Submission – Assignment 7

Marta Gulida – matriculation number 5585808 – mg776
Tillman Heisner – matriculation number 4517815 – th273
Erik Bode – matriculation number 4505199 – kb301

Pen and Paper Task

1) In what sense are convolutional neural networks and recurrent neural networks similar? In what sense are they different?

Similar: both are specialized types of neural networks designed to process data with a grid-like topology, both capable of handling complex tasks in machine learning (image recognition), often used in DL architectures, both learn features and patterns from data through a process of training where weights are adjusted to minimize the loss function.
However, they are tailored for different types of data inputs and problems. CNNs are designed for processing data that has a spatial relationship, such as images, where the convolutional layers can capture the hierarchical patterns and structures within the data.
RNNs are designed to handle sequential data, such as time series or natural language. They are distinguished by their ability to maintain a "memory" of previous inputs through their internal state, which influences the processing of subsequent inputs. This makes RNNs ideal for tasks where context and order of data points are crucial.
The fundamental difference in the way they process data: CNNs apply the same filters across the entire space of the input data, effectively sharing weights, whereas RNNs process sequences step-by-step, maintaining a hidden state that captures information about the sequence processed so far.
RNNs suffer from challenges like gradient vanishing and exploding problems due to long sequences.

2) How can one counteract vanishing or exploding gradients in RNNs, allowing to learn long-term dependencies?

To counteract these issues, several techniques can be used:

1. **Gradient clipping.** This technique involves clipping the gradients during backpropagation to prevent them from exploding.
2. **Weight initialization.** Careful initialization of weights can help mitigate the vanishing gradient problem.
3. **Gated architectures.**

3) 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 design pattern: *"Sequence-to-Sequence"*. It is characterized by the use of LSTM networks to process sequences of data. The model is designed to take in a sequence (the noisy signal) and produce a sequence of the same length (the denoised signal).

The input tensor's shape changes from $(batch\_size, sequence\_length, 1)$ to $(batch\_size, sequence\_length + self.shift, 1)$ after padding, then to $(batch\_size, sequence\_length + self.shift, hidden\_size)$ after the first LSTM, to $(batch\_size, sequence\_length, hidden\_size)$ after slicing, it remains the same after the second LSTM, and finally becomes $(batch\_size, sequence\_length, 1)$ after the linear layer, where the final 1 represents the feature size of the output signal (the denoised signal).

4) Hyperparameter Configuration

Configuration	Noise removed in percent
Initial Configuration	56.0792%
Train for 300 epochs	59.5056%
Batch size 5	58.6006%
Hidden size 70	60.6227%
Hidden size 80 and shift 5	59.0705%