BEUTH HOCHSCHULE
FÜR TECHNIK
BERLIN
University of Applied Sciences

Prof. Dr.-Ing. Kristian Hildebrand

Learning from Images
Pytorch, SVMs, Neural Networks and CNNs (15 Points)

WiSe 2019/20

The aim of this exercise is to introduce you to classification problems for images. For this we will compare Support Vector Machines and Neural Networks. You will get to know Pytorch *a deep learning framework* and to develop a deeper understanding of Convolutional Neural Networks (ConvNets), their architecural properties and most successful architectures.

Exercise 1: Support Vector Machines (5 Points)

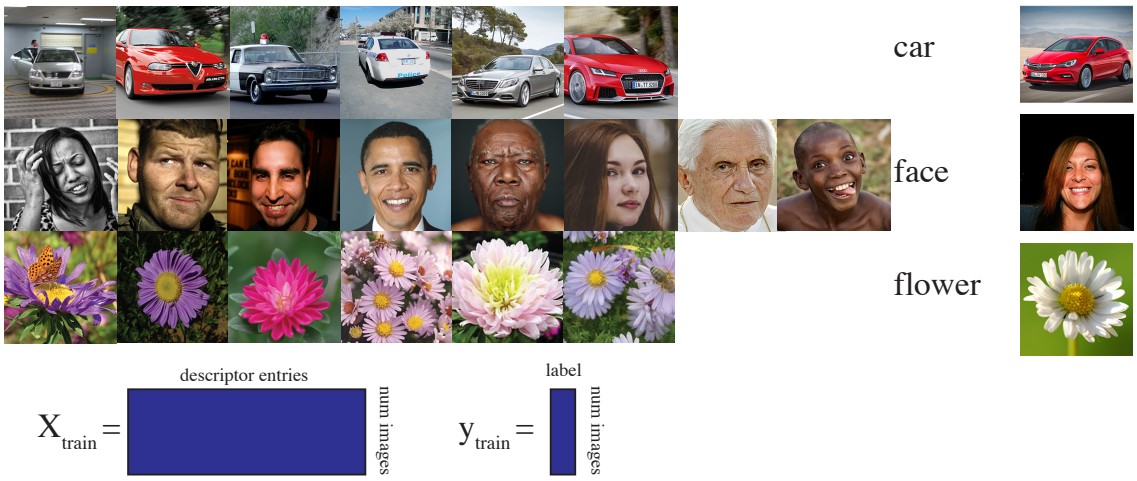

Figure 1: (Left) Training images in three categories. (Right) Test images which should be classified correctly after training.

A Support Vector Machine is a *supervised* learning algorithm and often serves as a classifier. For classification, image descriptors are used as discussed in the lecture (e.g. SIFT). The SVM must first be trained on the supplied image data set. Therefore you have to extract image features similar to assignment 2.1 and then write them as line vectors in a training matrix (X_{train}) (see figure 1). For each line entry in X_{train} there is a corresponding line entry in the label vector y_{train}. Please use the test pictures to ensure the trained classifier is working (please use the dataset from Task 2.1). Use scikit-learn *svm.SVC* to train the SVM and try different kernels (not all work well). Further comments can be found in the source code in *svm.py*.

Exercise 2: Implement your own neural network from scratch (5 Points)

For this task you need to get yourself familiar with Pytorch (https://pytorch.org/tutorials/). Therefore you need to implement a two-layer neural network from scratch as we have discussed in the lecture but use Pytorch variables and data structures. You can take the Jupyter Notebook from our lecture as a basis and extend it. The file *two-layer-nn.py* will give you some more detailed instructions on input, architecture, activation functions and training parameters. For training the network you will

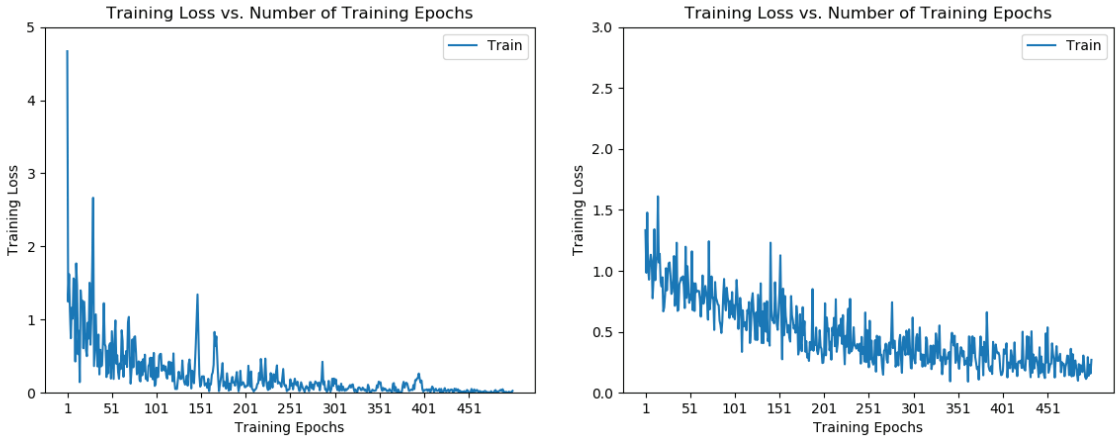

Figure 2: Plots of loss functions outcome from the two-layer neural network. (Left) MSE Loss (Right) Cross-entropy loss.

need to compare two different loss functions over the simple image database from assignment 2.1 (image retrieval):

a) mean-squared error loss function

b) the cross-entropy loss function

This will result in the following training plots and test results. **Note:** I am aware of the fact that this is not a realistic use case and will overfit your training data. It is just for educational purposes.

Please hand in your code so that I can run the code and view the plots and the test results as given without any modification from my side.

Exercise 3: Convolutional Neural Network Architectures (5 Points)

The third task in this homework builds on your new PyTorch skills. For this we use the so-called *Fashion MNIST* dataset from the Zalando Research team (https://github.com/zalandoresearch/fashion-mnist#get-the-data). It consists of 60k training images and 10k test images from 10 different categories. Each image has a size of 28x28px. The provided script *cnn.py* will download the data for you. Your task is to create three different neural network architectures with at least over 80% in their classification accuracy. The focus of the task is to implement different architectures.

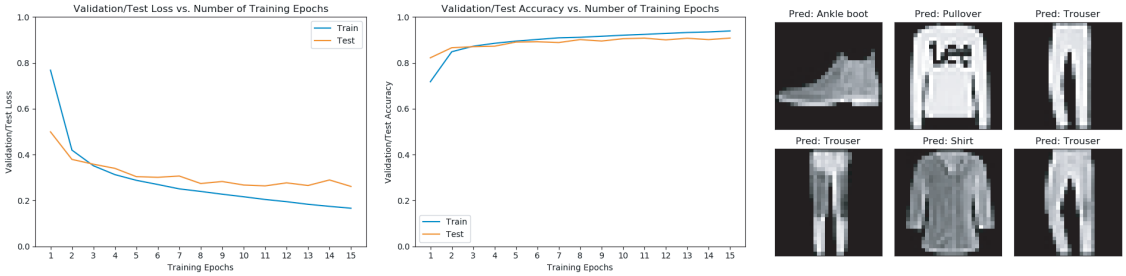

Figure 3: Example plots of one possible NN architecture. (a) Accuracy of classification results over increasing number of epochs. (b) Error of classification results over increasing number of epochs. (c) Examples of correct classifications.

To do this, please implement different Neural Network Model classes in *cnn.py* **according to the default architecture in Figure 4** and create at least two additional models (neural networks) with a different architecture inspired by architectures discussed in the lecture (e.g. LeNet, AlexNet, VGG-16). **Note:** Training CNNs can take a long time. Therefore start by training the network with only a few

epochs to be able to estimate whether the training will work flawlessly. Later they should train between 10-50 epochs. **Note:** Please use the *Cross-Entropy* cost function for the model (softmax).

You can change the default implementation as you like, if you find that easier. A good starting point is https://pytorch.org/tutorials/beginner/finetuning_torchvision_models_tutorial.html#model-training-and-validation-code which I used for the provided source code.

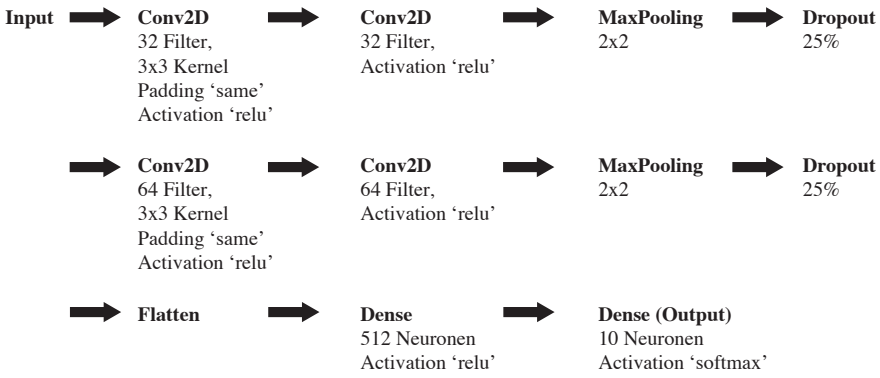

Figure 4: Architecture to be implemented.

Please hand in your result plots for all three architectures you implemented and your source code.

Exercise 4: Readings (voluntary)

Please read the publications in the reading area (Moodle).

Due: This is *Exercise 3/4*. The tasks are designed to be solvable in 3 weeks. **Information on due date and a link to upload your solution as a .zip file is given in the Moodle system**. Please submit only one single .zip file with the sources of your solution. **Please insert all necessary images so that each task is directly executable. Otherwise I reserve point deductions.**. Submission after due date will be deducted with 5 points for each delayed week.