Submission for Deep Learning Exercise 7

Team: dl2024-sigma_learning
Students: Uralp Ergin (5975013), İsmail Karabaş (7654321), Ahmet Yaşar Ayfer (5986167)

December 11, 2024

1 Preliminary Questions

1.1

Both CNNs and RNNs are layered and hierarchical structures that process data in a sequential/layered fashion. They learn features without any manual intervention, using an automated training process. Both architectures share weights across their networks: CNNs share filters across spatial dimensions, while RNNs share weights across time steps. They both utilize backpropagation to update weights, parameters, and other network components.

1.2

Several techniques can be used to counteract vanishing and exploding gradient problems in RNNs:

- **Gated Architectures:** Using LSTM or GRU networks, which are specifically designed to maintain long-term dependencies through their gating mechanisms
- **Gradient Clipping:** Enforcing a maximum threshold on gradient values to prevent explosion during backpropagation
- **Weight Initialization:** Implementing proper initialization techniques like Xavier or He initialization to maintain gradients within a stable range
- **Regularization:** Applying techniques such as dropout and L1/L2 regularization to stabilize the training process
- **Adaptive Optimization:** Using advanced optimizers like Adam or RMSprop that dynamically adjust learning rates, leading to more stable convergence

2 Design Pattern and Tensor Shapes

2.1 Which design pattern from the ones mentioned in the lecture is used for the Noise Removal Task and why? What is the shape of the input tensor and how does it change as we propagate through the two LSTM layers and the final linear layer?

The Noise Removal Task uses the many-to-many design pattern because it transforms a noisy input sequence into a denoised output sequence of the same length, with each output depending on both past and future inputs through the shift parameter.

The tensor shape transformations are:

- Input: (batch_size, sequence_length, 1)

- After first LSTM: (batch_size, sequence_length, hidden_size)
- After second LSTM: (batch_size, sequence_length, hidden_size)
- After final linear layer: (batch_size, sequence_length, 1)

The model maintains the temporal sequence structure while processing through hidden representations before returning to the original dimensionality.

3 Hyperparameter Optimization Results

Configuration	**Hidden Size**	**Shift**	**Epochs**	**Noise Removed (%)**
Initial	40	10	100	58.51
Increased Epochs	40	10	200	59.45
Decreased Hidden Size	20	10	100	53.80
Increased Hidden Size	60	10	100	57.76
Modified Shift	40	3	100	56.64

Table 1: Results of different hyperparameter configurations for noise removal

The results show that increasing the number of epochs to 200 yielded the best performance with 59.45% noise removal. Reducing the hidden size to 20 resulted in the worst performance (53.80%), suggesting that the model benefits from having more capacity. The initial configuration with hidden size 40 performed well, while modifications to shift and hidden size showed varying degrees of effectiveness.