Final Report

David Bücher, Timo Bürk, Félicien Hêche, Aleksandar Lazic, Zakhar Tymchenko

24.05.2020

General Organization

In our team, we organized our work as follows.
When we got the first instructions for the assignment, we waited a week before we made the first Discord meeting. It was really helpful to really understand what we should do and to start thinking about the implementation. It was also a way to catch up with the theory if something was not clear. Then, we discussed our different point of view until we were in agreement on the implementation. Finally, we tried to divide the work as equally as possible between the different team members.
We decided not to use a project management tool to organize our work, but we could have used something like Trello to clearly separate each member's tasks with the current status. Such tools are really useful in big projects as they give a better overview of the whole project (tasks to do, tasks done, ...)
Then, each member worked alone on the assigned part. We also used a WhatsApp group for small communications, to keep in touch during the week. It was helpful to get a quick answer from the team about some details (what should be the output of a certain function, what do you think about the README, ...)
To be more precise, we could say that for the task 2a to 2d, a single person was assigned to the task. Indeed, since it was not too much work to do, we thought it could be more complicated if many people worked on it. But, the important points like the architecture or the general implementations were discussed on the Discord meeting. You'll find a more detailed description in the next section for each specific task.

Tasks

2a SVM

In the first task we needed to use a support vector machine (SVM) to solve the MNIST dataset. Since we could choose any library we wanted, we used the sklearn library. Note, that we submitted two implementations of this task (version v1 and v2 on github). We had some misunderstandings about who should do the task and so, two people implemented it. But since, it seems to be a good work in both cases, we decided to put the two versions on github. Note that with this approach we were able to get an accuracy of 97,24% on the validation set.

2b MLP

Then, we implemented a Multiple Layer Perceptron (MLP) to solve the MNIST dataset. To build our MLP, we used the MLPClassifier class provided by sklearn, and for the grid search, we used the function GridSearchCV also provided by sklearn. With these different approach, we were able to get an accuracy of about 97.5% on the validation set.

2c CNN

In the task 2c, we needed to make a Convolutional Neural Network (CNN) to solve the MNIST dataset. To do it, we use the PyTorch library. The model we used was composed of three convolutional layers (always followed by the LeakyReLu activation function). And for the classification head, we used a simple linear layer. With this model, we get a 98,5% accuracy on the validation set.

2d MLP and CNN on permutated MNIST

After that, in the task 2d, we had to train and test our MLP and CNN model on a permutated MNIST dataset. We expected that the MLP will have about the same accuracy that the model train the real MNIST dataset. And for our CNN, we thought it will have a lower accuracy that on MNIST.
We observe an accuracy of 94,6% for the MLP (vs 97,5% on the normal MNIST) and for the CNN, we get an accuracy of 93% (vs 98.5%). So, as expected the MLP is less sensitive than the CNN to the original positions, but there is less differences than we thought.

3 Keyword Spotting with Dynamic Time Wrapping

Since this project was bigger than the previous tasks, we split it in the following part.
- Features Extraction Implementation.
- Dynamic Time Wrapping Implementation.
- Evaluation.

Then, each task were done by a different person with a defined deadline, to be sure that we would be able to submit the project in time.

5 Molecules

For the last task, since we have already implemented a Dynamic Time Wrapping in the previous task, we choose to work on the Molecules task. We split it in two different task. One person did the parsing and a person implemented the Graph Edit Distance. With our solution we get pretty good results despite using only a partial graph edit distance as we do not try to match the edges. Indeed we just compute a basic statistic on them (difference in number).

General Thoughts

In general, we think that our team worked quite well. We did not have any big issues. Of course, all was not perfect and we could had improved some things. For example, at the beginning, we had some communications problems. It was not really clear what each member should do. For this reason, we have two versions of the task 2b. But after this small misunderstanding, all worked fine and we are quite satisfied about the submitted project. Concerning the project itself, it was really nice that we were free to use either existing libraries or to implement the different algorithms ourselves. Although it is often unnecessary to re-implement existing libraries, it helps understanding the algorithm more in depth.