Hochschule Fulda
University of Applied Sciences

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

  `wget www.gepperth.net/alexander/downloads/data2.zip`

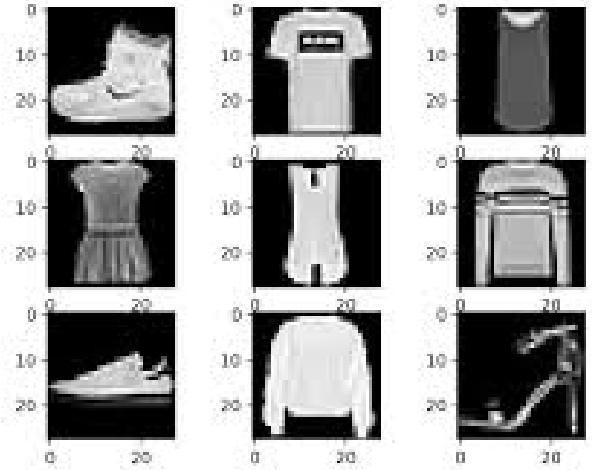
Project description

- 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> 0/1`
  `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?

Datasets

- Number of classes: find out!
- Datasets: FashionMNIST
  - 28x28 mono
  - 70.000 images
  - expected accuracy: >= 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.savez, np.load`
- `argparse` for processing command line arguments (optional)

Project workflow: training

- `python3 convert_data.py fashion_mnist/ 28 28 1 train.npz 1`
  - unzip data
  - read/correct data
  - write X,T to `train.npz`
- `python3 cnn.py train.npz train`
  - load `train.npz`
  - 80/20 train/test split
  - train on train data
  - evaluate on test data (acc, CM)
  - store model weights to file

Project workflow: testing on clean data (provided)

- `python3 convert_data.py fm_test/ 28 28 1 test.npz 0`
  - unzip data
  - read data, **do not correct**
  - write X,T to `test.npz`
- `python3 cnn.py test.npz test`
  - load `test.npz`
  - load model weights
  - evaluate on test data (acc, CM)