Deep Learning

Practical Exercises - Week 10-11

Deep Feedforward Network

ICAI

FS 2024

1 Topic

In the last few weeks we have acquired the knowledge and tools to build and train deep neural networks using TensorFlow. In the remaining lab sessions we will use this knowledge to design and optimize various deep networks, starting with deep feedforward neural networks.

2 Deep Feedforward Neural Network

In lab 8 and 9 we developed a softmax regression model which can classify handwritten digits (MNIST dataset). The training process is now split into four different modules to increase clarity.

main.py
Loads the datasets and starts the training.

train.py
Implements the training, validation and testing procedures and logs some metrics which can be displayed with TensorBoard.

data.py
Loads and preprocesses the MNIST images and prepares them as TF Dataset.

model.py
Implements the softmax regression model as derived tf.keras.Model class.

First try to understand how this softmax regression model and the corresponding training is implemented in these modules.

In its present form, the classifier achieves a performance of approximately 93 % on the test dataset. Try to increase the performance of the network by using some of the methods introduced in chapters 7 and 8 of the book. Possible adjustments could be:

- Adding additional hidden layers and/or changing the size of the hidden layers.
- Don't forget to add an activation function after a fully connected layer (e.g. ReLU, PReLU or LeakyReLU).
- Longer training (train network for more epochs).
- Adding L^2 parameter regularization (weight decay).
- Adding data augmentation to artificially increase the amount of training data.
- Implementing early stopping using the validation accuracy or validation loss as reference score.
- Using dropout on the hidden layers. Please note that the dropout has a different behavior during training than it does at inference.
- Trying different parameter initialization strategies (see function parameter `kernel_initializer` of Dense layer).
- Using different a optimizer with adaptive learning rates.
- Using batch normalization. Please note that the batch normalization has a different behavior during training than it does at inference.
- Make yourself familiar with the TensorFlow homepage. In particular, try to get an overview of the Python API and the tutorials and guides.

By the appropriate use of some of these methods it is possible to achieve a performance of up to 99 % on the test dataset.