This handout includes space for every question that requires a written response. Please feel free to use it to handwrite your solutions (legibly, please). If you choose to typeset your solutions, the `README.md` for this assignment includes instructions to regenerate this handout with your typeset LaTeX solutions.

3.a

Smoothed Results:

K=1, N=2 Accuracy: 0.9442 Loss: 0.3294

K=2, N=2 Accuracy: 0.9648 Loss: 0.2355

K=1, N=3 Accuracy: 0.8448 Loss: 0.5657

K=1, N=4 Accuracy: 0.5776 Loss: 1.1095

Increasing the number of classes (N) from 2 to 4 results in a significant decrease in model performance.

N=2: With 2 classes, the model achieves high accuracy (0.9442 and 0.9648) and low loss (0.3294 and 0.2355).

N=3: With 3 classes, accuracy drops to 0.8448, and loss increases to 0.5657.

N=4: With 4 classes, the accuracy drops further to 0.5776, and loss increases significantly to 1.1095.

This indicates that the model struggles to differentiate between more classes, which increases the complexity of the task and leads to higher misclassification rates and loss values.

3.b

Smoothed Results:

K=1, N=2 Accuracy: 0.9442 Loss: 0.3294

K=2, N=2 Accuracy: 0.9648 Loss: 0.2355

Increasing the number of examples in the support set (K) from 1 to 2 improves model performance.

K=1, N=2: The accuracy is 0.9442, and the loss is 0.3294.

K=2, N=2: The accuracy improves to 0.9648, and the loss decreases to 0.2355.

This shows that having more examples per class helps the model learn better representations and improves its generalization ability, resulting in higher accuracy and lower loss.

4.a

In this experiment, I analyzed the impact of different learning rates on the model's performance. The learning rates tested were 0.0001, 0.001, 0.01, and 0.1. Each experiment was run with different combinations of num/shot and num/classes to observe the model's generalization and convergence behavior.

Metrics Monitored:

Accuracy/train: Accuracy on the training set.

Accuracy/test: Accuracy on the test set.

Loss/train: Loss on the training set.

Loss/test: Loss on the test set.

Below is a detailed analysis of each learning rate across different metrics and cases.

Learning Rate: 0.0001

Training Accuracy

2.1.123.0.0001.128: Reached a final accuracy of 0.9297.

2.2.123.0.0001.128: Reached a final accuracy of 0.918.

3.1.123.0.0001.128: Reached a final accuracy of 0.6953.

4.1.123.0.0001.128: Reached a final accuracy of 0.668.

Test Accuracy

2.1.123.0.0001.128: Reached a final accuracy of 0.9023.

2.2.123.0.0001.128: Reached a final accuracy of 0.8906.

3.1.123.0.0001.128: Reached a final accuracy of 0.6901.

4.1.123.0.0001.128: Reached a final accuracy of 0.5938.

Training Loss

2.1.123.0.0001.128: Final loss - 0.2929.

2.2.123.0.0001.128: Final loss - 0.2966.

3.1.123.0.0001.128: Final loss - 0.8001.

4.1.123.0.0001.128: Final loss - 0.9946.

Test Loss

2.1.123.0.0001.128: Final loss - 0.3194.

2.2.123.0.0001.128: Final loss - 0.3418.

3.1.123.0.0001.128: Final loss - 0.8022.

4.1.123.0.0001.128: Final loss - 1.0862.

Learning Rate: 0.001

Training Accuracy

2.1.123.0.001.128: Reached a final accuracy of 0.9727.

2.2.123.0.001.128: Reached a final accuracy of 0.9766.

3.1.123.0.001.128: Reached a final accuracy of 0.8906.

4.1.123.0.001.128: Reached a final accuracy of 0.873.

Test Accuracy
2.1.123.0.001.128: Reached a final accuracy of 0.9453.
2.2.123.0.001.128: Reached a final accuracy of 0.9531.
3.1.123.0.001.128: Reached a final accuracy of 0.8698.
4.1.123.0.001.128: Reached a final accuracy of 0.5605.
Training Loss
2.1.123.0.001.128: Final loss was 0.298.
2.2.123.0.001.128: Final loss was 0.2167.
3.1.123.0.001.128: Final loss was 0.5084.
4.1.123.0.001.128: Final loss was 0.7814.
Test Loss
2.1.123.0.001.128: Final loss was 0.3286.
2.2.123.0.001.128: Final loss was 0.2499.
3.1.123.0.001.128: Final loss was 0.539.
4.1.123.0.001.128: Final loss was 1.1152.

Learning Rate: 0.01

Training Accuracy
2.1.123.0.01.128: Reached a final accuracy of 0.8828.
2.2.123.0.01.128: Reached a final accuracy of 0.8945.
3.1.123.0.01.128: Reached a final accuracy of 0.9094.
4.1.123.0.01.128: Reached a final accuracy of 1.0819.
Test Accuracy
2.1.123.0.01.128: Reached a final accuracy of 0.8867.
2.2.123.0.01.128: Reached a final accuracy of 0.8125.
3.1.123.0.01.128: Reached a final accuracy of 0.388.
4.1.123.0.01.128: Reached a final accuracy of 0.3789.
Training Loss
2.1.123.0.01.128: Final loss - 0.4187.
2.2.123.0.01.128: Final loss - 0.3958.
3.1.123.0.01.128: Final loss - 0.9094.
4.1.123.0.01.128: Final loss - 1.0819.
Test Loss
2.1.123.0.01.128: Final loss - 0.4282.
2.2.123.0.01.128: Final loss - 0.4494.
3.1.123.0.01.128: Final loss - 1.0986.
4.1.123.0.01.128: Final loss - 1.2895.

Learning Rate: 0.1

Training Accuracy

2.1.123.0.1.128: Reached a final accuracy of 0.5.

2.2.123.0.1.128: Reached a final accuracy of 0.5391.

3.1.123.0.1.128: Reached a final accuracy of 0.3411.

4.1.123.0.1.128: Reached a final accuracy of 0.5273.

Test Accuracy

2.1.123.0.1.128: Reached a final accuracy of 0.5.

2.2.123.0.1.128: Reached a final accuracy of 0.5.

3.1.123.0.1.128: Reached a final accuracy of 0.3229.

4.1.123.0.1.128: Reached a final accuracy of 0.25.

Training Loss

2.1.123.0.1.128: Final loss was 0.6931.

2.2.123.0.1.128: Final loss was 0.2167.

3.1.123.0.1.128: Final loss was 1.0986.

4.1.123.0.1.128: Final loss was 1.3863.

Test Loss

2.1.123.0.1.128: Final loss was 0.6931.

2.2.123.0.1.128: Final loss was 0.2167.

3.1.123.0.1.128: Final loss was 1.0986.

4.1.123.0.1.128: Final loss was 1.3863.

Conclusion:

Optimal Learning Rate: 0.001 provided the best balance between learning speed, convergence, and generalization.

High Learning Rates (0.1): Led to instability and poor generalization, likely due to overshooting during gradient descent.

Low Learning Rates (0.0001): Resulted in slow learning and potential underfitting, requiring more epochs to converge effectively.

Rationale for Choosing Learning Rate as the Hyperparameter:

The learning rate is a crucial hyperparameter in training neural networks as it controls the step size of the gradient descent optimization. It directly affects the convergence speed and the stability of the training process. By experimenting with different learning rates, I can find the optimal value that balances the trade-off between fast convergence and stable, effective learning.

4.b

In the experiment, I analyzed how varying the memory capacity of an LSTM model influences its ability to process and retain information over time, focusing specifically on the K = 1, N = 3 case with different hidden state sizes: 8, 128, and 256 units.

Here's what I observed:

High Memory Capacity (256 units)

Accuracy: Smoothed at 86.75

Loss: Smoothed test loss recorded at 0.5304, showing efficient error minimization.

With the largest memory capacity tested, the model displayed exceptional performance, managing to capture complex patterns and dependencies in the data. This size allows the model to hold a significant amount of information, facilitating a deeper understanding of long-term relationships in the sequence data.

Medium Memory Capacity (128 units)

Accuracy: Smoothed at 84.48

Loss: Smoothed test loss was 0.5657, a bit higher than the largest model.

This setup shows a balanced approach, maintaining good performance while being less resource-intensive than the largest model. It handles long-term dependencies well, though not as adeptly as the 256-unit model, suggesting that there is a noticeable trade-off between memory capacity and the ability to generalize over longer sequences.

Low Memory Capacity (8 units)

Accuracy: Smoothed at only 55.39

Loss: Smoothed test loss worsened to 0.9219.

This model struggled with long-term dependencies due to its limited capacity to retain earlier information, which is crucial for understanding sequences. The performance drop is stark, underscoring how essential adequate memory is for tasks that require detailed memory of past inputs.

Summary:

The experiment highlights a clear trend: increasing the memory capacity of an LSTM significantly boosts its performance by enhancing its ability to understand and remember long-term dependencies. Conversely, reducing the memory capacity severely hampers the model's performance, indicating that a certain threshold of memory is crucial for effectively processing sequential data.