AI pipelines in Jupyter Notebooks and Pytorch

Jupyter Notebooks have become a standard tool to showcase AI pipelines, including the most advanced models and complex software. They allow to present an AI process and code in a readable HTML text that can be visualised online, help dissecting its parts, visualise graph and results, and provide explanations. For these reasons, a successful computer science graduate with AI knowledge will be proficient with the use of such a tool to learn from and demonstrate AI pipelines.

In this CW, you will prepare a tutorial for a machine learning pipeline in Pytorch. PyTorch is an open-source machine learning library developed by Facebook's AI Research lab (FAIR). It is widely used for applications such as computer vision and natural language processing. PyTorch is known for its flexibility, ease of use, and as a deep learning research platform.

- Weight: 30% of the total mark
- Expected workload: approximately 30h for a student with no previous knowledge of Jupyter Notebook.
- Note: if this is a resubmission, please ensure that it is significantly different from your previous submission, i.e., at least 30% different.

Objective

You will prepare a tutorial in which you showcase the Jupyter Notebook capabilities in demonstrating a machine learning pipeline of your choice. You are encouraged to look up for online tutorials to use as examples, but you need to develop your own pipeline. The final aim is to create an example of developing an AI system that can be showcased online and can be used as part of your portfolio in job applications.

List of requirements

The tutorial has the following requirements:

1. A clear title that illustrates the topic of the tutorial.
2. An abstract of 80 to 100 words that summarises the content.
3. A list of learning objectives that will be achieved by following this tutorial.
4. A table of content.
5. Differences with pros and cons in relation to cited sources of similar online tutorials.
6. At least the following libraries: Pytorch, Matplotlib, Numpy.
7. Use of markup.
8. Clear identification of sources to avoid plagiarism.
9. Use one dataset, e.g., the MNIST
10. A training process for a machine learning algorithm, e.g., the optimization of a neural model.
11. How different configurations of hyper-parameters affects the results.
12. Results with both graphs and tables.

13. A list of references at the end.
14. A reading time of 10 to 15 minutes.

Additional notes

Rapid development in AI are mainly due to code reuse and the integration of complex process into Python libraries. You are encouraged to exploit libraries to achieve your objectives, e.g., TorchVision/Text/Audio, Hugging Face's Transformers, FastAI, PyTorch-RL, StableBaseline3, and many others.

You are allowed to reuse knowledge and topics that you might have developed in other context, e.g., as part of your final year project, as a personal project, or as part of your placement work. In all cases, make sure that you do not plagiarise previous work and respect all intellectual property.

Marking scheme

Marks will be awarded according to the following criteria:

- **Completeness**: Does the tutorial cover all required components (e.g., abstract, objectives, dataset, training process, results, references, etc.)? [25%]
- **Clarity and code quality**: Are the explanations logically structured and easy to follow? Do they enhance the learning experience? Is the Jupyter Notebook code well-structured, efficient, and properly documented? Does it execute correctly? [25%]
- **Presentation quality and professional elements:** : Are graphs, tables, and markdown explanations clear and informative? Do they enhance the tutorial's readability? [25%]
- **Originality and critical thinking**: Does the tutorial show a unique perspective, creativity, or novel insights beyond existing tutorials? Does it critically compare different techniques? [25%]

Submission

Submit on Learn a zip file that contains the Jupyter Notebook file (jpynp) and an executed version of it exported as HTML.

Examples of titles for tutorials

Here are examples of titles that could be suitable for this coursework

- A comparison of the performance and execution time of simple deep learning networks on the MNIST
- Implementing and visualizing a simulated annealing process
- Implementing and visualizing particle swarm optimization
- Learning to play a simple game with neural networks and reinforcement learning
- How to train a GAN on your data set to generate your personalized images
- How to interpret climate change data with a predictive machine learning tool

- How to train a neural network to recognise different vocal commands

Important note on plagiarism and collusion

The University uses an automated software that compares submitted coursework with:

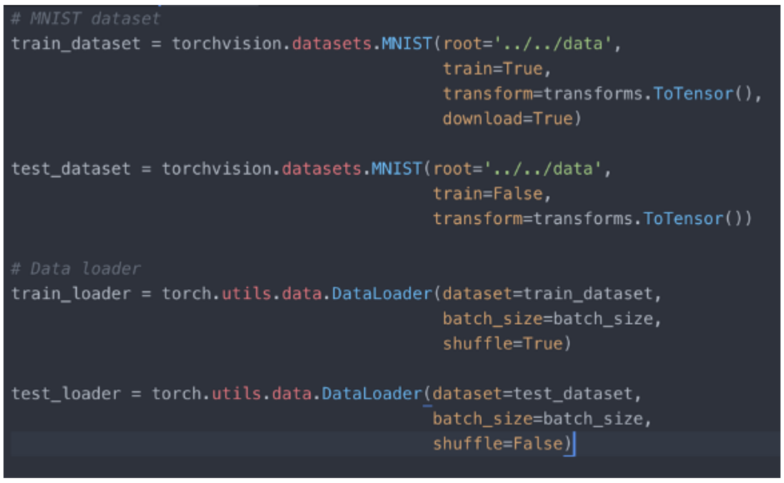
1. All coursework submissions by all students (e.g. it finds if two or more courseworks are too similar)
2. Other materials online including webpages, tutorials, scientific papers, and more.

After submission, the module leader receives a report with similarity percentages and automated and precise identification of similar parts. Some similarity is expected, e.g., in code where import of libraries are required and predefined functions. However, if similarities are found in the original implementation, structure of the code, the module leader is obliged to file a case of suspected academic misconduct, which in the worst case could result in the termination of studies.

How to avoid plagiarism/collusion:

- Do not share your code or any part of it
- Do not ask to see someone's else code, even if you are late or struggling to complete. Consider emailing the module leader or consider a mitigating circumstance claim if prevented to progress by external factors or illness.
- Do not work closely together ending up writing the same code line by line.
- Use citations correctly, both in the report and in the code when copying and pasting lines from an external source.

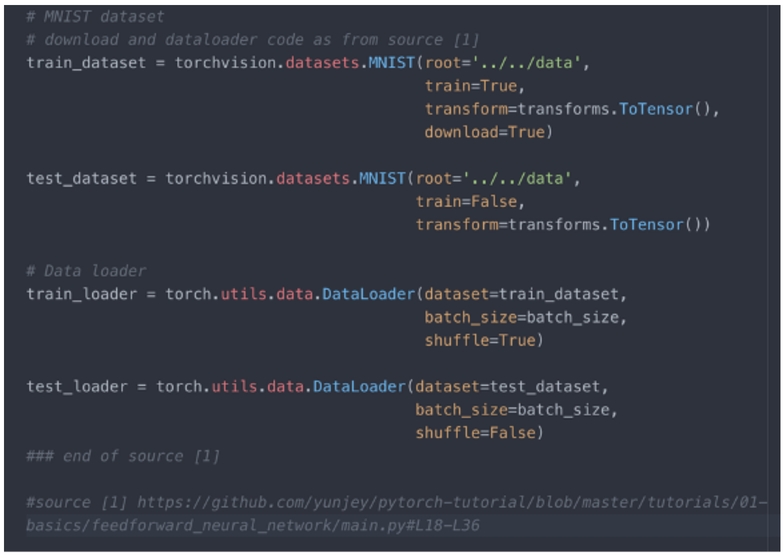
EXAMPLE: Case of potential plagiarism:



```
# MNIST dataset
train_dataset = torchvision.datasets.MNIST(root='../../data',
                                           train=True,
                                           transform=transforms.ToTensor(),
                                           download=True)

test_dataset = torchvision.datasets.MNIST(root='../../data',
                                          train=False,
                                          transform=transforms.ToTensor())

# Data loader
train_loader = torch.utils.data.DataLoader(dataset=train_dataset,
                                           batch_size=batch_size,
                                           shuffle=True)

test_loader = torch.utils.data.DataLoader(dataset=test_dataset,
                                          batch_size=batch_size,
                                          shuffle=False)
```

Figure 1: Potential plagiarism.

Correct way to do it:



```python
# MNIST dataset
# download and dataloader code as from source [1]
train_dataset = torchvision.datasets.MNIST(root='../../data',
                                           train=True,
                                           transform=transforms.ToTensor(),
                                           download=True)

test_dataset = torchvision.datasets.MNIST(root='../../data',
                                          train=False,
                                          transform=transforms.ToTensor())

# Data loader
train_loader = torch.utils.data.DataLoader(dataset=train_dataset,
                                           batch_size=batch_size,
                                           shuffle=True)

test_loader = torch.utils.data.DataLoader(dataset=test_dataset,
                                          batch_size=batch_size,
                                          shuffle=False)
### end of source [1]

#source [1] https://github.com/yunjey/pytorch-tutorial/blob/master/tutorials/01-
basics/feedforward_neural_network/main.py#L18-L36
```

Figure 2: Correct use of referencing.

- Note 1: We know that code can often look very similar if not identical for certain parts, e.g. the dataLoader or the specification of a network architecture. Therefore, if you cite the sources correctly, use comments.
- Note 2: While some similarity is expected in the first blocks of the Jupyter Notebook, we do not expect significant similarities in the presentation of the results and in particular in the report.