LAB: 03

Course: Deep Learning

Lab instructor: Irum Tariq Bhatti

NAME: RAJA HAIDER ALI

CMS ID: 346900

CLASS: BESE 11-A

# TASK 1:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Combination No. | Layer 1 | Layer 2 | Learning rate | Optimizer | Epoch | Validation loss | Accuracy / % |
| 1 | 128 | 128 | 1e-3 | SGD | 10 | 0.764056 | 81.5 |
| 2 | 128 | 128 | 1e-3 | SGD | 20 | 0.416261 | 88.5 |
| 3 | 128 | 128 | 1e-3 | Adam | 10 | 0.106768 | 97.4 |
| 4 | 128 | 128 | 1e-3 | Adam | 20 | 0.126162 | 97.8 |
| 5 | 128 | 128 | 1e-4 | SGD | 10 | 2.276998 | 23.3 |
| 6 | 128 | 128 | 1e-4 | Adam | 10 | 0.145050 | 95.6 |
| 7 | 256 | 256 | 1e-3 | SGD | 10 | 0.681894 | 83.3 |
| 8 | 256 | 256 | 1e-3 | Adam | 10 | 0.104157 | 97.7 |
| 9 | 256 | 256 | 1e-3 | SGD | 20 | 0.398885 | 88.9 |
| 10 | 256 | 256 | 1e-3 | Adam | 20 | 0.122975 | 97.8 |
| 11 | 256 | 256 | 1e-4 | SGD | 20 | 2.218220 | 47.2 |
| 12 | 256 | 256 | 1e-4 | Adam | 20 | 0.078639 | 97.5 |

# TASK 2:

For the hyperparameter tuning in the task 2, I have only chosen the best and worst scenarios from the table of task 1.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Combination No. | Layer 1 | Layer 2 | Learning rate | Optimizer | Epoch | Validation loss | Accuracy / % |
| 1 | 128 | 128 | 1e-3 | Adam | 10 | 0.351616 | 87.5 |
| 2 | 128 | 128 | 1e-3 | Adam | 20 | 0.437884 | 87.5 |
| 3 | 128 | 128 | 1e-4 | SGD | 10 | 2.218948 | 25.4 |
| 4 | 128 | 128 | 1e-4 | Adam | 10 | 0.395294 | 85.9 |
| 5 | 256 | 256 | 1e-3 | Adam | 10 | 0.375166 | 87.3 |
| 6 | 256 | 256 | 1e-3 | Adam | 20 | 0.446886 | 88.1 |
| 7 | 256 | 256 | 1e-4 | SGD | 20 | 1.984717 | 57.6 |
| 8 | 256 | 256 | 1e-4 | Adam | 20 | 0.341582 | 87.9 |

## Q1: You may change the width of each layer from 128 to 256 and 512 and observe the difference in network performance.

## Try adding and removing layers. Keeping all other setup the same, how does the network behave with the change in depth (number of layers) and width (size of each layer)?

In combination 7 of task 1, I have changed only the number of neurons in both the layers to 256 and there isn’t a significant difference in the accuracies and validation losses.

In combination 1, task 1: the accuracy is 81.5%, validation loss is 0.764056

In combination 7, task 1: the accuracy is 83.3%, validation loss is 0.681894

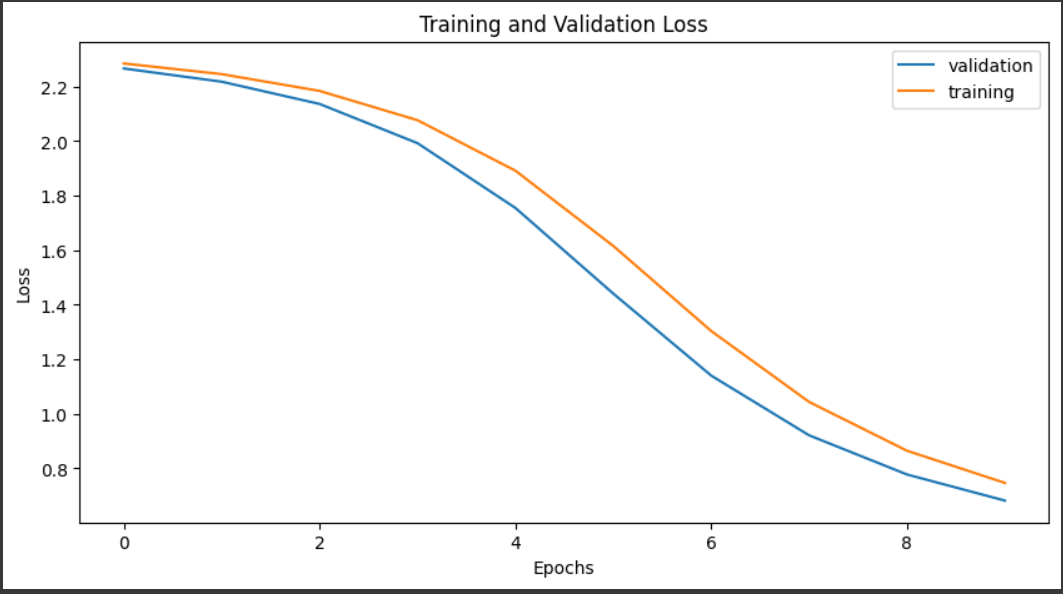
Hence, there is only a difference of 1.8% in the accuracy and 0.082162 in the validation loss which in my opinion isn’t significant enough. For better results, it seems that other hyperparameters should also be tuned.

**Combination 1, task 1:**

A graph of a person with a blue line

Description automatically generated with medium confidence

**Combination 7, task 1:**



Both combinations have almost similar learning curves.

## Q2: Try changing the learning rate from 1e-3 to 1e-4. What do you observe?

## Try the Adam optimizer instead of SGD. How the learning process changes?

By only changing the learning rate from 1e-3 to 1e-4, the performance of the optimizing algorithm degrades a lot. This degradation in performance is due to the vanishing gradient problem as the performance of the optimizing algorithm slows down. The results can be observed in combination 5 of task 1 and combination 3 of task 2 in which I have only changed the learning rate to 1e-4 while keeping all the other hyperparameters the same. These combinations give the worst accuracies and validation loss in both the tasks.

Task 1, combination 5: accuracy is 23.3% and validation loss is 2.276998.

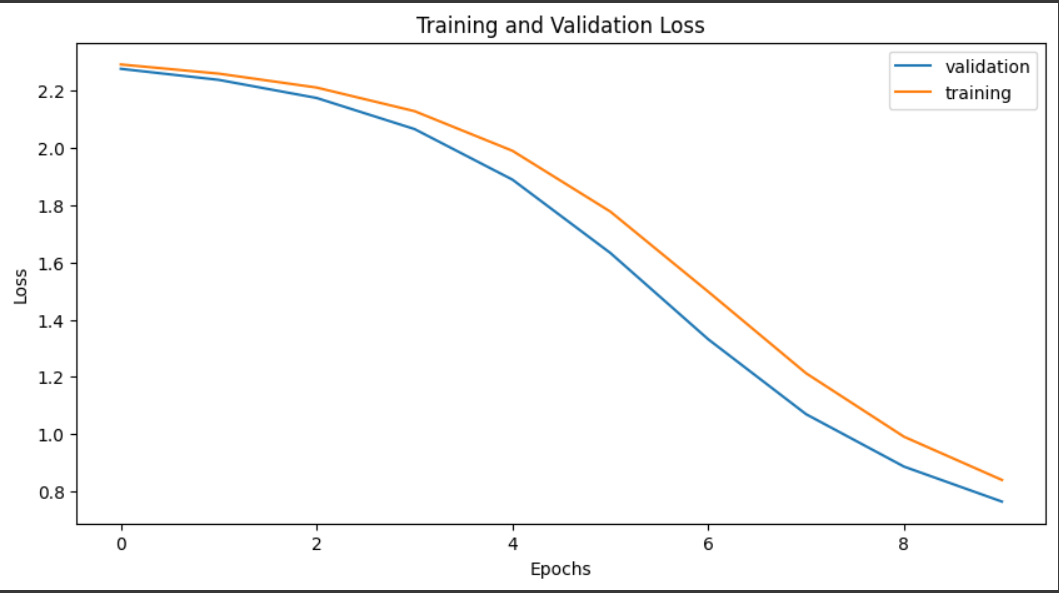
Task 2, combination 3: accuracy is 25.4% and validation loss is 2.218948

By using the Adam optimizer, the learning becomes faster in both task 1 and task 2. The effect can be observed in combination 3 of task 1 which gives the 4th best accuracy of 97.4% and validation loss of 0.106768.

The same effect in the learning process can also be observed in combination 1 of task 2 in which I have done the same change i.e. used Adam optimizer instead of SGD. The accuracy obtained is 3rd highest i.e. 87.5% and a validation loss of 0.351616.

The overall effect of using the Adam optimizer is that the learning process gets faster as the algorithm converges at a faster rate as compared to SGD.

**Combination 1, task 1:**



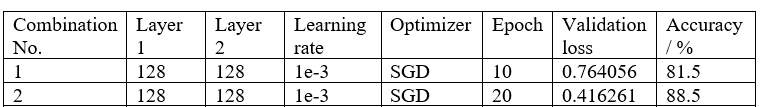
**Combination 3, task 1: showing faster convergance**



## Q3: Try 5, 10 and 50 epoch and record values for average validation loss and accuracy.

By doubling the number of epochs i.e. from 10 to 20 using the SGD optimizer, the accuracies and validation losses improve. The change can be observed in combination 2 of task1 in which the accuracy has improved by 7%.

**Task 1:**



But, if we use the Adam optimizer, then the change in the accuracies and validation losses is not significant. See the following combinations below in these I have used the Adam optimizer and have doubled the number of epochs.



However, in **task 2**, there isn’t any change in the accuracies and validation losses if we use the Adam optimizer and change the number of epochs.



## Q4: What activation function is generally used for multiclass classification? Why didn't we use an activation function at the end of our neural network that we have defined?

For multiclass classification, **Softmax activation function** is used.

We did not explicitly apply the softmax activation function at the end of the neural network because the CrossEntropyLoss function typically combines the softmax activation and the cross-entropy loss calculation into a single step. The nn.CrossEntropyLoss function expects raw scores (logits) as input for each class without applying softmax explicitly. Internally, it applies the softmax activation function and computes the cross-entropy loss, which is more numerically stable and efficient.

So, even though we didn't add a softmax activation explicitly to our model's forward pass, the CrossEntropyLoss function takes care of it when calculating the loss during training. This is a common practice in PyTorch when working with multiclass classification problems.