
There are $785 \times 64 + 64 \times 10 = 50880$ parameters in the nework.

4 Task 3

Shortly comment on the change in performance, which has hopefully improved with each addition, at least in terms of learning speed. Note that we expect you to comment on convergence speed, generalization/overfitting, and final accuracy/validation loss of the model. Include a plot of the loss detailing the improvements for each addition. We've included task3.py as an example on how you can create this comparison plot. You can extend this file to solve this task.

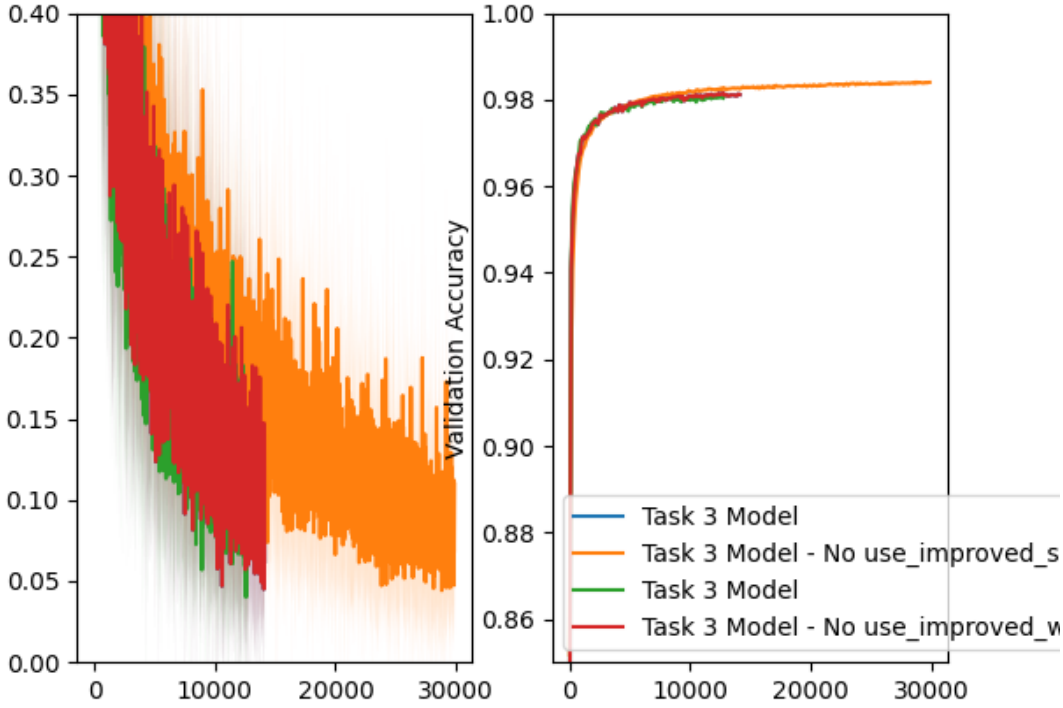
We add more more *improvement* at each step. The result doen't look good...

4.1 Task 3a) use_improved_sigmoid



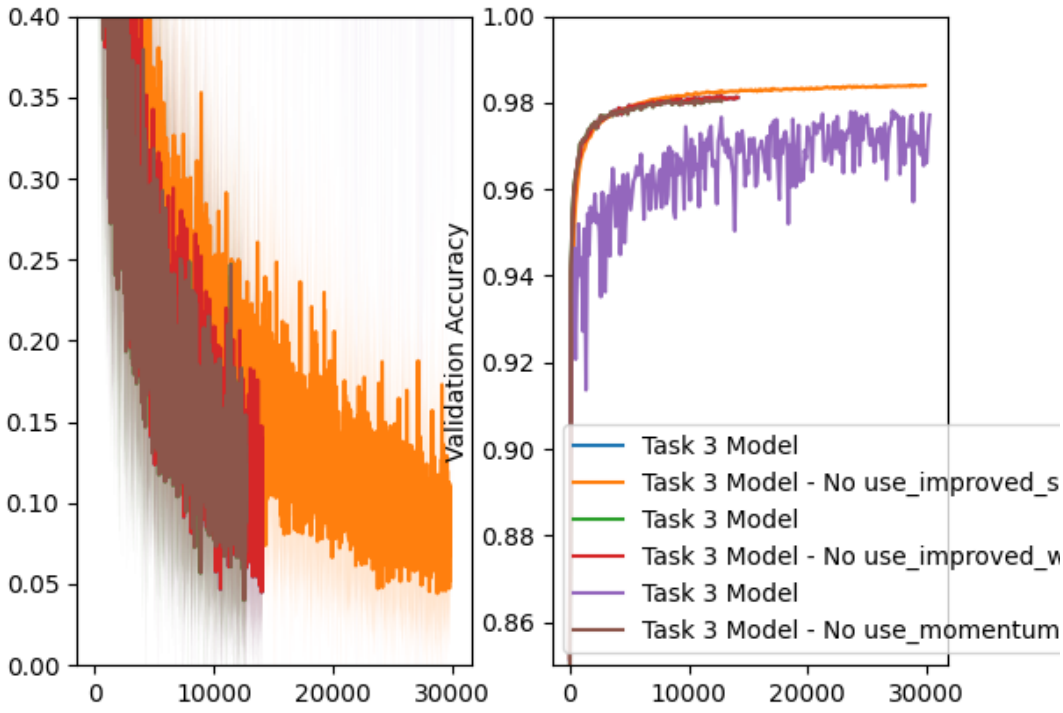
Improving the activation function help to reduced vanishing gradient problem.

4.2 Task 3b) use_improved_weight_init



Faster convergence.

4.3 Task 3c) momentum_gamma



Faster convergence and improve generalization.

5 Task 4

5.1 Task 4a)

Set the number of hidden units to 32. What do you observe if the number of hidden units is too small?

If the number of hidden units is too small, $785 \times 32 + 32 \times 10 = 25440$ (we have half as many parameters as before), the model can be too "simple" and fail to fit the data (and thus generalize well). In this case it would be underfitting.

5.2 Task 4b)

Set the number of hidden units to 128. What do you observe if the number is too large?

If the number of hidden units is too small, $785 \times 128 + 128 \times 10 = 101760$ the model can be too "complicated" and fail to generalize well the data. In this case it would be overfitting.

So the number of units in the hidden layer is an hyperparameter that should be well choosen regarding the trade off bias variance.

5.3 Task 4d)

FILL IN ANSWER

5.4 Task 4e)

FILL IN ANSWER