MNIST Miniproject

Yan Michellod & Ralf Jandl

Toy Hypothesis

https://www.ilovephd.com/what-is-research-hypothesis/

Hypothesis:

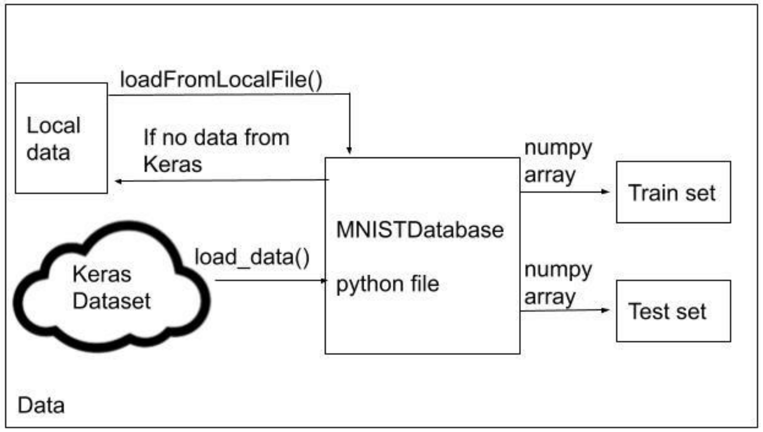
It is not possible to achieve a classification accuracy for MNIST test set higher then 95%!

Measurement:

Reject hypothesis if we can find a model which reach a better accuracy then 95%.

1 - Load MNIST Database

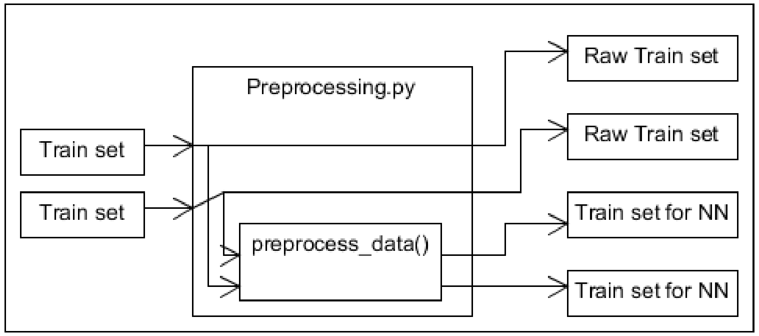
- MNISTDatabase.py
- Load from Keras
- Load from local zipped file
- Output
  - x_train: training data with shape (60000, 28, 28)
  - y_train: training digit labels with shape (60000,)
  - x_test: test data with shape (10000, 28, 28)
  - y_test: test digit labels with shape (10000,)



https://keras.io/api/datasets/mnist/

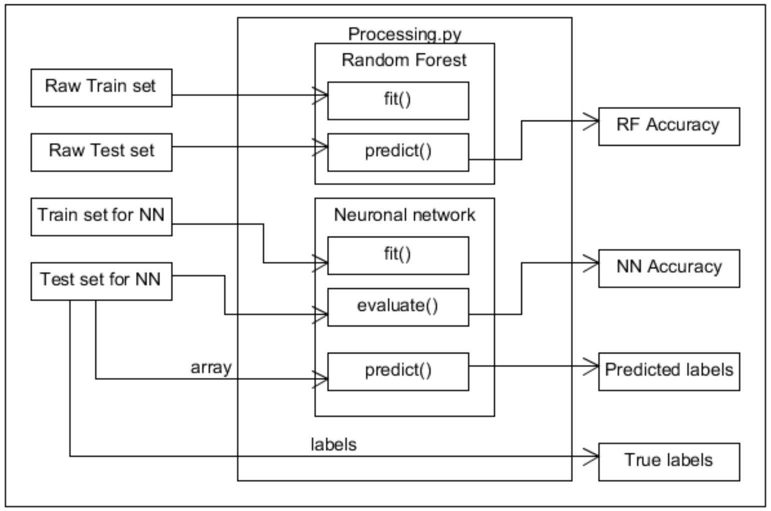
2 - Preprocessor

- Preprocessing.py
- Return original MNIST data
  - x_train, y_train, x_test, y_test
- Return preprocessed data
  - Reshape for CNN
  - Convert to categorical (One Hot)
  - Normalize
  - x_train_preprocess, y_train_preprocess, x_test_preprocess, y_test_preprocess



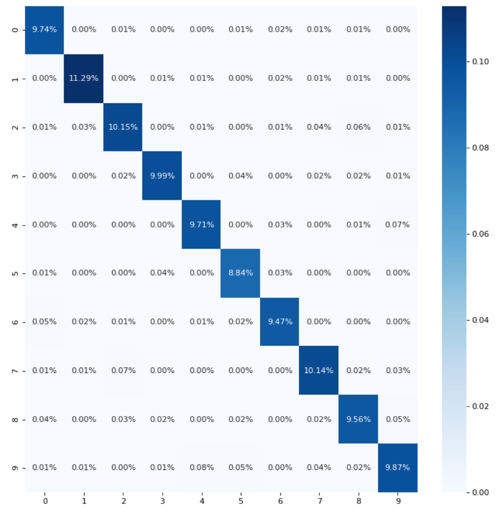
3 - Processor

- Processor.py
- Train Random Forest Model
- Train Convolutional NN
- Returns
  - Accuracy of RF
  - Accuracy of CNN
  - True and predicted labels of test set



4 - Analysis

- Analysis.py
- Check vs. Baseline: Is the CNN accuracy higher than the accuracy of the random forest?
- Check Hypothesis: Is the CNN accuracy higher then 0.95 (95%)
- Save confusion matrix

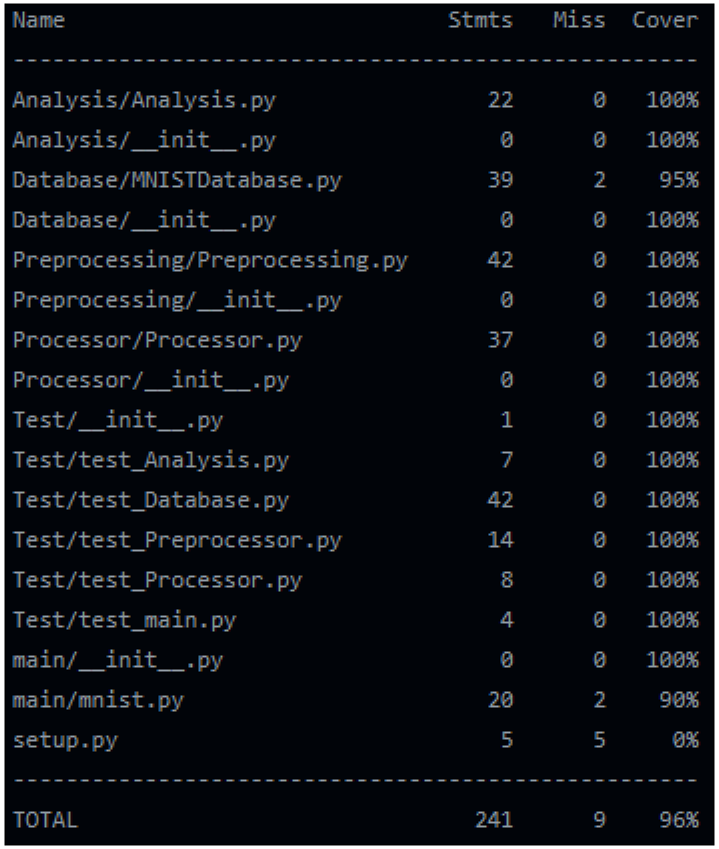


	0	1	2	3	4	5	6	7	8	9
0	9.74%	0.00%	0.01%	0.00%	0.00%	0.01%	0.02%	0.01%	0.01%	0.00%
1	0.00%	11.29%	0.00%	0.01%	0.01%	0.00%	0.02%	0.01%	0.01%	0.00%
2	0.01%	0.03%	10.15%	0.00%	0.01%	0.00%	0.01%	0.04%	0.06%	0.01%
3	0.00%	0.00%	0.02%	9.99%	0.00%	0.04%	0.00%	0.02%	0.02%	0.01%
4	0.00%	0.00%	0.00%	0.00%	9.71%	0.00%	0.03%	0.00%	0.01%	0.07%
5	0.01%	0.00%	0.00%	0.04%	0.00%	8.84%	0.03%	0.00%	0.00%	0.00%
6	0.05%	0.02%	0.01%	0.00%	0.01%	0.02%	9.47%	0.00%	0.00%	0.00%
7	0.01%	0.01%	0.07%	0.00%	0.00%	0.00%	0.00%	10.14%	0.02%	0.03%
8	0.04%	0.00%	0.03%	0.02%	0.00%	0.02%	0.00%	0.02%	9.56%	0.05%
9	0.01%	0.01%	0.00%	0.01%	0.08%	0.05%	0.00%	0.04%	0.02%	9.87%

Unit tests and coverage

- Unit testing : pytest
- Test file for each module
- Tests to cover all possibilities
- Coverage: coveralls.io

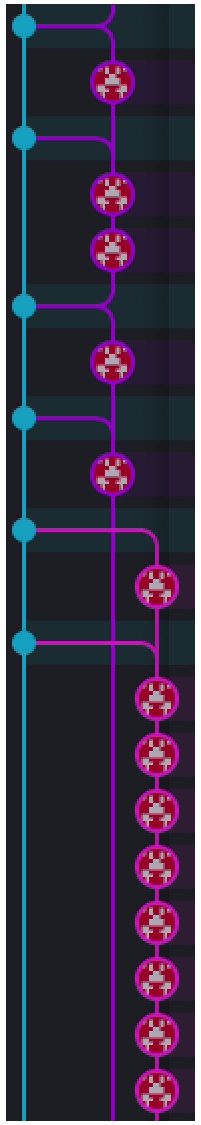
coverage 95%



```
Name                               Stmts   Miss  Cover
------------------------------------------------------
Analysis/Analysis.py                  22      0   100%
Analysis/__init__.py                   0      0   100%
Database/MNISTDatabase.py             39      2    95%
Database/__init__.py                   0      0   100%
Preprocessing/Preprocessing.py        42      0   100%
Preprocessing/__init__.py              0      0   100%
Processor/Processor.py                37      0   100%
Processor/__init__.py                  0      0   100%
Test/__init__.py                       1      0   100%
Test/test_Analysis.py                  7      0   100%
Test/test_Database.py                 42      0   100%
Test/test_Preprocessor.py             14      0   100%
Test/test_Processor.py                 8      0   100%
Test/test_main.py                      4      0   100%
main/__init__.py                       0      0   100%
main/mnist.py                         20      2    90%
setup.py                               5      5     0%
------------------------------------------------------
TOTAL                                241      9    96%
```

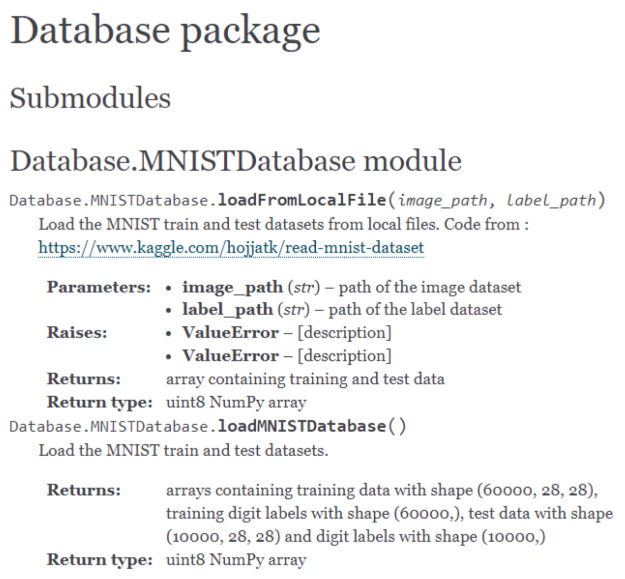
Git flow / Versioning

- Branches :
- Database => implement and test data access
- Preprocessor => implement and test data preprocessoring
- Processor => implement and test ML algorithm
- Analyzer => Analysing the accuracy
- Continuous_Integration => Branch used for testing CI before integrate it into each other branch
- And some temporary branches (f.e. Packaging)
- **CI : github actions**
  - Tests + Coveralls
  - Documentation



Documentation

- Pre-commit : black, isort and interrogate
- Documentation : sphinx



Database package

Submodules

Database.MNISTDatabase module

`Database.MNISTDatabase.loadFromLocalFile(image_path, label_path)`

Load the MNIST train and test datasets from local files. Code from : https://www.kaggle.com/hojjatk/read-mnist-dataset

Parameters:
- **image_path** (*str*) – path of the image dataset
- **label_path** (*str*) – path of the label dataset

Raises:
- **ValueError** – [description]
- **ValueError** – [description]

Returns: array containing training and test data

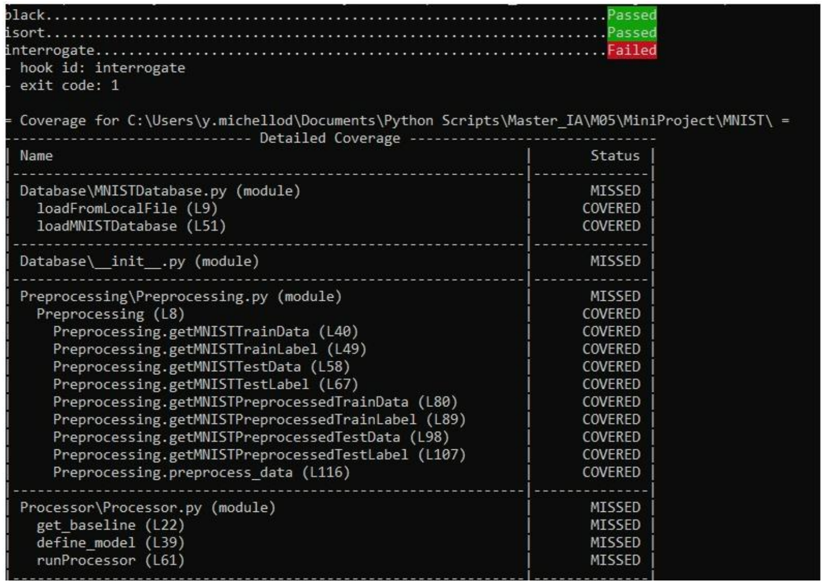
Return type: uint8 NumPy array

`Database.MNISTDatabase.loadMNISTDatabase()`

Load the MNIST train and test datasets.

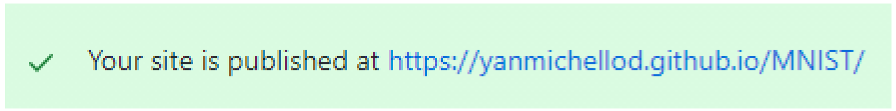
Returns: arrays containing training data with shape (60000, 28, 28), training digit labels with shape (60000,), test data with shape (10000, 28, 28) and digit labels with shape (10000,)

Return type: uint8 NumPy array



```
black...................................................................Passed
isort...................................................................Passed
interrogate.............................................................Failed
- hook id: interrogate
- exit code: 1

= Coverage for C:\Users\y.michellod\Documents\Python Scripts\Master_IA\M05\MiniProject\MNIST\ =
------------------------------ Detailed Coverage -------------------------------
| Name                                                              |    Status |
|-------------------------------------------------------------------|-----------|
| Database\MNISTDatabase.py (module)                                |    MISSED |
|   loadFromLocalFile (L9)                                          |   COVERED |
|   loadMNISTDatabase (L51)                                         |   COVERED |
|-------------------------------------------------------------------|-----------|
| Database\__init__.py (module)                                     |    MISSED |
|-------------------------------------------------------------------|-----------|
| Preprocessing\Preprocessing.py (module)                           |    MISSED |
|   Preprocessing (L8)                                              |   COVERED |
|     Preprocessing.getMNISTTrainData (L40)                         |   COVERED |
|     Preprocessing.getMNISTTrainLabel (L49)                        |   COVERED |
|     Preprocessing.getMNISTTestData (L58)                          |   COVERED |
|     Preprocessing.getMNISTTestLabel (L67)                         |   COVERED |
|     Preprocessing.getMNISTPreprocessedTrainData (L80)             |   COVERED |
|     Preprocessing.getMNISTPreprocessedTrainLabel (L89)            |   COVERED |
|     Preprocessing.getMNISTPreprocessedTestData (L98)              |   COVERED |
|     Preprocessing.getMNISTPreprocessedTestLabel (L107)            |   COVERED |
|     Preprocessing.preprocess_data (L116)                          |   COVERED |
|-------------------------------------------------------------------|-----------|
| Processor\Processor.py (module)                                   |    MISSED |
|   get_baseline (L22)                                              |    MISSED |
|   define_model (L39)                                              |    MISSED |
|   runProcessor (L61)                                              |    MISSED |
|-------------------------------------------------------------------|-----------|
```



✓ Your site is published at https://yanmichellod.github.io/MNIST/

License

Lot of different libraries used

License MIT

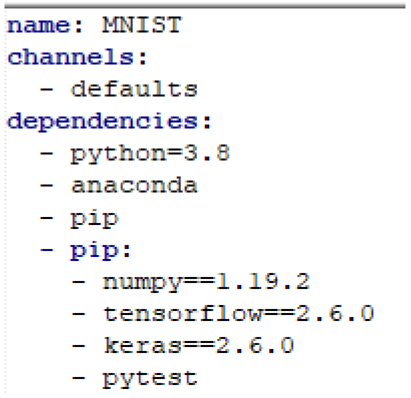
- MNIST Dataset: Yann LeCun and Corinna Cortes hold the copyright but is available under the terms of the Creative Commons Attribution-Share Alike 3.0 license (https://keras.io/api/datasets/mnist/)
- Tensorflow: Apache License 2.0 (https://github.com/tensorflow/tensorflow/blob/master/LICENSE)
- Sklearn: BSD license encouraging academic and commercial use (https://scikit-learn.org/stable/)
- Seaborn: Copyright (c) 2012-2021, Michael L. Waskom (https://github.com/mwaskom/seaborn/blob/master/LICENSE)
- Numpy: liberal BSD license (https://numpy.org/)

→All Allow to use and distribute copies of the work

Therefore we use MIT License to grant permission to copy and use our github code.

What do we have in place for reproducibility

- Create a controlled env with pinned version
- Save data
- Deployement as easy as possible:
  - **pip install**
- Automated tasks as max as possible:
  - **Pre-commit (Black, iSort and interrogate)**
  - **Tests (part of the code, env.)**
  - **Documentation**



```
name: MNIST
channels:
  - defaults
dependencies:
  - python=3.8
  - anaconda
  - pip
  - pip:
    - numpy==1.19.2
    - tensorflow==2.6.0
    - keras==2.6.0
    - pytest
```

How to deploy the project

Open the terminal with python 3

1) pip install git+https://github.com/yanMichellod/MNIST#egg=MNIST_Classification
2) Ask some help : mnist –h
3) mnist –full=True

Conclusion

Hypothesis:

It is not possible to achieve a classification accuracy for MNIST test set higher then 95%!

Rejection:

We reject the hypothesis as we achieve an accuracy of 98.76%!

Project Feedback:

The mini project enabled an optimal introduction to GitHub and its various possibilities for collaboration and reproducibility. We had meetings several times per week to share our experiences and to plan our next tasks.

Live Demo