



Machine Learning

Project 1

Alexander Gepperth, December 2025



Project description

- Duration: 2-3 weeks
- Overall goal: train a classifier on data I give you!
- Data: .zip with png images, label is encoded in file name
- Images/data are corrupted → visualize, explore, remove problems!
- Expected results: CNN classifier that performs well on a holdout dataset that I will provide



Strategy

- Data exploration, visualization and correction: use numpy & matplotlib
- DNN/CNN training: use tensorflow/keras



Data

- Please download data (tomorrow!):
`wget www.gepperth.net/alexander/downloads/data1.zip`
- Decompress the file to you local home directory using , e.g., `unzip`
- This is training data only! In order to train a classifier, you will have to perform a train-test split (later)!



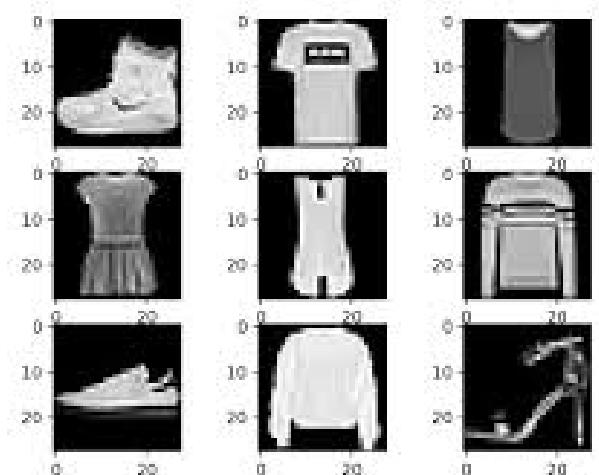
Project description

- Create two python scripts
 - convert_data.py: reads image files, corrects problems, stores data to a .npz file for faster processing
 - cnn.py: train/test split, trains or evaluates a model on data
 - Invocations:
 - `python3 convert_data.py <image_dir> h w c <npz>`
 - `python3 cnn.py <npz> train|test`
- If parameter 2 is “train” → train DNN on images, save weights,
otherwise: load weights, compute accuracy on provided image data
only, displays confusion matrix
- correct problems?
- 0 / 1



Datasets

- Number of classes: find out!
- Datasets: FashionMNIST
 - 28x28 mono
 - 70.000 images
 - expected accuracy: $\geq 88\%$





Strategy for convert_data.py

- Read images into numpy, extract label from file names. Take image path and size (W/H/C) from command line (using `sys.argv`)
- Data analysis and preprocessing, remove problems from data **in a generic fashion!**



Strategy for `cnn.py`

- (Once data has been read from numpy array): split into train/test like 80/20
- Train a simple DNN (dense layers), optimize parameters w.r.t. performance
- Then: train a CNN with structure similar to LeNet5, optimize parameters



Useful functions and packages (look them up!)

- Access to command line arguments: `sys.argv`
- Path name manipulations and directory listing:
`os.listdir`, `os.path.join`
- Image reading routines from `PIL.Image`
- Image to array conversion: `np.array`, `np.asarray`
- `string.split()`
- `np.concatenate`, `np.random.choice`
- Loading and saving arrays: `np.save`, `np.savetxt`, `np.load`
- `argparse` for processing command line arguments (optional)