TI3155TU: Deep Learning Assignment 1

Ali Alper Ataşoğlu, Koen Tuin, Elena Congeduti

November 2024 - January 2025
Delft University of Technology
Faculty of Electrical Engineering, Mathematics and Computer Science

TUDelft

Contents

1: Introduction

Throughout the course, we have explored various deep learning techniques in simplified scenarios. Now, you have the chance to tackle a real-world learning challenge and investigate neural network approaches suitable for the task. The goal is to explore how deep learning models perform on an image classification task for brain tumors. You are expected to design, train, and optimize a deep learning pipeline for this problem.

This assignment is not mandatory; however it provides good practice for the final (graded) Assignment 2. Additionally, if you submit your deliverables by the deadline, you can participate in the peer-feedback lab, where you can provide and receive feedback. Submissions will not be graded, so we encourage you to submit even if your solution is incomplete or not fully refined.

The deadline for the submission is Monday, 2nd December, at 15:30.

Please read carefully the entire document before diving into the implementation.

1.1 Deliverables

You are expected to deliver the following files:

- *report.pdf*: Use the provided template (TeX or Word) to create your report. Adhere to the specified page limit and do not modify the template sections. Submit it in PDF format.
- *code.ipynb* or *code.zip*: You can choose to submit a notebook or include all your source code in a single folder. If you select this second option, make sure to include all the `.py` files that you have used to produce your results.

Create a ZIP archive containing your report and code. Name the file as *<lastname>_assignment1.zip*. Submit your zip file through Brightspace.

1.2 Requirements

Implement your code exclusively using `PyTorch`, without using any other deep learning frameworks (e.g., TensorFlow or Keras). You may use any functionality and structure your code as you prefer. Please ensure that your source code is accessible in either `.py` or `.ipynb` format, as only accessible submissions in this format will be accepted. Partially non-accessible submissions will be penalized.
You must use the provided report template and make sure to keep your answers concise. The report must not exceed 1000 words (approximately 2 pages, excluding figures). Submissions that significantly exceed this limit will be penalized.

Submissions that do not comply with these requirements will not be assessed and will consequently receive a grade of 1.

1.3 Assessment

While this assignment will not be graded, it serves as an opportunity to familiarize yourself with the assessment criteria that will be applied in Assignment 2.
You will mainly be assessed based on your report. The code will only be checked for consistency with the information included in the report and fraud checks. Therefore, make sure that the code you provide is entirely accessible, and that the answers included in the report are consistent with the code. Any inconsistency may result in points deductions on the final grade or, in some cases, invalidation of the submission.
Please note that the performance of your model accounts for only a small portion of the grade, and the complexity of the techniques used will not be a criterion for evaluating your assignment. Instead, the focus will be on your ability to correctly design, implement, and assess the performance of at least one suitable model for the specific learning task. Precisely, the final grade of Assignment 2 will be composed by the following parts:

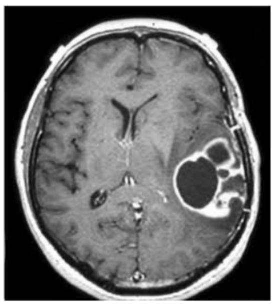
- Problem formulation (30%)

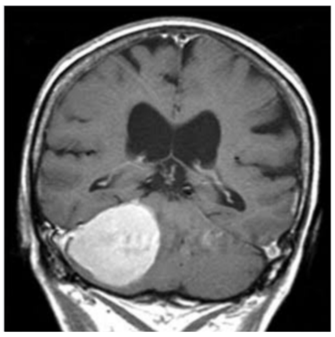
- Description of the methods (10%)
- Model choice and tuning (10%)
- Model performance (10%)
- Significance and consistency of the results (15%)
- Interpretation of the results (25%)

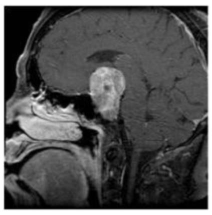
Additionally, any submission received after the deadline will be penalized. For each (rounded up) day of delay, one point will be deducted from the grade.

2: Assignment

The objective of this assignment is to design, implement, train, tune, and assess neural network-based models on an image classification task for brain tumors. Magnetic Resonance Imaging (MRI) is widely used to detect and classify brain tumors. The scanning process generates a large number of images, which are usually reviewed by radiologists. However, due to the complex nature, abnormalities in the sizes and location of brain tumors, manual analysis can sometimes result in errors. Automated deep learning techniques have consistently demonstrated great promise for these tasks.
You will be working with a dataset that you can find here, consisting of 2764 MRI images of patients with one of the three tumors: glioma, meningioma and pituitary tumor. The dataset also includes an annotation file containing the class labels for each image. A sample of images from the dataset is shown below.


Glioma


Meningioma


Pituitary

Note that there is no single correct way of solving this assignment, which is often the case in deep learning projects. You are allowed to use any network architecture, including pre-trained models. To document your methods and results, please complete the information in the provided report template.

3: Tips and Tricks

Here we give you some suggestions that might be helpful for carrying on the project. Keep in mind that those are not mandatory steps, but rather potentially useful tips.

3.1 Reporting

1. We suggest starting by defining the learning problem, inspecting the dataset and selecting which model(s) might be suitable to tackle this task. Consider checking few samples, dimensions, class balance etc. This may help you in the selection of appropriate methods.

2. Ensure that all figures in your report are self-explanatory. It can help to include a legend, axis labels, and a caption. Additionally, ensure that all the variables presented in the same plot are on the same scale and that units are consistent. If different scales or units are used, make sure to clearly highlight these differences.
3. Reflect on which metrics to use and what plots to include for interpreting your results. Ensure that the chosen metrics align with the learning task and that your conclusions are consistent with the displayed plots.
4. To assess your model, make sure that you select suitable baseline(s). These can include commonly used baselines for the specific learning task, simpler models, or state-of-the-art models. You may find inspiration from Papers with Code.

3.2 Coding

1. You are encouraged to use external accelerators such as Kaggle, which provides access to CPUs, GPUs, or even TPUs, with some limitations on free usage. If you choose any other external accelerator service, make sure that you can download your source code in either `.py` or `.ipynb` format, as only submissions in this format will be accepted.
2. We recommend consulting the PyTorch documentation and PyTorch tutorials for specific functions you intend to use.
3. If you decide to use a pre-existing network architecture, a reference to the model documentation or paper can serve as the full description of your model. Be sure to familiarize yourself with the documentation first. Many models already include integrated training loops and inference functions that you can leverage. Pay close attention to the performance metrics used, as you will be required to discuss them in your report.
4. For reproducibility, it is recommended to set the random seed(s) for any libraries you are using (e.g., PyTorch, NumPy).