Neural Networks (AI) Practical Project: Introduction, Instruction and Hints, Grading Criteria

Herbert Jaeger, April 25, 2020. Update 1: June 12, 2020

Outline. The practical project is an important part of this course. In this project you should install the entire pipeline of an experimental neural network project. In most cases this will be a machine learning application of MLPs or RNNs, for two reasons: these kinds of networks are introduced at the beginning of the course, and they are the entry portal to deep learning which I believe many of you find intriguing. You are however free to also define yourself a project on the basis of other types of NNs which are introduced later in the course.

I provide four topic suggestions for semester projects in the form of detailed instruction sheets which you can find on the "Hands-on miniprojects" Nestor page. These four suggestions are just that, suggestions. Whether you go for one of them, or modify one of them, or define your own theme entirely – it's up to you. In any case, make sure that at the outset you clearly state *goals* (*"objectives"*) for your project.

You may use any **programming language** or toolbox you like. The standard language in the field is Phython. Thus it is a good idea to go for this language because this choice will give you access to enormous open-domain resources that you can find on the web. Matlab is a good alternative to Phython if you do not want to use ready-made toolboxes but program your code from scratch – Matlab is a commercial product and the user interface, especially the graphics functionalities, are more convenient to use than in Python, and productivity is higher.

Choosing your project topic

You can either pick one of the ready-made project topic suggestions that I publish on Nestor, or define your own topic.

If you opt for a ready-made theme, not much needs to be said. If you choose your own topic, here are some pieces of advice:

- Don't over-reach. You only have a short time for your project. All that is expected for a 100% grade is that you demonstrate that you can define, design, implement, run, evaluate and discuss a machine learning / neural network processing pipeline aiming at some interesting project objective, starting from a clear statement of the task that you want to solve. This task should be realistically doable in a few weeks by 4 group members working together and sharing their competences.
- NN research is a dynamical, open-ended, diverse field and there are many intriguing "real research" questions that could be made the topic of a semester project. However, go for an "original research" project only if your group is composed of 4 partners who already know a lot about machine learning / neural networks and if you have a clear project strategy right at the beginning. For the great majority of course participants I would strongly advise not to think of this semester project as a "research" project, but as a "lab job" project – carrying out a mundane modeling or model comparison task of the kind that one is likely to get assigned when working in a company. The most common real-world bread-and-butter type of task for which neural networks are used is *pattern classification* (static or temporal patterns), a supervised learning task. Thus a most standard real-worldish project would be defined like this:
    - Find an application domain which your group finds interesting (Finance? Robotics? Astronomy? Social networks? Medical? Industrial process engineering? Environmental?)
    - Determine a relevant classification task in that domain. Consider some options for reasonably looking classification tasks and search the web for existing datasets that you could use. Hint: the blog entry https://towardsdatascience.com/top-sources-for-machine-learning-datasets-bb6d0dc3378b gives a most useful overview of online ML dataset resources. The task that

you specify must be matched by an available dataset, so this step may require an iterative search.

- Besides classification (of static or temporal patterns), timeseries *prediction* is another standard kind of supervised learning task.

Programming language and tools. You are entirely free here. Today's standard programming language for machine learning / neural networks is Python, and there are some well known toolboxes and interfaces available, which you can use as you wish. However, I must speak out a little warning here: if you use ready-made algorithms, for example from TensorFlow for MLPs or LSTMs, I expect that you understand what the respective algorithm is actually doing. Not in every detail (these professionally implemented machine learning algorithms are super complex), but in principle. If from your write-up I get the impression that you just ran a black box miracle machine without actually having insight why or how it worked, this will lead to a grade reduction.

Deliverables and grading. Please write a project report and submit this together with your reasonably commented code (or a link to your Git repository if you use that). If it is illustrative of your work, also submit supplementary materials (for instance, if you generated music in your project, some mpeg4 demo files). I will give rather detailed feedback on your report. I will base my grading only on the report by default, and will inspect the code only in cases of doubt.

A reasonable size for the report is 5-7 pages, 10 pt. font, not counting graphics, title page and references. Together with title page, graphics (use them a lot!) and references you will end up in a typical range of 10-15 pages.

The main grading criterion is the technical quality of your experimental work and the clarity of your technical writing, *not* the accuracy or otherwise measurable performance level of your final results. A report which demonstrates a clear experimental strategy with a transparent formal specification of your data, processing steps, optimization routine and quality assessment method can get a perfect grade even if the final results are disappointing. Conversely, super results that come out of work that is poorly documented or with methods that are apparently hardly understood by the authors will earn a poor grade.

Not using Latex gives a grade reduction of 0.5 pts (on the RUG scale from 1-10).

A compilation of essential advice for good technical writing is in Appendix A.

The Excel sheet that I use for grading is appended at the end of this document (Appendix B).

The submission deadlines are given below.

Don't stress yourselves out. I emphasize again that I do not expect or require scientific innovation or state-of-the-art performance from your project. This semester project is meant as a hands-on **exercise** where you go through the entire routine of training a NN on some interesting task with real-world data. I want you to experience all the steps from data preprocessing to deciding for a NN type and architecture to implementing it (preferably using a public toolbox) to optimizing your system by cross-validation to documenting everything in a report. If all of that is done correctly and insightfully, super. The final results need not be sparkling at all - all I want to see is a systematic walk-through through the routines.

Thus, it is good enough for a perfect grade if you choose a simple dataset and a simple NN architecture, and do a clean and complete job with those.

A general **strategy**: in a first stage, use the simplest possible version of training data that you can procure, possibly even only mock data that you produce synthetically (no real data). And use a small network (training is fast and numerically robust). All tasks that I can imagine can be done with simple MLPs. While LSTMs or other RNNs might be more elegant and powerful in some projects, MLPs are generally easier to work with. Get this simplest-possible set-up to work, including cross-validation. Then you already have something that would be good enough for the project report if carefully

documented! If then ambition and time allows, you can expand the scope of your work (richer data, wider exploration of NN types and architectures, more effort in finetuning).

It's also a good idea to split work within the group. Specifically, in many cases it is possible that one subgroup takes care of data preprocessing, while another subgroup implements the NN machinery in a basic version and tests it for technical correctness with mock data. These two lines of work can initially very well be done separately.

Nestor-based group management: On Nestor you find a subpage "Groups", in which you find a collection of ready-made groups called "Semester project workgroups. Each of the groups has been created empty, ready for self-enrollment. Find yourself together in groups of exactly four members, not more, not fewer. Once you have formed your group of 4 classmates, go to Nestor, pick the first of the preinstalled groups that is still empty, and self-enroll your group's four members. Make sure that within your group there is consensus on which of the groups you enroll before you start the self-enrollment (avoid at all costs that members of a group start enrolling themselves into different Nestor groups in an uncoordinated manner). I configured these groups with a "discussion board", "file exchange", and "email". For me in my role as course instructor, these groups are my main semester project management interface to students – in particular, this is where I see what groups actually have formed and what project topic they have chosen. You will also upload your final results in the "file exchange" section of your group.

Once you have set up your group and chosen your project theme, please write a **short abstract** (10 lines *max*) specifying your project title and the modeling task that you have chosen, and post this as a pdf file on your Nestor group in the "file exchange" section – giving me the opportunity to check whether groups have started and are pursuing themes that make sense. This abstract is only for structuring and monitoring the workflow and will not be graded.

Thus there are two **deadlines** ahead:

1. The short abstract of your chosen project, to be posted in your Nestor group: deadline **May 17 midnight**. This is a soft deadline meant for your (and my) orientation.
2. The final project report: deadline **July 7, 23:59:59 hrs**. This is a hard deadline. Every day of delay will reduce the project grade by 0.5 pts. The delay penalty is due even if the report is submitted 1 second after the deadline (Nestor timestamp counts). I do it like this because that is how it is done in real life. For instance, project proposals submitted to the EU (worth a lot of money if approved) have a deadline that is specified by the second. After that second the upload server is closed and the work spent on a belated proposal is lost. For EU projects, this is a lot of work lost – such proposals easily have 100 pages and represent the joint authorship of 10 research groups spread over Europe, collaborating for about half a year just to write that proposal. However, both the EU servers and Nestor allow file overwrite. Thus: upload pre-final versions well before the deadline to be on the safe side, and overwrite them if you can prepare better versions before the deadline has passed.

Cheers,

Appendix A: hints for report writing

1. The type of reports for semester projects.

Consider this report more of a *lab report* than a *scientific paper*. Specifically, you do not have to do a thorough literature research, which would be mandatory for a true scientific report. Your report should give a comprehensive account of the goals that you set for yourself in your project, the processing pipeline that you design and implement (if you go for a machine learning application of neural networks, this includes a description of raw and preprocessed data, neural architecture, optimization criterion (loss function) and optimization algorithm, hyperparameter optimization through cross-validation), framed by a brief introduction and a not-so-brief discussion at the end where you

summarize results, point out highlights and shortcomings of your work as well as options for future work (there will be surely many things you would have liked to do more or better in retrospect). Use mathematical formalism wherever precision in the documentation is needed. While a solid literature review on the scientific context of your project is not needed (though it wouldn't do harm), do cite all technical references that you looked up for the machine learning methodologies that you implemented.

2. Overall structure

A recommendable structuring goes like this:

1. A title page, including a list of team members
2. A separate page with an abstract (10 lines) and a table of contents
3. **Introduction**: a statement of given or self-chosen task, including a description of the dataset used. You do *not* need to write a scientific-paper style introduction with an outline of scientific background, related work or research motivation (unless you were bold enough in your group to tackle a "real research" problem for the semester project). The introduction should also specify the objectives you had for your project, i.e. what you wanted to achieve and how to measure success.
4. **Data**: Describe the dataset that you are using. The reader should get a good intuitive grasp on them – best achieved by graphical representations.
5. **Methods:** This is the main technical part of the report. You describe all steps in the processing pipeline in formal detail (mathematical specification, or pseudocode, or at the very least: clean and precise technical English). The description should be detailed enough to allow an informed reader to *replicate* your experiments. In particular this means a precise documention of your learning architecture and hyperparameter settings. A too imprecise description of methods is the most common reason for downgrading reports.
6. **Results:** Document the results of your final optimized architecture (also, if that is of interest, results of other architecture versions that you tried).
7. **Discussion:** what you learnt, why you think some things didn't work, what potential you see for improvement, … whatever flows from your heart after suffering through this project.
8. **References** list.

You can find two "role models" for how a very well-written report looks like on the Nestor subpage *Hands-on miniprojects*.

3. A quick guide for technical writing

Here is a list of Do's and Don'ts that you may find helpful as a checklist.

Using material from other papers or online sources

!!This is the most important point of all!!: *every* import of wisdom must!! be credited by giving the source. This holds for word-by-word quotes (which MUST be put into quotes), for rephrased arguments, for graphics, for data. Even when you follow a source paper/book but re-word it entirely, you MUST acknowledge the source, for instance by beginning your import of material with something like „In the account to be given below, I follow the presentation given in [XY]“. Failure to cite sources is considered as violation of the code of academic conduct and will *automatically and irrevocably* lead to a grade reduction, up to zeroing the grade if the infringement is serious.

Intellectual honesty

Only make claims and write down equations that you understand. I frequently find equations taken out the the hardly understood context of some paper, with not all symbols explained, and not well connected to the rest of the report. Bad for the grade. If you want to use a method that is difficult but for which ready-made program packages exist which you want to use – you are free to use them, but you should honestly explain the extent to which you understood them. Engineers working "out there" often use methods they don't understand. No-one blames them if the results are good: but if they are not good, it's likely that they didn't use the methods properly. If they (or you) are aware of this, and overtly describe the limitations of your understanding and the details of how you did employ the methods, no reason for blame.

Never state something without a justification

EVERY statement in scientific writing must be backed up by a justification. This holds, in particular, also for the introduction. The justification can take the form of a pointer to a reference (which is why well-written introduction sections often cite many dozens of references); or your own reasoning; or a mathematical or experimental reasoning from your own working; or (dangerous) a phrase like „… it is well-known that …<claim follows>".

English grammar and style

- If you are not firmly commanding of English grammar, pass your manuscript through the hands of a proficient speaker/writer friend. Poor English leaves the reader with an impression of lack of professionality and in real life will harm the impact of your work, or worse, make the reader quit reading. When I grade reports, bad English or more than a few typos/grammar errors is penalized by some grade reduction.
- Do not use colloquial short forms like „we're", „isn't" etc. Write them out: we are, is not, etc.
- Avoid colloquial superlatives like „huge", „enormous", "sort of", "tremendously"etc. If you want to emphasize impact/size/importance of something, express this in dry terms, best backed up by numbers and references, like „… this method is suitable for very large datasets (up to 10 GB, for example in [XY])…" instead of „… this method is suitable for gigantic datasets …"
- Avoid global judgements of the kind "the most famous...", "the most widely used..." – because you don't actually know this! Statements of this sort (expressed in proper terms) are only admissible for authors that have a long experience and a good standing in a field. In a student's report, they sound presumptuous.

Spaces before braces

I really don't know why, but an estimated half of technical writing beginners do not insert spaces before brackets. I often see something like „…as shown in earlier work[2]…" or „… seems a good argument(but …)" . Why, oh why, do so many students not insert spaces correctly: „…as shown in earlier work [2]…" or „… seems a good argument (but …"

Equations

Equations (regardless whether they are inline and non-numbered or set apart and numbered) are part of the text flow and must be surrounded by the appropriate interpunctuation. For instance, write

"… as can be seen in by the inequality

$$A < B, \qquad (12)$$

which demonstrates that A is always smaller than B."

Every variable or constant in an equation must be explained in words somewhere in the text before, or right after the equation. A typical phrasing goes like this:

"bla ...

$$f(\mathbf{x}) = 3, \qquad (13)$$

where f is any sigmoid-shaped function and $\mathbf{x}$ is the normalized network state. "

Acronyms

The full wording of any acronym that you use must be given at the first appearance of the acronym.

Math symbols

Mathematical symbols appearing in the main text must be rendered in the same mathematical font as when they appear in set-apart formulas. In Latex, write N in your latex code if you want to refer to the number N in the main text! When you want to use bold font math symbols (for instance, for matrices), write $\mathbf{X}$ etc.

References list

- use of bibtex saves you from a lot of references formatting errors
- spend much care on the reference list. It is the quality fingerprint of a technical article and is one of the first things to be inspected by professional readers and reviewers. When it is not perfectly formatted, or when it contains low-quality or irrelevant references, the reviewer will immediately have doubts about your qualification.
- use the appropriate bibitem category (article, proceedings, techreport, unpublished…)
- do not simply use the bibtex records that Google Scholar offers. They are often incomplete and often contain errors. Grab the bibliographic information from the original version of the paper, and/or open the website of the journal / conference / institution where it actually appeared to fix the details. This means some work.
- use a homogeneous formatting for all your references (e.g. use full names for all items; or use initial + family name for all items, don't mix the two)
- Capitalize names of Journals and Conferences (e.g. „Neural Computation", not „Neural computation"
- when the article title contains uppercase elements - for instance acronyms - make sure that they appear in uppercase; bibtex turns everything in titles to lowercase which can be prevented by putting those parts into {}
- Study my publications webpage or the reference list in a paper published in a good journal to see how a properly formatted reference list looks and feels.

Graphics and figures

- when creating graphics, make sure you get high-quality displays in the report. Never use low-resolution jpg or low-resolution pixel graphics. Thin lines should come out crystal-clear. The best thing to do is to use a graphics tool that outputs eps or pdf formats.
- Figures must be referenced from within the main text. For every figure there must be a sniplet in the text like „… see Figure 3 …" or „… this leads to significant improvement (Figure 3)".
- *All* symbols in a figure must be explained, either directly in the figure, or in the caption, or in the main text.
- avoid pointing to figures by „as shown in the following figure" etc. Instead, always use numbering: "Figure 2.1 shows ...". Besides tradition, a good reason for this strategy is that journal editors are prone to shift figures around on a page or across pages, destroying references by relative location.
- axes annotations must be legible-sized fonts (not microscopic!)

Numbered items

… like figures, sections, tables, must be written in Uppercase when referred to. Example: „… as can be seen in Table 3, …" (not: …in table 3).

Sectioning

When you use subsections (latex: \section{}, \subsection{}, \subsubsection{}), it is standard to insert a brief overview text or motivation after \section{}, explaining what is going to happen in the section; then start the technical contents with \subsection.

Program code

NEVER reprint original program code in scientific writing. Instead, provide *pseudocode* descriptions of the algorithmic bone of your procedures (there are a number of latex packages for nice pseudocode environments), or/and specify what your program does in abstract mathematical formalism. -- In professional academic writing it has become good practice to supply easily runnable code online on the author's homepage, so that readers can re-run the procedures from the article. Some journals also offer "supplementary online materials" associated with a published paper, where one can also deposit runnable code.

Appendix B: Grading template for final report

Evaluation form for semester project report, Neural Networks (AI)

Name of students:
Name of grader:
Date:

Fill in violet fields with grades on scale 0-10. If not applicable, put weight to zero. For bonus, fill in green field; this number will be adde For Latex use, enter a 1 in the red field when Latex was used, else 0.

	grade	weight %	total	criteria	comments
Presentation (30%)					
general appearance		3	0	general visual appearance; not over-formatted (too many fonts); meaningful, standard sectioning; nice title page; table of contents	
Figures, tables, pseudocode		7	0	good quality figs? Captions? Symbols explained in caption? Figs etc. referenced in text? Sources of imported graphics given? Legible in B/W printout? Figs informative (not redundant)? Pseudocode transparent?	
Technical writing		30	0	clear formulations? Concise formulation? Mastership of technical formulations / math formulations? Good text flow? Appropriate language (not sloppy in technical sections, not too dry in motivation sections?) Clear flow of argumentation? Contents well structured? Correct English, typos? All used external sources (literature, code) referenced? Reference list uniformly and correctly formatted?	
Latex	1.00		0	was Latex used? If yes leave the 1.00, else flip to 0.00 in the red field	
Technical quality (70%)					
Penetration of subject		20	0	Good task statement? Challenges inherent in dataset perceived and adequately addressed? Appropriate choice of model type? choice of preprocessing, feature extraction, learning algorithm motivated? Motivational examples?	
Technical work		30	0	are experiments/implementations clearly, completely, succintly described? Could experiments be checked/reproduced? results clearly, completely, succintly documented? Results connected back to starting question? Statistical basics done properly where applicable (error bars!)? Maths correct? Implementations / experiments reasonably thought out? Planned professionally (modular, efficient, transparent, documented)?	
Exhaustiveness		10	0	Is the amount of work done adequate for the given project time? Variations of methods explored?	
Bonus	0.00			extraordinary achievements, e.g. much extra work, very independent work, very difficult topic, interdisciplinary connections, original thinking. Up to 2 bonus points possible	
Raw grade			***0.00***		