Heidelberg University
Institute of Computer Engineering
Computing Systems Group

Holger Fröning
Hendrik Borras
Christian Simonides

Embedded Machine Learning
Summer Term 2024

Exercise 2

- **Return electronically until Wednesday, May 1, 2024, 09:00**
- **Include your names on the top sheet. Hand in only one PDF.**
- **A maximum of four students are allowed to work jointly on the exercises.**
- **When you include plots add a short explanation of what you expected and observed.**
- **Hand in source code if the exercise required programming. You can bundle the source code along with the PDF in a .zip file.**
- **Programming exercises can only be graded if they run on the cluster in the provided conda enviroment. Make sure to document additional steps, which you might have taken to run the exercises.**

2.1 Reading

Read the following paper and provide a review:

- Sharan Chetlur, et al. 2014. cuDNN: Efficient Primitives for Deep Learning.

(5 points)

2.2 Neural network from scratch

In this exercise we are going to implement a small three layer MLP with sigmoid activations from scratch and then train it on the MNIST dataset. For this we provide a template script, which you can use to start off with. The template already contains a lot of the structure of the network, dataset loading and basic training routine, so the main task boils down to implementing the forward and backward paths and and update routines for different components.

- Note: If you follow implementation in the template, then the script won't use any GPU capability. Which is fine for such a small MLP and task, so you are fine just running it on a CPU.
- First, setup your conda environment on the cluster or on your own machine, as explained in the brook user manual. Please use the included `conda_envyaml` file to create the environment. This is the same environment as in the last exercise, so you can reuse the existing one if you had already set it up.
- Implement the missing forward and backward paths and update routines for the three layer MLP, as outlined in the `exercise02_template.py` file. These are in particular the central components of the Linear/Dense layer, Sigmoid activation function and softmax function.
- Train the network with the given default parameters for 30 epochs. Then plot how the test and train loss develop with the number of epochs. Additionally create a plot, which shows how the test accuracy develops.
- Run the training again for 30 epochs, but this time with varying learning rates. Choose at least five different learning rates between 1. and 0.001. Create a plot with the test accuracy over epochs to compare the different learning rates.
- Discuss the results obtained in the previous plots.

(40 points)

2.3 Willingness to present

Please declare whether you are willing to present any of the exercises from this sheet. The declaration should happen on the PDF which you hand in.
The declaration can be made on a per-exercise basis. Each declaration corresponds to 50% of the exercise points. You can only declare your willingness to present exercises for which you also hand in a solution. If no willingness to present is declared you may still be required to present an exercise for which your group has handed in a solution. This may happen if as example nobody else has declared their willingness to present.

- Reading (Section: 2.1)
- Neural network from scratch (Section: 2.2)

(23 points)

Total: 68 points