Recurrent Neural Networks

Introduction

With this exercise, you will learn about recurrent neural networks (RNNs). In contrast to fully connected or convolutional neural networks, RNNs operate on sequential data. Thus, they are used for tasks such as translation, image captioning, and next element prediction.

Looking at a text-prediction task, one objective for the network could be to continue a text snippet given just a part of it. This way, one can generate diverse text snippets from a given starting point.

A recurrent neural network can consist of multiple layers, which in turn consist of multiple RNN cells. For simplicity, we will focus on LSTM cells, which are one of the most used cell architectures in this context.

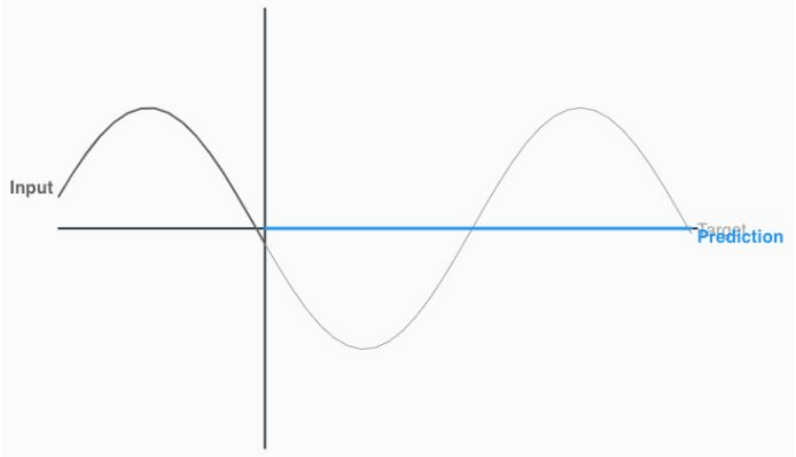
Training

As with other types of neural networks, there are different parameters that can be manipulated during training. In the following, you will learn how a few of these parameters can influence the training success for the example application of function continuation.

Learning Rate

Low Learning Rate

When the learning rate is too low, one can see that the model prediction is not good, even after 20 epochs. Indicating a learning process that is too slow.



High Learning Rate

By contrast, when the learning rate is too high, the model prediction is also not good after 20 epochs, indicating overshooting of optima during training.

Medium Learning Rate

When the learning rate is selected appropriately, the model finds an optimal solution fast, even after 10 epochs.

Low Batch Size

When the batch size is low, the model is learning slowly, not being converged even after 20 epochs.

High Batch Size

By contrast, when the batch size is too high, the loss is averaged over too many samples, thus, also not converging after 20 epochs.

Medium Batch Size

Again, when the batch size is selected appropriately, the model finds an optimal solution fast, even after 10 epochs.

No Noise

Without noise in the training data, it is easy for the model to learn. However, this is not a very realistic scenario, as real-life data is often not as smooth.

Some Noise

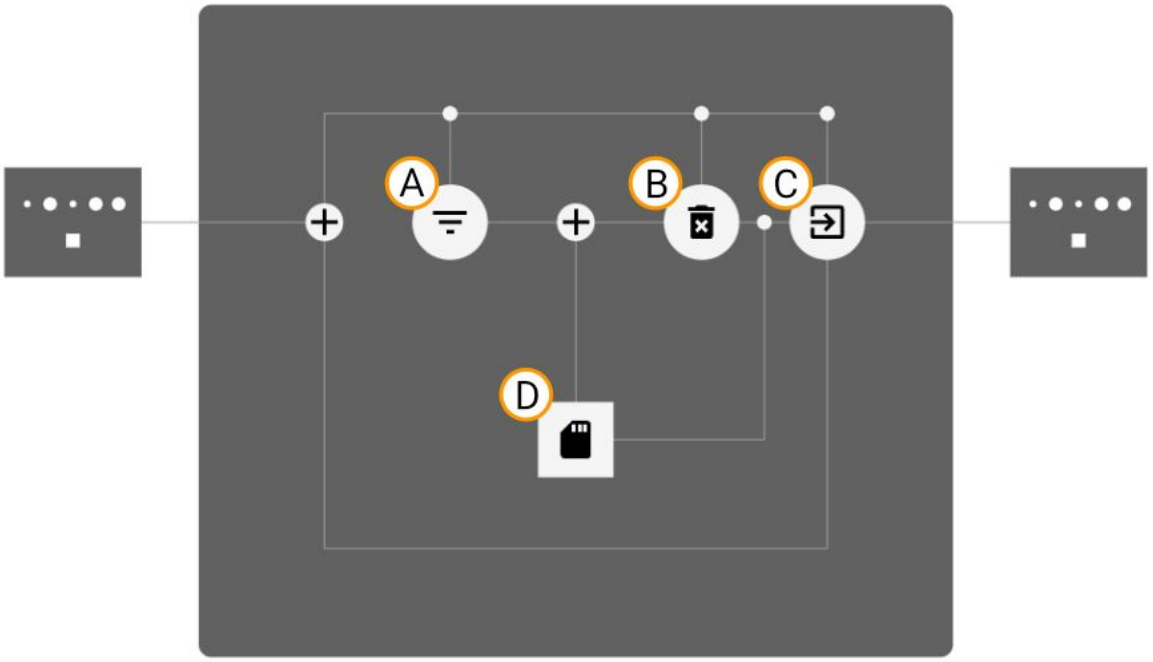
Adding noise reflects a more realistic scenario. As this is harder to learn, it takes longer for the modele to converge.

High Noise

Noise can also be too much for the model to handle. When the noise is too much, the model might not converge at all.

LSTM Cells

An LSTM cell consists of different components, which process the data that is fed into the network. We will explain these components in the following.



Data for a cell can come from previous layers, or directly be the input to the network. Upon arrival of new data, this data is combined with the previous activation of the cell for further computations.
The aforementioned combination of input data and previous activation is then fed to the three gates that a LSTM cell has. They are called Input Gate (A), Forget Gate (B), and Output Gate (C). All of these Gates also interact with the Cell State (D).

Cell State

The cell state is the heart of any LSTM cell. By having a cell state, and deciding how to update it based on the filtered input and previous state, LSTM cells are able to capture long-term dependencies.

$$c^t = \text{filtered\_input} + \text{filtered\_state}$$

Symbols:

$$c^t : \text{the cell state at timestep t}$$

Input Gate

The input gate is used as a measure for what part of the input is used for updating the cell state with new information. It only lets through information that might be helpful for future predictions of the cell.

$$i^t = sigmoid(W_{ix}x^t + W_{ia}a^{t-1} + b_i)$$

It is used to calculate the filtered input to update the internal cell state.

$$\text{filtered\_input} = i^t \circ tanh(W_{cx}x^t + W_{ca}a^{t-1} + b_c)$$

Symbols:

i^t : the input gate

x^t : the input from the previous cell at time step t

a^{t-1} : the activation of this cell from the previous time step

W : trainable weight parameters

b : trainable bias parameters

Forget Gate

The forget gate determines what part of the old cell state can be forgotten.

$$f^t = sigmoid(W_{fx}x^t + W_{fa}a^{t-1} + b_f)$$

It is used to calculate the filtered state to update the internal cell.

$$\text{filtered\_state} = f^t \circ c^{t-1}$$

Symbols:

f^t : the forget gate

x^t : the input from the previous cell at time step t

a^{t-1} : the activation of this cell from the previous time step

c^{t-1} : the cell state from the previous time step

W : trainable weight parameters

b : trainable bias parameters

Output Gate

The output gate is responsible for filtering what part of the current cell state is used as the output activation of the cell.

$$o^t = sigmoid(W_{ox}x^t + W_{oa}a^{t-1} + b_o)$$

Using the output gate, one can then compute the cell activation, which is used as input to the next cell, as well as recurrently in the same cell at the next time step.

$$a^t = o^t \circ tanh(c^t)$$

Symbols:

o^t : the output gate

a^t : the activation of this cell for the current timestep

x^t : the input from the previous cell at time step t

a^{t-1} : the activation of this cell from the previous time step

c^t : the current cell state

W : trainable weight parameters

b : trainable bias parameters