**Deep Learning Exercise Contents**

**Exercise 1 Perceptron learning**

* Data sample generation, labelling and separation
* Perceptron activation and sign function
* Perceptron learning and weight initialization

**Exercise 2 Gradient descent (GD)**

* Gradient computation (by hand) and loss function implementation
* Gradient implementation and termination criteria
* GD implementation and finding minimas

**Exercise 3 Universal function approximator: two-layer fully connected (FC) network with GD**

* Network, gradient, GD implementation
* Dataset generation and weight initialization with lots of criteria

**Exercise 4 Multi-output networks and batch processing: three target values**

* Input data standardization and batch processing
* Network, loss, gradient implementation with iterative GD

**Exercise 5 Classification in PyTorch: stochastic gradient descent (SGD)**

* Dataset generation and splitting with Tensors
* Input data standardization and network implementation with torch (sequential)
* Accuracy computation (with torch) and torch SGD training
* Binary classification (BCE) and categorical classification (CrossEntropyLoss)

**Exercise 6 Convolutional networks with CUDA**

* MNIST dataset instantiation
* Three-layer FC network, convolutional output (by hand)
* Convolutional network with validation and accuracy, training with CUDA 🡪 .to(device)!!!
* FC and convolutional training, learnable parameters computation (by hand)

**Exercise 7 Transfer learning with pre-trained networks**

* CIFAR-10 dataset transformation and loading
* ResNet-50 pre-trained network, network by extracting & replacing logits from last FC layer
* Accuracy computation (alternate variant from exercise 5) and torch SGD training
* Network fine-tuning with and without freezing layers, confusion matrix plotting

**Exercise 8 Open-set classification with known/unknown samples**

* Instantiation and vectors for known/negative/unknown samples
* Dataset constructor 🡪 \_\_init\_\_ , super() , \_\_getitem\_\_ , \_\_len\_\_
* Batch splitting into known/unknown
* AdaptedSoftMax loss function, static forward and static backward function
* SoftMax confidence evaluation 🡪 check comments for better solution
* Network constructor and forward function with ReLu activation 🡪 topology of network!
* Network training with AdaptedSoftMax and training/validation set instantiation

**Exercise 9 Convolutional auto-encoder and denoising images**

* Dataset instantiation, random noise generation 🡪 torch.clamp
* Encoder and decoder network for deep features 🡪torch.deconv , torch.sigmoid
* Joint auto-encoder network (encoder+decoder) and instantiation, training with noise, Adam optimizer and MSELoss

**Exercise 10 Learn to write like Shakespeare with a recurrent network**

* Unique character list instantiation from text, one-hot dictionary for each character
* Sequence function for inputs and targets, dataset implementation (including sequence)
* Elman network including three FC layers (W1, WR, W2) and PReLU activation, logit sequence
* Elman network training including loss permutation 🡪 z.permute(*indexes of dimension order*)
* Text sequence encoding and next element prediction (SoftMax) using 🡪 argmax() or random
* Sequence completion and text production

**Exercise 11 Adversarial training using fast gradient sign (FGS) and fast gradient value (FGV)**

* Dataset and network instantiation (similar to exercise 8)
* Adversarial sample computation using FGS & FGV, random noise (slightly different than ex9)
* Network training with additional sample (FGS and FGV)
* Validation & classification accuracy, generate adversarial samples, compute original accuracy
* Networks training three ways: clean, noise and FGS and their accuracies

**Exercise 12 Radial basis function (RBF) network**

* Dataset instantiation, RBF layer instantiation 🡪 \_\_init\_\_ , super , forward function
* RBFActivation and RBFNetwork (similar to ex8)
* Network training and validation, deep feature extraction in 2D and plotting