Milestone 1

All the steps necessary to solve the following tasks (up to task 6) need to be added to your report. Keep track of what you did. Especially the errors and problems you encountered and how you dealt with them. The report should not be longer than 4000 words (roughly 8 pages). It does not need to be a scientific paper. However, make sure that it is well formatted.

Data set: http://yann.lecun.com/exdb/mnist/
Code: https://github.com/keras-team/keras-io/blob/master/examples/vision/mnist_convnet.py

Note: Please use the above code for this milestone. However, for future milestones, you can also work with different Deep Learning Frameworks if you like.

The goal of this project is to checkout ("download") the code and make it run on your machine (and explain what you did; so log your progress). Furthermore, you will use Git to version the code.

Important Note:
Do not commit large files to Git (for example data files larger than a couple of Megabytes). Git is meant for versioning code and text files, not large "binary" files (like .exe or .docx files). Since Git tracks the history of your changes, a large file committed to the Git repository **will stay within the history** and "bloat" the repository (it will be slow to download). As a rule of thumb, only commit .py (files with Python code) and your reports to your repository.
Read about the ".gitignore" file and how to use it in order to prevent accidental addition of large binary files to your Git history.

1) Read about the data set assigned to you. Explain what it is about and what problem is solved by the machine learning models trained on it.

- Is it a classification or regression problem?
- What are the characteristics of the data set (size, type of data, etc.)

2) Check out the code base assigned to you from GitHub

3) Commit the relevant python file to your Git Repo. Every commit needs to have a meaningful comment.
It should include the following information:
- Summary of what was done
- Why a change or commit was necessary
- More details about what was done
- More details on why it was done, as well as references to other commits

A clean Git History is important to track changes and understand why a certain change was made in the past.

Note, you are not allowed to push / commit anything directly on the main branch of your project. It is good practice to block any direct pushes to the remote git repository main branch through the GitHub settings of the project repository. The main branch should always contain a working version of your code that can be released to "clients" (that's us). Create a new feature branch instead to work on. Once the work is done, you can merge this branch back into the main branch using a "pull request", followed by a code review by other members of the team (see below).

Errors in the commit history (wrong/bad comments, or wrong files/changes committed) need to be cleaned up. Read about the git reset and revert commands. Note that you should think about what belongs to a commit. If you can't explain what you did concisely, chances are that your commit is not well structured and contains too many unrelated changes. You can change / clean commits using for example an "interactive rebase". This is a more advanced topic but feel free to read into it.

4) Run the Python code

- Explain in detail which steps were necessary to run the code. How did you install Python?
- Find out what versions are being used to run the code (Python version and all dependencies)
- Are the versions dependent on the system the code is being run on? (try running it on different machines, by checking out the code onto these machines via Git. Does it work out of the box?)

5) Explain what the code does

- What is the input to and the output from the neural network
- What is Keras? And how does it relate to Tensorflow?
- How is the data loaded
- Which dependencies are imported, what do they do?
- What kind of neural network architecture are you dealing with?
- Why do you need a neural network for this task in the first place?

6) Add a documentation file to the root folder of your GitHub project. It should explain step by step how to run your code (we will try this on our machines). It should be updated with each pull request to your main branch. Make sure it is formatted correctly using the markdown syntax (refer to the *Markdown Guide* online). The documentation should be immediately visible in the web browser when accessing your GitHub project page (note: this will only happen once you merge it into your main branch, again see below).

8) Create a folder for your report of this milestone in your GitHub project repository. Add the report and push it to GitHub. Note: As mentioned above, use the markdown syntax (like your documentation file) to write your report. Like this, you can also collaborate on your report and version it using Git as if you were writing code. Microsoft Word documents or anything similar will not be accepted.

9) Once you have a running version of your code and report, as well as adding all necessary steps to run the code to the documentation file in your feature branch (the git branch you created to work on. Again, don't directly work on the main branch!). Create a "Pull Request" using the GitHub UI to merge your feature branch into your main branch. This pull request has to be reviewed, commented and accepted by another team member. (use the GitHub Reviewing functionality).

10) Once your code, report and documentation are merged into the main branch, use the GitHub UI to create a tag/release of the commit on the main branch. Create a tag called "milestone_1". This is the tag we will check out to grade this milestone.

Tip: If you want to check if everything works correctly, try checking out the main branch from your GitHub Repository into a "fresh" Virtual Machine and try to run the code following the steps in your documentation (you can also check it out on the server we provide via SSH). Also make sure that the report is accessible.

Deadline: 21.10.2024 23:59 We will look at the timestamp at which the Pull Request is merged into the main branch

Remember: Ask questions using the appropriate Slack channels