Exercise: 2

Deadline: 02 June 2025 23:59:59 GMT+1

Aufgabe 1: Your own small Deep Learning Framework

The aim of this exercise is to add additional layers to the framework. The focus should be on implementing the Convolution Layer. For their implementation, the numpy functions correlate and convolve, as well as similar functions, are not allowed. Other numpy functions are allowed.

(a) Create a Conv2DLayer class with:

A filter F (3-Dimensional), as well as the number of filters as parameters. For this purpose, implement the following methods:

1. forward(...) $\rightarrow Y = X * F + bias$
2. backward(...)$\rightarrow \delta X = \delta Y *_F rot^{180}_{\text{x,y}}(trans_{0,1,3,2}(F))$
3. calculateDeltaWeights(...) $\rightarrow \frac{\partial L}{\partial f} = X *_{ch} \delta Y$, $\frac{\partial L}{\partial bias_f} = \sum_i \delta y_{\text{i, f=c}}$

(b) (Optional) Extend the Conv2DLayer class to include:

1. A **stride** parameter, a 2-Dimensional shape, which indicates by how much the filter is shifted.
2. A **dilation** parameter, which extends the field of view of the filter by inserting zeros.

(c) Convolution Layers are often used in combination with Pooling Layers to reduce the number of trainable parameters in the network. Implement a Pooling2D class with:

- the options **MAX** or **AVERAGE** (optional) to distinguish between Max-Pooling and Average Pooling.
- The **axis** parameter, which indicates in which plane the pooling filter is moved.
- the associated forward() and backward() methods
- the parameters **kernel_size** and **stride**

(d) Saving and loading

Implement loading and saving for the Convolution Layer, as you have already done for the Fully Connected Layer.

Aufgabe 2: MNIST

(a) Use a CNN for the classification of the MNIST data. Let your implementation run for 10-20 epochs and note down for each epoch, the epoch no., the runtime for the epoch and the average loss for the epoch. Also note down the accuracy of the trained network on the test data. Your implementation should achieve an accuracy of $> 95\%$. What observations do you make? Can you achieve better quality than with exclusively Fully Connected Layers? Write down your notes

and observations in the README.md file and commit them together with the saved network weights to the GitLab repository. Your implementation should be efficient enough that a CNN can be trained on the MNIST dataset in an acceptable time.