THOMAS MORE | UNIVERSITY OF APPLIED SCIENCES

INTERNSHIP UNIVERSITÉ DE BORDEAUX

Evidence of realization

Iris Loret

2023 - 2024

Content

1 Introduction

In this document, I will walk you through my internship at LaBri (Laboratoire Bordelais de Recherche en Informatique) for the University of Bordeaux. I will explain the steps I took to achieve the final result of the project. LaBri is a research lab, and my internship is focused on visualizing the layers of a deep learning model.

The goal is to make AI models more understandable. Currently, the processes within these layers are often referred to as a black box because we don’t fully understand what happens inside them. By making these processes visually easier to interpret, AI models could be used in more domains. For example, in the legal domain. Better visualization could lead to increased acceptance and trust from users, as it will be clear how an AI model arrives at its predictions.

In the following pages, I will describe in detail what I did during my internship, and include some images to illustrate my work.

2 What I did

2.1 Get and save the activations and clusters

The first thing I did was to start with making a draft of the code in a python notebook so it would be easier later to make the final code. The very first step in the draft was to make a model and train it. The model is structured by following the Lenet architecture and trained on the MNIST dataset. Which consists out of handwritten digits and is a well-known dataset used for image classification.

With that model we can go to the next step, retrieving the activations of every layer of the model and storing it in an HDF5 file. The activations are the outputs of a layer in a model. HDF stands for Hierarchical Data Format and is designed to store and organize large amounts of data. After the activations are saved I need to apply clustering on them. Clustering data means that the data will be grouped together. I chose to do this with Optics (Ordering Points To Identify the Clustering Structure) and I also saved those cluster labels in the same HDF5 file.

The next step entailed making dummy data that is used for making a dummy graph. So, I made three small HDF5 files that I could use to make a graph. I did this so it was easier to work with and more clear. I added nodes and edges to a matplotlib graph and then I got to this result. The nodes are the blue boxes which represent the clusters. The edges are the arrows which are the links between different nodes. So, it's a connection in the network.

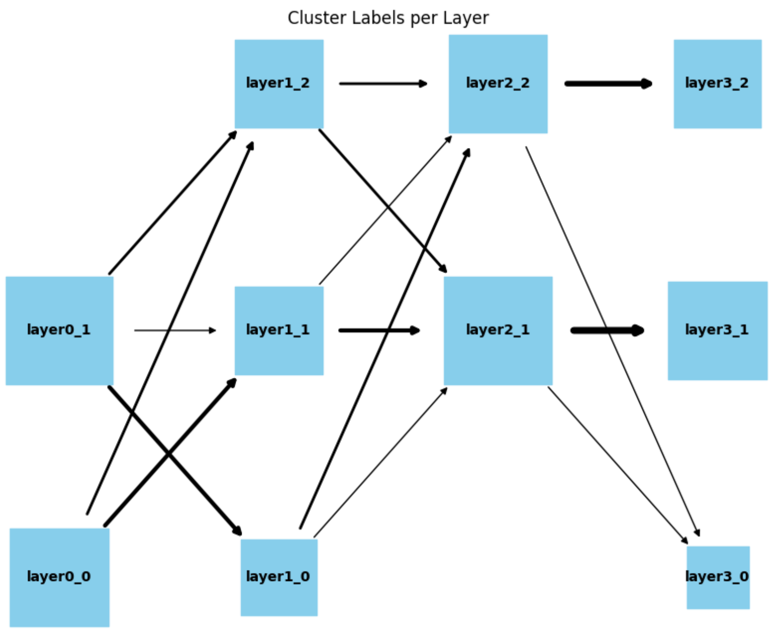

Figure 1

We can clearly see all the layers of the model and the clusters on Figure 1. The size of the square is determined by the number of elements in the cluster and the width of the arrow is determined by the weight which is the length of the indices of that group. A group are the samples where they leave from, and go to the same cluster, so they have the same source and destination cluster label.

2.2 Order the clusters

The next step is to arrange the clusters to minimize the number of edge crossings. There will always be some crossings but we want to limit the number. We can accomplish this by using the Sugiyama algorithm. This is also done in the form of a draft.

To make it more understandable of what needs to happen I have these two drawings. Figure 2 shows the original order, where you can see several edges crossing each other. The goal is to achieve a graph where the edge crossings are minimized such as in Figure 3. This way, it will be easier to understand and read the graph.

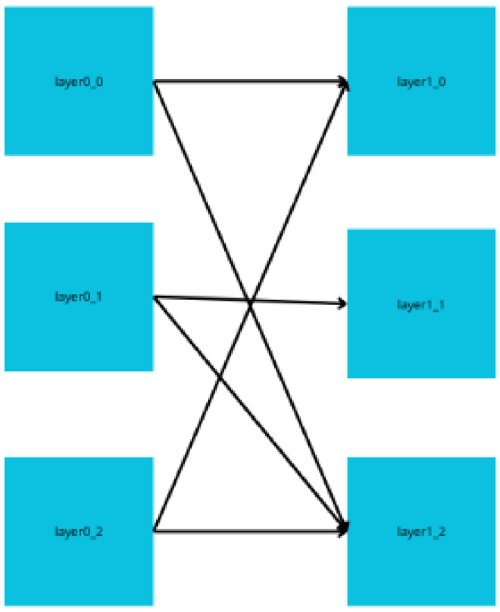

Figure 2

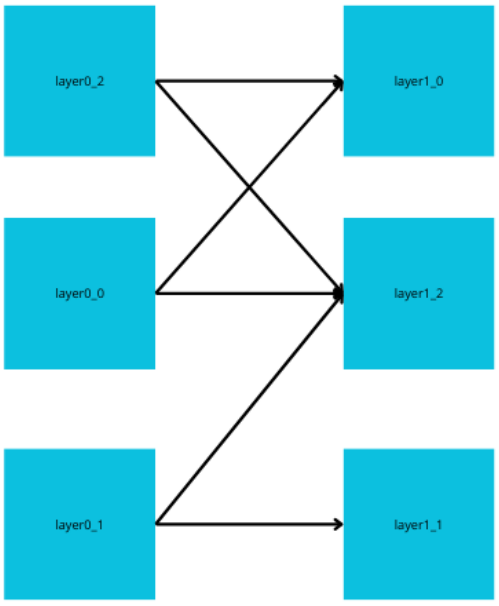

Figure 3

The Sugiyama algorithm will help us minimize the edge crossings. It uses the barycentre method to compute the new position while also considering the positions of neighbouring nodes. There can be a small issue sometimes with overlapping so that is also taken care of in the code. The new position will be computed by doing multiple passes over the graph, from left to right and from right to left.

After applying the algorithm to the graph, we need to compute the position by using the relative order because Sugiyama provides the correct order but not the exact position. So, I sorted the nodes and made sure there is some margin added. After that Figure 1 now looks like the following graph. We can clearly see that there are less edge crossings.

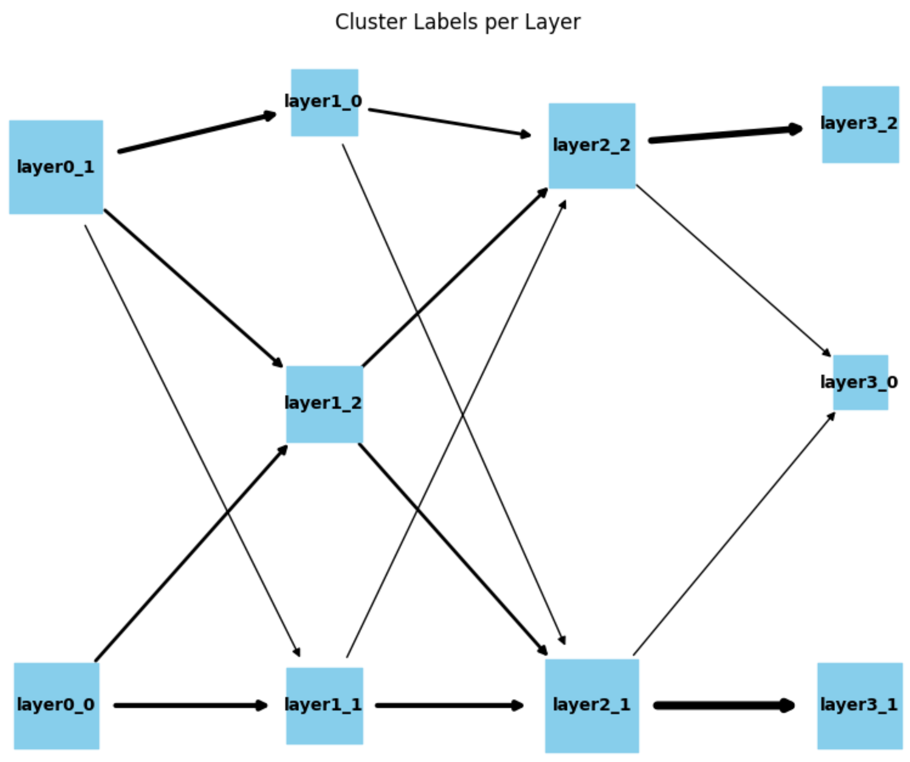

Figure 4

2.3 Order the samples

The same process used for ordering the clusters needs to be applied to the samples within the clusters. Also as a draft. To illustrate this, I created 2 drawings so it is easier to understand. In Figure 5 we see several edge crossings between different samples (black arrows). When we take a look at Figure 6, some of them are resolved. This is the result we want to achieve by applying Sugiyama on the samples inside the clusters, using the same method as we did for the clusters.

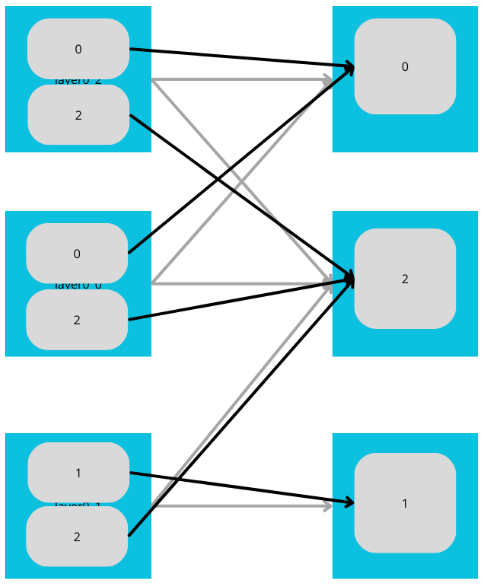

Figure 5

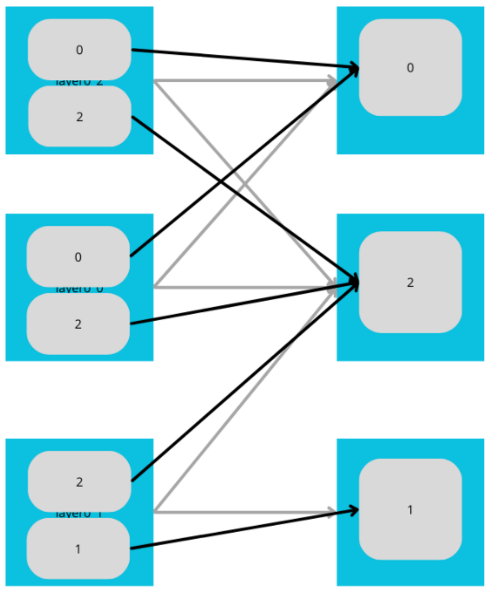

Figure 6

I started working on this but encountered some difficulties. Romain helped me with this part. But even after his help I was unable to get to a correct result. This will later be finished by the person who continues my project. Therefore, I cannot show the difference between the visualization before and after applying the algorithm.

2.4 Add local explanation

For the local explanation I again started with a draft in a python notebook. The goal of this part is to allow uploading a single image and visually seeing to which cluster it belongs and what path it takes in the graph.

I created a loop over all the layers in the model. Then retrieved the cluster numbers and activation values that are stored in the HDF5 file. The next step is to train a model by using these activation values and the cluster number, and then store that model. Thus, every layer has its own model. After that, I specify a sample and then loop over the layers where it gets the activation of the sample and predict the cluster by using the previous made model.

I was not able to add this part in the visualization but I do have the code in the draft and in the library (that is explained below) so it could definitely be used later on.

2.5 Make a library

After testing and finalizing all the notebooks (drafts), I moved on to the next step which is creating a library to streamline the use of all the code and set up the visualisation. I named my library euralFlowBuilder and added several files with functions. By using this library you can choose different architectures and datasets to build models, get activations, save activations, save cluster labels and execute the local explanation.

Another functionality that will be added later, when someone finished the project, is the ability to use the library for ordering the clusters and samples in the graphs. This feature will then also be used in the visualization.

I also didn’t have enough time to implement a pipeline that makes use of this library to setup all the parts including a web server.

2.6 Make the visualization

For the visualization I started by hardcoding everything. This means that I wasn't using the real data but rather data that I manually input. I made the graph look like the graph from the previous project that was done and achieved the following result.

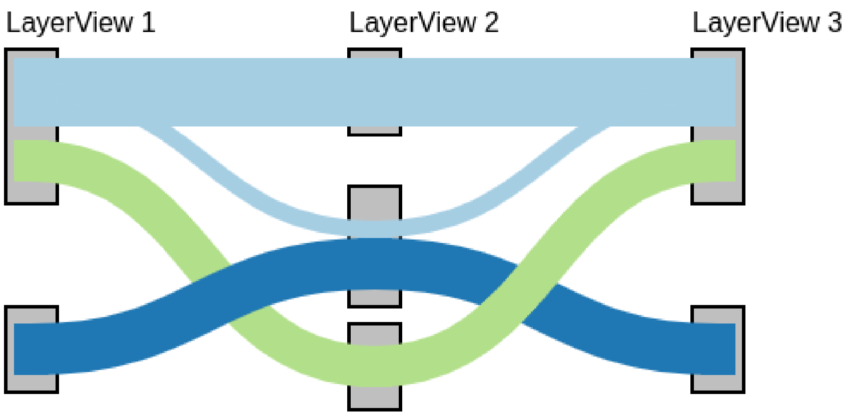

Figure 7

I encountered a few errors that needed correction, such as adjusting the position of the clusters, nodes and edges, centring the layer name and styling issues. After fixing all of these bugs, the next graph looked like this.

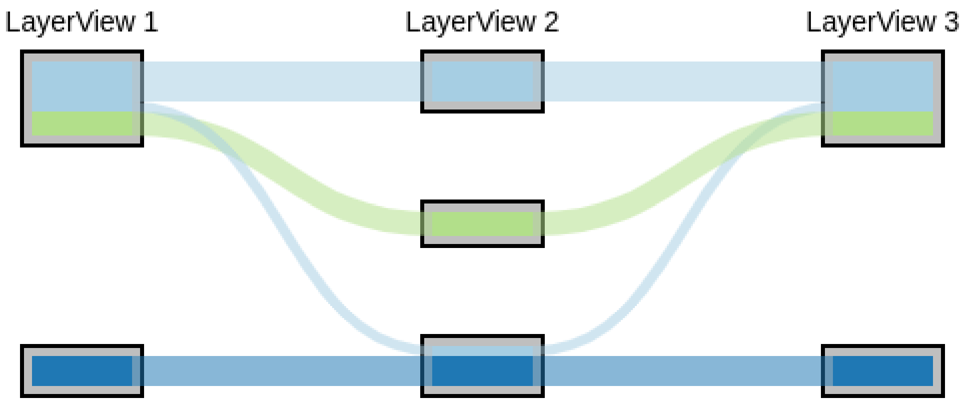

Figure 8

This was also the final design of the graph. The next step was to convert the hardcoded data into JSON structure. This is a standard text-based format for representing structured data. The graph looked exactly the same as Figure 8 since the only thing that changed is how the data is used.

This was all I could finish but the next step would have been to use the HDF5 files and transform them in JSON format so the graph could be made in the exact same way.

3 Conclusion

In conclusion, I can say that I am proud of everything I achieved with this project, even though not everything is completely finished I am still glad with the result. I've made it easy for someone else to continue working further on this project by storing everything on GitHub and providing a README file explaining the purpose of the files. Additionally, I created issues on GitHub outlining the remaining tasks. So, the next person can easily find what is left to do and in which file they need to work.

I learned a lot through this project and it helped me develop my technical skills further. Working on this project also thought me the importance of good documentation of the code so it's easier to understand.

I would like to thank Romain Giot for his help when I needed it. And also to Romain Bourqui, since they provided me this project. Also thanks to Luc-Étienne Pomme-Cassierou and Frédéric Lalanne for helping me when I encountered difficulties.