**Neural Network Architecture and Hyperparameter Experiments**

1. **Experiments on Validation Set**

**Experiment 1**:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Lr = 0.001, Loss\_fn = CrossEntropyLoss(), optimizer = Adam, batch\_size = 32, epoch = 100 | | | | |
| No. of Hidden Layers | Macro-Recall | F1-Macro | Accuracy | Training Period |
| 1 (50) | 0.53 | 0.53 | 0.58 | 1099.45 sec |

**Experiment 2:** Increase the batch size to 128.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Lr = 0.001, Loss\_fn = CrossEntropyLoss(), optimizer = Adam, batch\_size = 128, epoch = 100 | | | | |
| No. of Hidden Layers | Macro-Recall | F1-Macro | Accuracy | Training Period |
| 1 (50) | 0.54 | 0.55 | 0.6 | 434.28 sec |

**Experiment 3**: With batch size of 128, decrease the learning rate from Lr=0.001 to Lr=0.0001

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Lr = 0.0001, Loss\_fn = CrossEntropyLoss(), optimizer = Adam, batch\_size = 128,  epoch = 100 | | | | |
| No. of Hidden Layers | Macro-Recall | F1-Macro | Accuracy | Training Period |
| 1 (50) | 0.58 | 0.59 | 0.62 | 200.68 sec |

At epoch 100, the Ave. Loss was still 0.5393, but it still produced better results. Next, we’ll increase the epoch size from 100 to 200.

**Experiment 4:** Training with 200 epochs: Batch size = 128, learning rate = 0.0001

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Lr = 0.0001, Loss\_fn = CrossEntropyLoss(), optimizer = Adam, batch\_size = 128,  epoch = 200 | | | | |
| No. of Hidden Layers | Macro-Recall | F1-Macro | Accuracy | Training Period |
| 1 (50) | 0.57 | 0.58 | 0.61 | 388.17 sec |

At epoch 200, the Ave. Loss was at 0.48, which means that the loss only reduced by 0.06 after another 100 epochs, which took an additional 188.17 sec. The results did not improve too.

**Experiment 5:**

Using 100 epochs, batch size = 128, Lr = 0.0001, we will try to add more layers to our network.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Lr = 0.0001, Loss\_fn = CrossEntropyLoss(), optimizer = Adam, batch\_size = 128,  epoch = 100 | | | | | |
| Model | No. of Hidden Layers | Macro-Recall | F1-Macro | Accuracy | Training Period |
| A | 1 (50) | 0.58 | 0.59 | 0.62 | 171.04 sec |
| B | 2 (500, 50) | 0.55 | 0.56 | 0.60 | 992.17 sec |
| C | 2 (50, 5) | 0.58 | 0.59 | 0.62 | 213.36 sec |
| D | 3 (1000, 500, 50) | 0.52 | 0.52 | 0.57 | 1851.47 sec |
| E | 3 (50, 500, 50) | 0.53 | 0.54 | 0.58 | 226.24 sec |

It seems like our Model A (1 layer with 50 neurons) is still the best. Model C (2 Layers (50,5)) achieved the same results as Model A even with an additional layer and training duration. Adding additional layers to the left or to the right of our initial hidden layer (1 layer with 50 neurons) does not seem to benefit the performance of our neural network.

**Experiment 6:**

To try regularization, we will add Dropout(p=0.75) to Model A.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Lr = 0.0001, Loss\_fn = CrossEntropyLoss(), optimizer = Adam, batch\_size = 128,  epochs = 100, Dropout (p) = 0.5 | | | | |
| No. of Hidden Layers | Macro-Recall | F1-Macro | Accuracy | Training Period |
| 1 (50) | 0.58 | 0.59 | 0.62 | 465.44 sec |

At 100th epoch, the average loss is at 0.61. Adding Dropout with probability of 0.75 of neuron being zeroed did not improve the performance of our network, even the training period was doubled.

**Experiment 7:**

Doubling the number of neurons from 50 to 100.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Lr = 0.0001, Loss\_fn = CrossEntropyLoss(), optimizer = Adam, batch\_size = 128,  epochs = 100 | | | | |
| No. of Hidden Layers | Macro-Recall | F1-Macro | Accuracy | Training Period |
| 1 (100) | 0.58 | 0.58 | 0.61 | 587.16 sec |

Increasing the number of neurons extended the training period, but the network did not benefit from these changes. A network with 1 hidden layer of 50 neurons still performed the best.

**In the end**, we will use these hyperparameters for our model:

Number of Hidden Layers: **1**

Number of Neurons in Hidden Layer: **50**

Epochs = **100**, Batch size = **128**, Learning rate = **0.0001**,

Loss\_fn = **CrossEntropyLoss()**, optimizer = **Adam**

1. **Result on Test Set using the Hyperparameters we selected and the full training dataset.**

Model\_1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Lr = 0.0001, Loss\_fn = CrossEntropyLoss(), optimizer = Adam, batch\_size = 128,  epochs = 100 | | | | |
| No. of Hidden Layers | Macro-Recall | F1-Macro | Accuracy | Training Period |
| 1 (50) | 0.60 | 0.61 | 0.63 | 240.45 sec |

1. **Results on Test Set of Models Trained with Balanced\* Dataset**

**With Variational Auto Encoder (**num\_features=50, learning rate = 0.0001,

loss\_fn = nn.BCELoss(), optimizer = Adam, batch size = 128)

\*The distribution of this dataset:

0 (Negative) – 33.33%

1 (Neutral) – 35.39%

2 (Positive) – 31.28%

It is not perfectly balanced, but it has an improved distribution from the original dataset with a distribution of:

0 (Negative) – 16.60%

1 (Neutral) – 44.80%

2 (Positive) – 39.60%

The model was trained from the balanced\* dataset which is the result after performing Auto Encoder on the negative class.

The model is trained from 80% of the whole balanced\* dataset, and 20% was reserved for testing.

**Experiment 1: 100 epochs**

1. On the same test set used with Model\_1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Lr = 0.0001, Loss\_fn = CrossEntropyLoss(), optimizer = Adam, batch\_size = 128,  epoch = 100 | | | | |
| No. of Hidden Layers | Macro-Recall | F1-Macro | Accuracy | Training Period |
| 1 (50) | 0.74 | 0.74 | 0.75 | 298.15 sec |

1. On its own test set after performing train-test split (80-20)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Lr = 0.0001, Loss\_fn = CrossEntropyLoss(), optimizer = Adam, batch\_size = 128,  epoch = 100 | | | | |
| No. of Hidden Layers | Macro-Recall | F1-Macro | Accuracy | Training Period |
| 1 (50) | 0.70 | 0.70 | 0.70 | 298.15 sec |

**Experiment 2: 200 epochs**

1. On the same test set used with Model\_1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Lr = 0.0001, Loss\_fn = CrossEntropyLoss(), optimizer = Adam, batch\_size = 128,  epoch = 200 | | | | |
| No. of Hidden Layers | Macro-Recall | F1-Macro | Accuracy | Training Period |
| 1 (50) | 0.87 | 0.87 | 0.87 | * 1. sec |

1. On its own test set after performing train-test split (80-20)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Lr = 0.0001, Loss\_fn = CrossEntropyLoss(), optimizer = Adam, batch\_size = 128,  epoch = 200 | | | | |
| No. of Hidden Layers | Macro-Recall | F1-Macro | Accuracy | Training Period |
| 1 (50) | 0.69 | 0.69 | 0.69 | * 1. sec |

Increasing the epoch size by 100 achieved better results on the first test set, achieving **+0.13** on all metrics on average. But the model performed similarly on its own test set, with a deviation of 0.01.

**Experiment 3: 200 epochs with Dropout(p=0.8) on the Hidden Layer**

1. On the same test set used with Model\_1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Lr = 0.0001, Loss\_fn = CrossEntropyLoss(), optimizer = Adam, batch\_size = 128,  epoch = 200, Dropout (p) = 0.8 | | | | |
| No. of Hidden Layers | Macro-Recall | F1-Macro | Accuracy | Training Period |
| 1 (50) | 0.72 | 0.70 | 0.72 | * 1. sec |

1. On its own test set after performing train-test split (80-20)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Lr = 0.0001, Loss\_fn = CrossEntropyLoss(), optimizer = Adam, batch\_size = 128,  epoch = 200, Dropout (p) = 0.8 | | | | |
| No. of Hidden Layers | Macro-Recall | F1-Macro | Accuracy | Training Period |
| 1 (50) | 0.68 | 0.68 | 0.68 | 646.66 sec |

Adding Dropout on the hidden layer seems not to work in our favor. Because our network is small, with only one hidden layer, it didn’t provide the effect that we were hoping for. The model performed worse on the same test set used with Model\_1 and didn’t help it on its own test set too.

**Experiment 4: 300 epochs without Dropout**

1. On the same test set used with Model\_1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Lr = 0.0001, Loss\_fn = CrossEntropyLoss(), optimizer = Adam, batch\_size = 128,  epoch = 300 | | | | |
| No. of Hidden Layers | Macro-Recall | F1-Macro | Accuracy | Training Period |
| 1 (50) | 0.8943 | 0.8967 | 0.9013 | 854.30 sec |

1. On its own test set after performing train-test split (80-20)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Lr = 0.0001, Loss\_fn = CrossEntropyLoss(), optimizer = Adam, batch\_size = 128,  epoch = 300 | | | | |
| No. of Hidden Layers | Macro-Recall | F1-Macro | Accuracy | Training Period |
| 1 (50) | 0.6932 | 0.6933 | 0.6927 | 854.30 sec |

In the end, our **model\_2** has an improved performance over **model\_1**. At the same 100 epochs of training, model\_2 received 74% on Macro-recall, 74% on Macro-F1, and `75%` on Accuracy. These results are better than what model\_1 achieved. On the other hand, it got 70% on all metrics when tested on its own test set. So, it can be concluded that performing resampling on the `Negative` class helped the neural network to learn much better.

**model\_2** was trained for 300 epochs so that it can have more time to adjust its weights and learn better, until it arrived on its final performance shown on the two tables above.

It may or may not have seen already the data from the test set we have used on **model\_1** because of the random nature of splitting, so **model\_2** was also tested against its own test set and achieved close to 70% performance on all metrics.

Although the model is not improving any better on its own test set even for an additional 200 epochs, it can be concluded that it is still capable of getting right 70% of the time on classifying the correct sentiment of an unseen text, and 30% of the time it will classify it wrong.