TUM EI 70360: Machine Learning and Optimization
Fall 2023
Lecturer: Reinhard Heckel
Teaching Assistant: Tobit Klug

**Problem Set 10**
Issued: Tuesday, Dec. 19, 2023
Due: Thursday, Jan. 11, 2023

---

**Problem 1** (Computational cost of stochastic gradient descent for multi-layer perceptrons)**.** Consider a multi-layer perceptron mapping a $d$-dimensional input feature vector to a scalar output. The neural network has $q$ hidden layers and $k$ neurons in each layer. We train the network on a training set consisting of $n$ training examples with a quadratic loss. What is the computational cost of one step of stochastic gradient descent with batch-size one? Explain your result. Assume that for computing gradients you are using either backpropagation or an auto-differentiation package as discussed in class, and providing the answer up to a constant factor (in O-notation) is sufficient.

**Problem 2** (Neural networks with PyTorch)**.** In this problem, the goal is to utilize the PyTorch library to train a simple fully connected neural network to classify RGB color images from the CIFAR10 dataset. The jupyter notebook *neural_networks_with_pytorch.ipynb* already contains code to load the data. The dataset consists of 10 classes plane, car, bird, cat, deer, dog, frog, horse, ship, truck, where there are 5000 training and 1000 test images per class. Each image has dimensions $3 \times 32 \times 32$. The dataset also includes labels for each image.

1. In this problem we only consider images of the classes cat, dog and ship and train/test on 3000/1000 images per class. Start by reducing the size of the training and test set accordingly. Then, initialize the train and test dataloaders. We use a batch size of $B = 4$.

2. Inheriting from `nn.Module`, implement a simple fully connected neural network with a single hidden layer of dimension 512, ReLU activations (not for the output layer) and an output dimension equal to the number of classes. Note that for the first linear layer to process a batch of input images the batch needs to be flattened across the color channels and spatial dimensions to a size of $B \times 3072$.

3. Define a SGD optimizer with learning rate 0.001 and momentum 0.9 from `torch.optim`. With momentum the optimizer adds a weighted running average of the gradients per parameter to the current gradient in each step improving both stability and speed of convergence. Use the `torch.nn.CrossEntropyLoss()` loss and train the network for 10 epochs. Report the train and test accuracy after every epoch and save the best model.

4. Load the best model and report the overall test accuracy and the test accuracy per class. On which class does our classifier perform best and why?