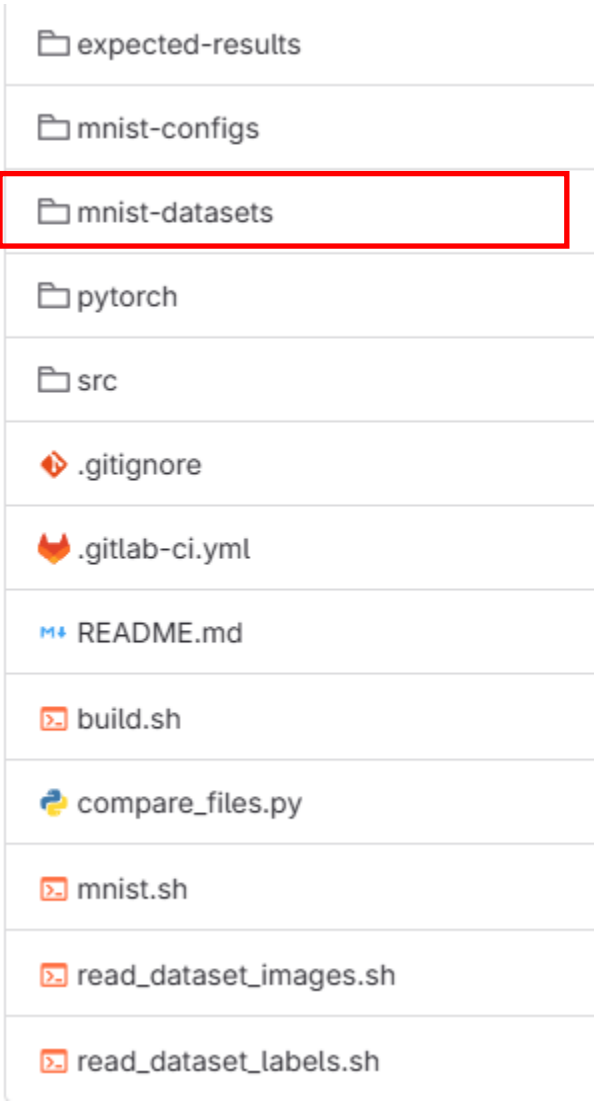
AdvPT Project

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

TECHNISCHE FAKULTÄT

The Starting Point

The template repository you are provided:

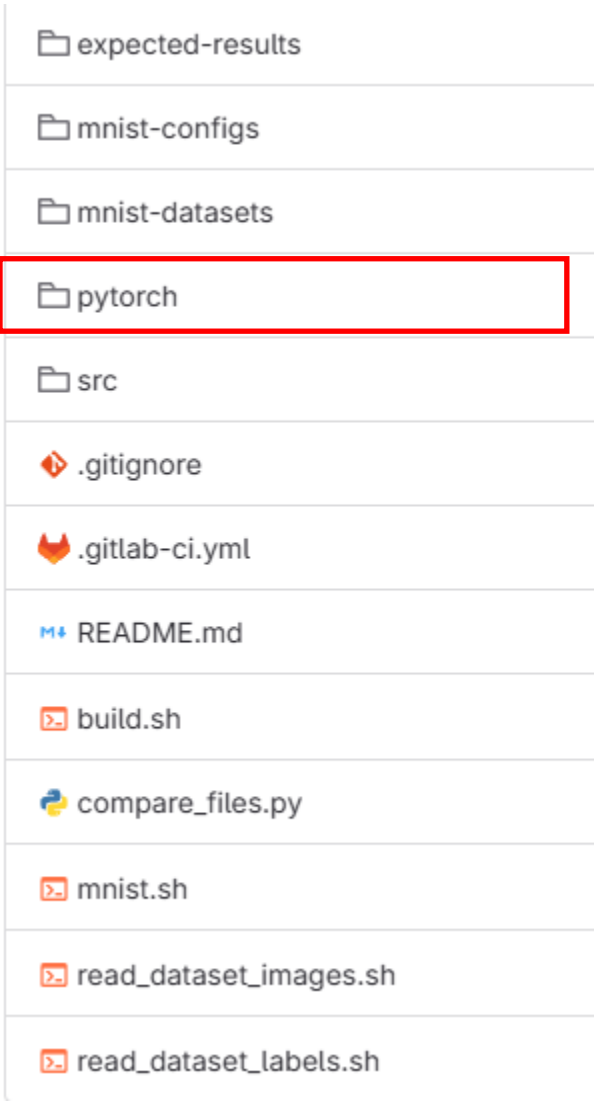


Binary files with MNIST dataset:

- *train-** files: 60000 training images + labels
- *t10k-** files: 10000 testing images + labels
- *single-** files: 1 training & testing image + label

The Starting Point

The template repository you are provided:

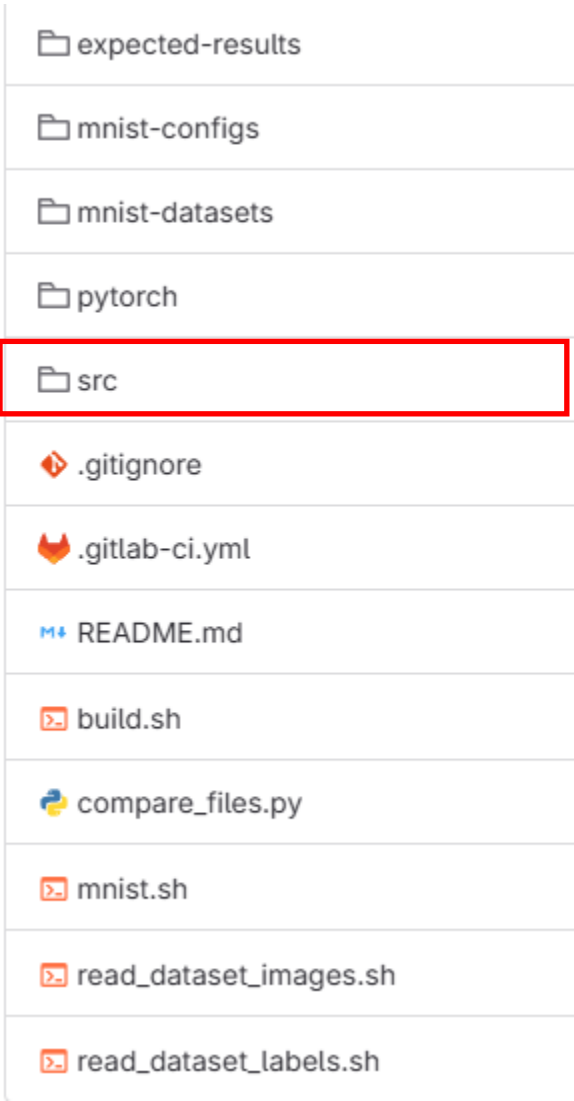


PyTorch reference implementation:

- Same NN architecture as in the project
- Helpful for initial understanding of NN workflow

The Starting Point

The template repository you are provided:



Implementation of tensor class and mat-vec product:

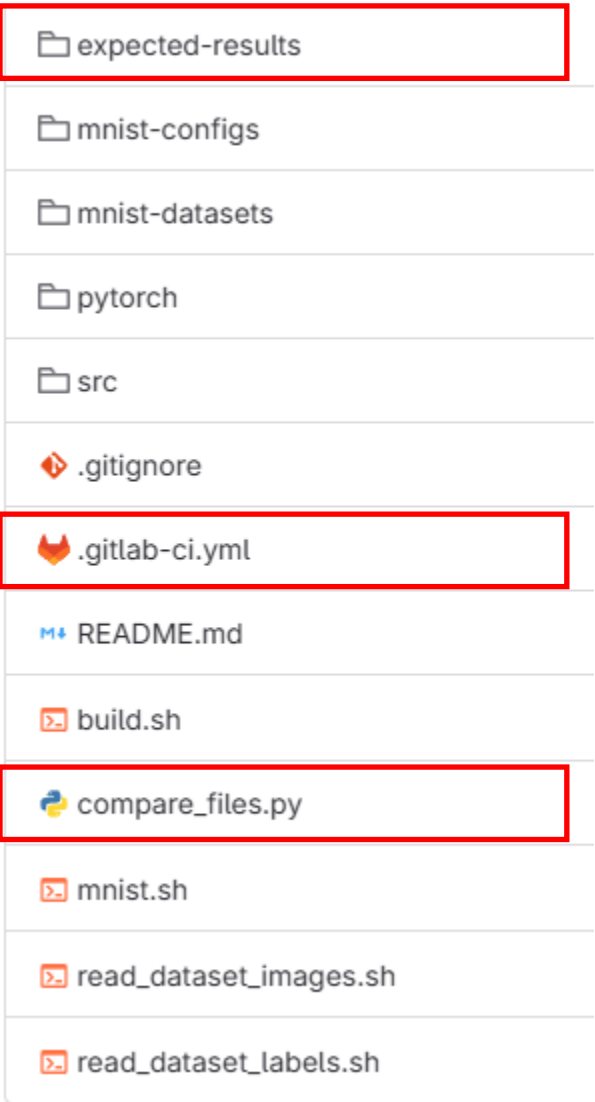
- Implementation is **very slow**
  - Profile with e.g. **gprof** to find bottleneck
  - Figure out **optimization** techniques

Use of this tensor class is **not mandatory**!
Other options are:

- Selfmade data structures, e.g. your Tensor variant
- Data structures from C/C++ libraries, e.g. Eigen

The Starting Point

The template repository you are provided:



Configuration file for your own CI pipeline:

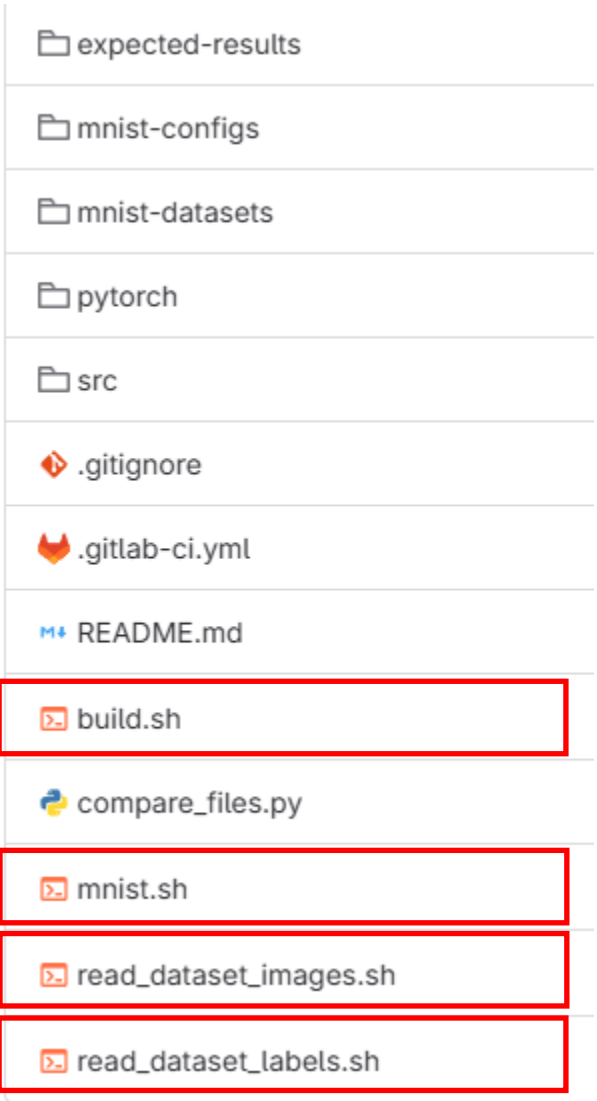
- Builds and runs your code after each commit
- Compares files output by your code with `expected-results` via `compare_files.py`
- Skipable via `[skip ci]` in commit message
- Marked files should not be adapted

This is different from the evaluation pipeline:

- CI tests are visible to you, evaluation tests are not
- Evaluation is run in regular interval (hourly up to daily)

The Starting Point

The template repository you are provided:



Scripts used by your CI and the evaluation pipeline.
Need to be extended for successful testing!

`build.sh`: Triggers build pipeline for your code

- Can be done via CMake or Makefiles
- Produces executables used by next scripts

`read_dataset_images.sh`: runs code for Task 1a)

`read_dataset_labels.sh`: runs code for Task 1b)

`mnist.sh`: runs code for model evaluation in Task 2a)

Testing is fully automated
➔ Make sure to follow the signature of the input arguments for all scripts as described in the sheet.

Task I: I/O for Neural Networks

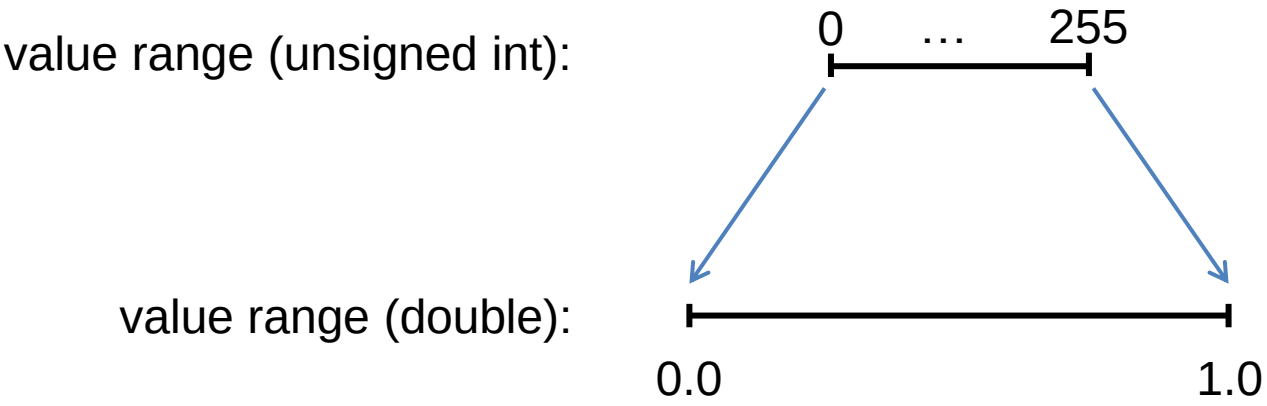
Task 1a) I/O for MNIST dataset images

https://yann.lecun.com/exdb/mnist/

[offset]	[type]	[value]	[description]
0000	32 bit integer	0x00000803(2051)	magic number
0004	32 bit integer	60000	number of images
0008	32 bit integer	28	number of rows
0012	32 bit integer	28	number of columns
0016	unsigned byte	??	pixel
0017	unsigned byte	??	pixel
.....			
xxxx	unsigned byte	??	pixel

3 Subtasks:

- Reading of MNIST image data
  - Can be done with standard library
  - Each dataset item has an image with 28x28 pixels (row-wise storage)
  - **Careful**: Binary data format and values stored in **big-endian**
  - **Write checks if values are read in correctly**
- Linear mapping of image data to FP values (double)



- Outputting the resulting Tensor<double> with 2D shape {28, 28}

Task I: I/O for Neural Networks

Task 1b) I/O for MNIST dataset labels

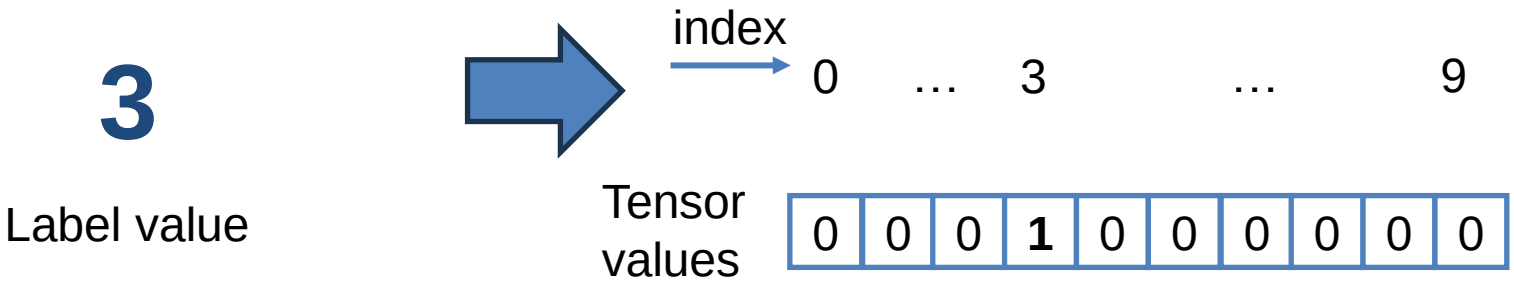
https://yann.lecun.com/exdb/mnist/

[offset]	[type]	[value]	[description]
0000	32 bit integer	0x00000801(2049)	magic number (MSB first)
0004	32 bit integer	60000	number of items
0008	unsigned byte	??	label
0009	unsigned byte	??	label
.....			
xxxx	unsigned byte	??	label

The labels values are 0 to 9.

3 Subtasks:

- Reading of MNIST label data:
  - Also big-endian encoded
  - Each dataset item has one label value
- **One-hot encoding** of label values to Tensor<double> with shape {10}, e.g.



- Outputting the resulting tensor

Thank you and good luck!

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

TECHNISCHE FAKULTÄT