Small Project
(50.039 Deep Learning, Y2021)

Matthieu De Mari, Liu Jun

1. Introduction

In this project, you will have to design a deep learning model, whose task is to assist with the diagnosis of pneumonia, for COVID and non-COVID cases, by using X-ray images of patients. We will be working on a custom dataset, which has already been curated for you, so you do not have to bother with cumbersome image processing tasks, and can instead focus on your model design and training.

For this small project, we have purposely decided to guide you through the different steps one has to take when attempting to solve such a problem. Guided notebooks have been provided to get you familiar with the dataset and the concept of custom PyTorch datasets/dataloaders.

Once you have become familiar with the dateset, you may download the full dataset online from our OneDrive repository. You will have to create your custom datasets/dataloaders and may seek inspiration from the demonstration notebook.

Then, you will have to design your custom model, following a list of requirements detailed in this document. These requirements and typical questions are guiding you on important concepts and good practices when it comes to Deep Learning projects. Please assume that this project is to be submitted to a serious client (played by us professors) and take this project as an opportunity to work on your project delivery skills. More specifically, pay attention to the way you code and document your code.

Finally, your submission will have to be uploaded for grading. Please only submit your code and notebooks, not the dataset! (Otherwise, the submission will be too large). We also advise to upload your final project on Github, to enrich your personal portfolio of projects.

As a final note, keep in mind that this small project is designed as an introduction to the Big Project, to be announced later, will require more or less the same steps/concepts, but will be a lot less guided than this small project!

2. TLDR

Groups of 2-3 people (FYI, large project will most likely be 3-5). Mixing people from both classes is allowed.

Submission deadline: 21st March 2021, 11.59pm.

3. Dataset and Dataloader

A. Creating a custom Dataset and Dataloader on a mock dataset

Your first task is to get familiar with the dataset, and the concepts of PyTorch Datasets/Dataloaders. As we are playing with a custom dataset, which cannot be retrieved automatically using torch functions, like the MNIST dataset for instance, we will have to design our own datasets/dataloaders for this problem.

Refer to the Jupyter Notebook "Creating a custom Dataset and DataLoader - Demonstration Notebook.ipynb for a demonstration of how to create your own custom dataset and Dataloader objects. The notebook operates on a simplified version of the dataset, which is stored in the dataset_demo folder.

B. Small Project Dataset

The full dataset to be used for this project, is NOT contained in the dataset_demo folder. It has to be downloaded online from our OneDrive, using this link.

It contains thousands of 150 by 150 pixels images, similar to the ones found in the mock dataset, but it has two major differences.

The first one is that it contains images with two labels: 'normal' and 'infected' as in the mock dataset. The 'infected' class, however, has been split in two subclasses, 'covid' and 'non-covid'. The 'covid' labelled images consists of the x-ray images of patients, suffering from pneumonia and diagnosed with COVID-19. The 'non-covid', however, consists of the x-ray images of patients, suffering from pneumonia and but NOT diagnosed with COVID-19.

The full dataset also contains a lot more images, namely:

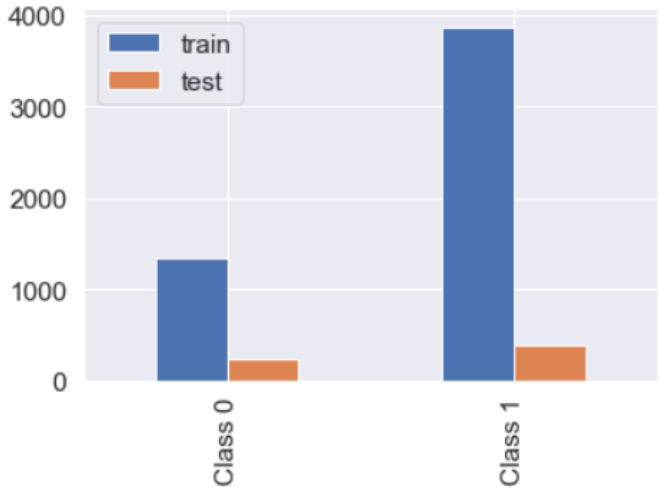
- 1341 images for the train dataset, normal class,
- 2530 images for the train dataset, infected and non-covid class,
- 1345 images for the train dataset, infected and covid class,
- 234 images for the test dataset, normal class,
- 242 images for the test dataset, infected and non-covid class,
- 138 images for the test dataset, infected and covid class,
- 8 images for the val dataset, normal class,
- 8 images for the val dataset, infected and non-covid class,
- 8 images for the val dataset, infected and covid class.

Important note: you should not work on the mock dataset used for demo.
This will result in a large penalty!

C. Expectations (Dataset and Dataloader)

Below is a list of expectations for your code and questions you should answer in your report.

- You are expected to write your own dataset and dataloader objects and may seek inspiration from the Demo Notebook discussed in Section 2.A.
- You are expected to explore the dataset slightly. Provide graphs showing the distribution of images among classes and discuss whether or not the dataset is balanced between classes, uniformly distributed, etc. Curves like the one below would be much appreciated.



- You are expected to discuss the typical data processing operations to be applied to your images. Typically, you have seen in the Demo Notebook, that our images needed some normalization was one, but why did we need it anyway? Are there other pre-processing operations that are needed for our images?

D. Bonuses

Below is a list of non-mandatory questions, which might yield bonus points if implemented and correctly discussed.

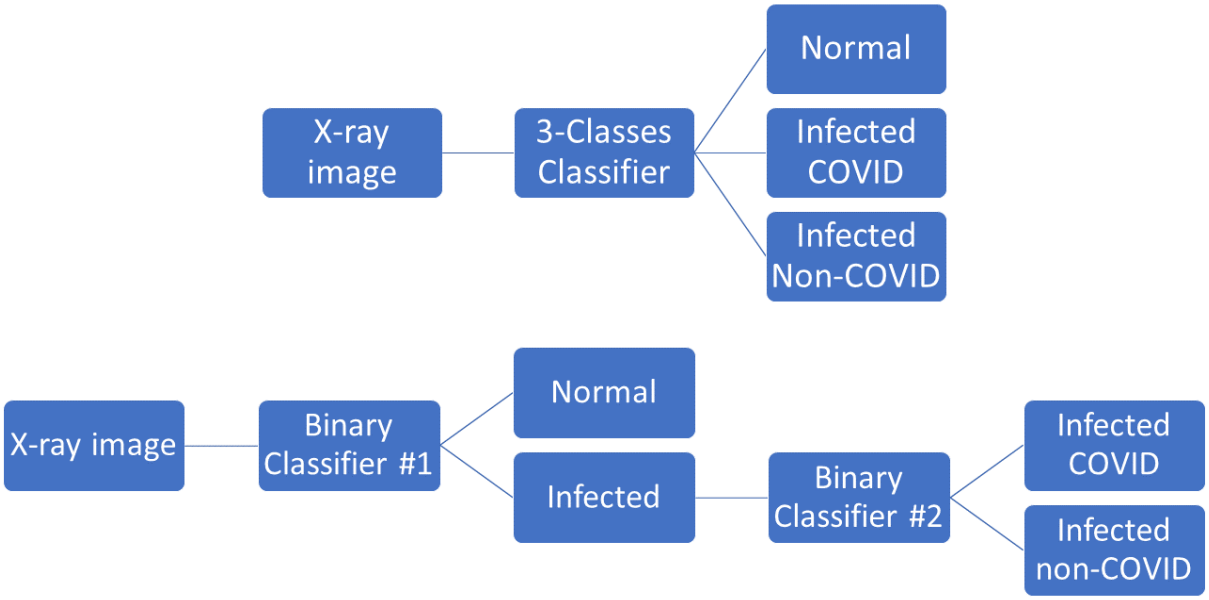
- Extra points might be given to projects who attempt to implement advanced concepts from the Data Augmentation chapter of this course. If you do so, please explain in your report the reasons that motivated you to perform data augmentation and show proof of how it benefited your model.

4. Expectations for proposed model

A. Objective

The objective of this project is to propose, train and evaluate a Deep Learning model, which attempts to classify the X-ray images of patients and help doctors with the diagnosis of COVID/non-COVID pneumonia.

A decision will have to be made, on whether to train a three-classes classifier, or two binary classifiers as shown below.



B. Expectations

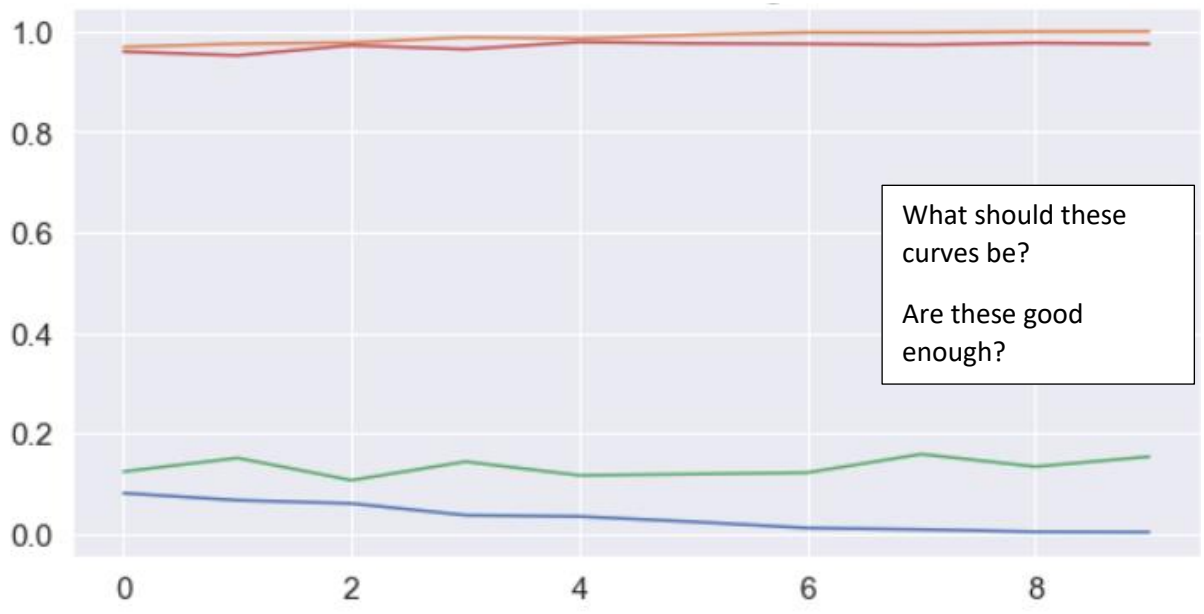
Below is a list of expectations for your proposed model.

- You are expected to discuss the differences between the two architectures above and defend which architecture you decided to go for and why.
- The model should be designed as a subclass of the nn.Module class of the PyTorch library, as shown in the Demo notebook and multiple notebooks in class. You are free to decide on the architecture (types of layers, and their number of parameters), but are expected to discuss the reasons behind this choice of architecture.
- The model should be trained from scratch, and a train function needs to be defined. You should not use pre-trained models found online. This means, no transfer learning! Doing so will result in large penalties!
- You are expected to train your model using mini-batches of the training set and may freely decide on the value for a mini-batch size. Discuss it in your report.
- Explain your choice of a loss function and its parameters, if any.

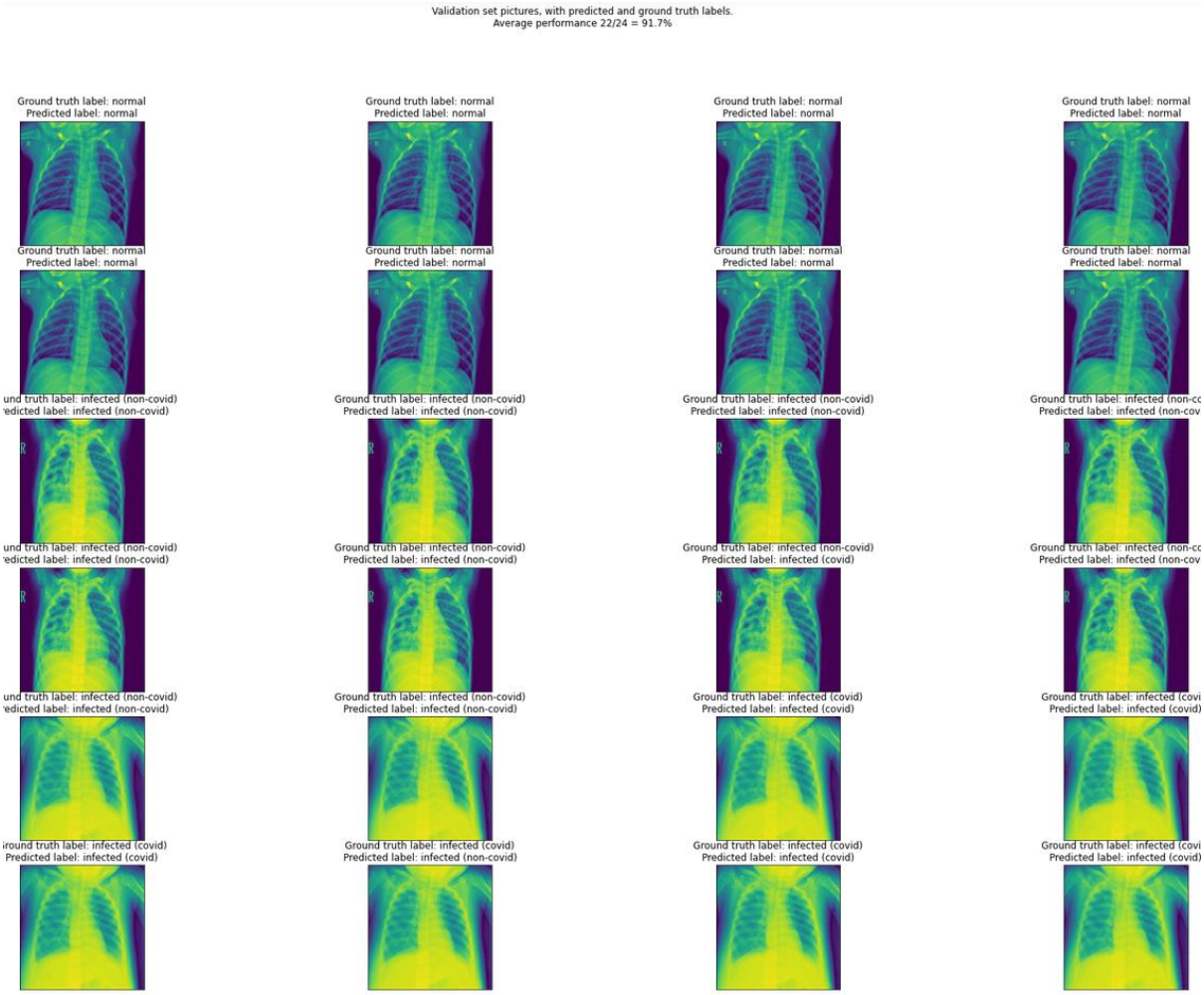
- Explain your choice of an optimizer and its parameters, if any (e.g. learning rate, momentum, etc.).
- Explain your choice of initialization for your model parameters.
- Your trainer function should display progress over time and show how the training progresses on periodic values of epochs. It is considered good practice for your trainer progress display to show the time at which the epoch ended, the loss values on your train sets, but also on your test set. It is also strongly advised to think about other relevant performance metrics, such as accuracy, recall, etc. An example of a (very minimal) display is given in the screenshot below. You may use the datetime and/or tqdm libraries if you see fit.

```
--- Epoch: 1/10 @ 2021-02-18 20:04:04.270534
Train Loss: 0.7115151882171631
```

- Your training function should also save your model regularly during the training phase. This is considered good practice: imagine training a model for days and its training gets interrupted by Windows Update restarting the computer. If you have saved your model, you may restart from where the training got interrupted, which is better than restarting from scratch.
- You also need to write a loader function, which will recreate the trained model from file, without the client (us) having to re-run the training entirely. This again, is considered good practice: your client may not have the time to re-train the whole model from scratch. More importantly, you want the client to be able to recreate the curves and performance results you showed in your report. Otherwise, there is no way to prove your model indeed achieved the performance you present in your report. Reproducibility is essential, especially when publishing research papers in the field, and not being able for us, professors, to reproduce your presented results will lead to penalties.
- You are expected to provide learning curves showing the evolution of your loss function and your other performance metrics over your successive training epochs for both your train and test sets. An example of such a graph is given in the screenshot below.



- After a complete training, you are expected to display the performance of your trained model on the 24 images in the validation set. We expect to see a subplot that looks like this and contains relevant information quantifying the performance of your model. Typically, ground truth and predicted labels for all 24 images, and possible overall metrics (Accuracy? Other metrics?).



- You might find it more difficult to differentiate between non-covid and covid x-rays, rather than between normal x-rays and infected (both covid and non-covid) people x-rays. Was that something to be expected? Discuss.
- Final question: would it be better to have a model with high overall accuracy or low true negatives/false positives rates on certain classes? Discuss.

C. Bonuses

Below is a list of non-mandatory questions, which might yield bonus points if implemented and correctly discussed.

- Implementing and discussing a learning rate scheduler and discuss its appropriate choice of parameters and benefits for your model.
- Implementing a regularization on your loss function and discuss its appropriate choice of parameters and benefits for your model.
- Briefly look up online how doctors diagnose infections based on x-rays. Does our AI seem to be able to reproduce this behavior correctly? Show typical samples of the dataset on which your AI failed and discuss what might have been the reasons.
- Feature maps visuals could also be interesting.

5. Project Delivery

A. Delivery details

Groups of 2-3 people (FYI, large project will most likely be 3-5). Mixing people from both classes is allowed.

Submission deadline: 21st March 2021, 11.59pm.

B. Recapitulative report

Your recapitulative report shall be submitted in a PDF format, along with your code.

You code may consists of .py files, ipynb jupyter notebooks, or Google Colab notebooks. Your PDF report should explicitly mention what needs to be done to run your code.

Properly documenting your notebooks/code files would be much appreciated, and you should practice it anyway (they are good practice!). Personally (Matt), I believe the most presentable format is a Jupyter Notebook, combining MarkDown cells and code cells, along with .py files containing the largest parts of your code: you just have to import them later in your Jupyter Notebook, to minimize the amount of code in your Notebook!

C. Project delivery

We strongly advise to upload your submission (code/notebooks + PDFs, but no dataset due to space restrictions) on a Github repository. You can then submit the link to your PUBLIC Github repository, during your submission on edimension.

Your Github repository for this project should contain your PDF report, your DOCUMENTED code/notebook files. It should also contain directions showing the required libraries and steps needed to re-train the model from scratch. And more importantly, it should also contain clear directions on how to recreate the exact trained model and its performance results you are presenting in the PDF, by loading some save PyTorch weights for your model. This is essential, for reproducibility reasons and something that we, researchers, must do constantly for our papers.

Below is a simple example of one of my projects on Github, which I shared with other professors at SUTD, showing how to randomize, password protect and send exam papers. Feel free to use it for inspiration!

https://github.com/matthieudemari/SUTD_quiz_randomizer

Important note: you are nearing the end of your curriculum at SUTD, gathering your projects and uploading them on your personal Github is much appreciated by recruiters as it allows them to immediately identify what your coding capabilities are. If you have not done so yet, please consider starting your project portfolio on Github!