RÉPUBLIQUE FRANÇAISE
Liberté
Égalité
Fraternité

utbm
université de technologie Belfort-Montbéliard

University of Technology of BELFORT-MONTBELIARD

Development of a Large Language Model for the pharmaceutical industry

ST40 Internship Report – P2024

MORALES Léon
Engineer
FISE-INFO

KSILINK
16 rue d'Ankara
67 000 Strasbourg
www.ksilink.com

Company Supervisor
OGIER Arnaud

UTBM **Supervisor**
LOMBARD Alexandre

Acknowledgements

I would first like to express my deep gratitude to Arnaud Ogier, my internship supervisor at Ksilink. From my spontaneous application via LinkedIn, he reacted with impressive speed and availability, responding to me and organizing an interview within the same day. Throughout my internship, his wise advice, patience, and teaching methods not only allowed me to progress but also to feel comfortable within the team quickly.

I also thank Keyhan and Zahra, who always welcomed my countless questions with kindness and enthusiasm. Their ability to answer them clearly, or to guide me toward suitable solutions, was invaluable support in the advancement of my projects.

My thanks also go to all the Ksilink teams, the "Ksilinkers," who warmly welcomed me from the very first day. Their friendliness and team spirit made this experience a memorable time, whether during coffee breaks, lunches, or afterwork gatherings.

Finally, I want to thank Alexandre Lombard, my advisor at the University of Technology of Belfort-Montbéliard. His attentive follow-up of my internship and his valuable advice for writing this report were a great help to me throughout this adventure.

Table of Contents

Table of Abbreviations:

- ◊ **CNN : Convolution Neural Network,**
  Used mainly for image processing and analysis.
- ◊ **KG : Knowledge Graph**
  A structure that links entities (objects, concepts) by relations, used to represent and query complex data.
- ◊ **LLM : Large Language Model**
  Category of models trained using immense amounts of data to understand and generate text in natural language. The best-known examples are OpenAI's ChatGPT or Google's Gemini.
- ◊ **SMILES : Simplified Molecular Input Line Entry System**
  Textual format for representing the chemical structure of molecules as character strings.
- ◊ **SQL : Structured Query Language**
  Language used to interact with relational databases.
- ◊ **CRISPR : Clustered Regularly Interspaced Short Palindromic Repeats**
  Gene editing technology that allows for precise DNA modification.
- ◊ **UMAP : Uniform Manifold Approximation and Projection**
  Dimensionality reduction algorithm used to visualize complex data while preserving their global structures.
- ◊ **Graph RAG : Retrieval-Augmented Generation Graph**
  A technique combining data graphs with generative models, used to improve the search and response capabilities of models.
- ◊ **IA : Intelligence Artificielle**
  A computer science discipline aiming to design systems capable of performing tasks that usually require human intelligence, such as visual recognition, language processing, or decision-making.
- ◊ **MNIST : Modified National Institute of Standards and Technology**
  Database used for training and validating classification algorithms, composed of images of handwritten digits.

I. Introduction

A. Ksilink: A revolution in drug discovery

Ksilink[1], founded in 2014 in Strasbourg, is an innovative Franco-German association that is revolutionizing drug discovery through approaches based on patient-derived cellular models. With the status of an SME, this unique biotech combines exceptional skills in high-throughput phenotypic screening (HTS-HCS) and artificial intelligence-assisted image analysis to accelerate the development of next-generation therapies.

Ksilink's approach is based on the use of human cellular models that faithfully reflect diseases, thereby reducing the risk of failure in the clinical phase. By collaborating with academic, industrial, and clinical partners, the organization combines expertise and resources to meet unmet medical needs in areas such as neurological, muscular, cardiac[2] and oncological diseases.

Thanks to its network of excellence, Ksilink integrates unique capabilities in disease modeling, phenotypic screening, and preclinical development. This strategy, which combines technological innovation and interdisciplinary collaboration, helps to speed up the time-to-market for new therapies while minimizing risks.

Ksilink conducts drug discovery programs based on human cellular models, primarily targeting neurological and muscular diseases. These collaborative programs are developed until innovative therapeutic candidates are obtained, which are then transferred to the pharmaceutical industry for their final development.

The launch of Ksilink was supported by an initial financing of 16 million euros from the "Programme d'Investissement d'Avenir" (PIA), with support from Inserm and Sanofi. After a structuring and acceleration phase, Ksilink strengthened its independence by recruiting its own staff and investing in an automated screening platform. In 2022, its technological activities were transferred to a subsidiary in the form of a simplified joint-stock company (SAS) in order to valorize the results and develop concrete solutions for the industry.

Ksilink currently has 24 members spread across several departments; here is an overview in Figure 1:

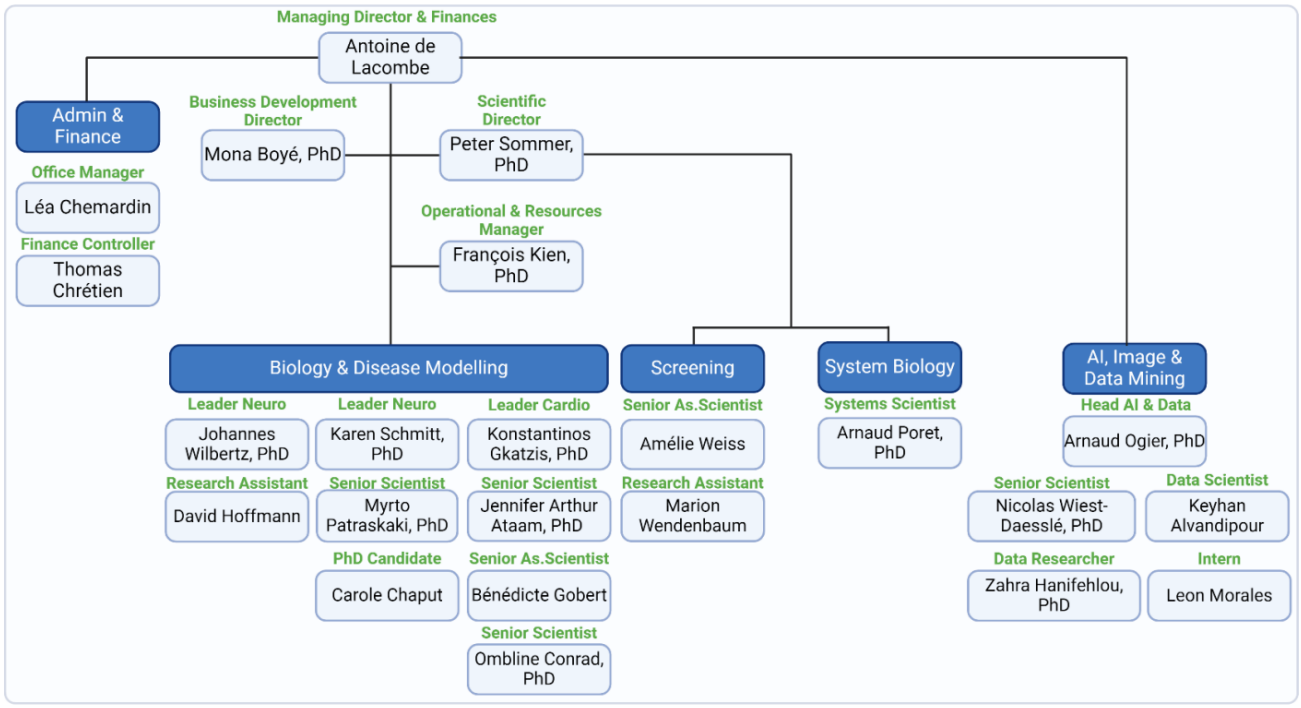


Figure 1: Ksilink organizational chart

The main departments at Ksilink are:

- ◊ **Biology & Disease Modelling**
  This team focuses on modeling human diseases using patient-derived cellular models. It works on specific areas like neurological and muscular diseases to develop solid foundations for drug discovery.
- ◊ **Screening**
  This department performs high-throughput phenotypic screening (HTS-HCS) to identify promising compounds. The team collaborates closely with other departments to ensure the relevance of the analyses.
- ◊ **System Biology**
  This department integrates biological data to understand the complex interactions between biological systems. It supports projects by providing an integrated vision and identifying potential therapeutic targets.
- ◊ **AI, Image & Data Mining (My Department)**
  This department leverages artificial intelligence and advanced image analysis to extract valuable information from cellular models.

B. The AI, Image & Data Mining department

The AI, Image & Data Mining department is a central pillar of innovation at Ksilink. It leverages cutting-edge technologies in artificial intelligence to optimize high-throughput phenotypic screening and thus accelerate the development of new therapies.

By analyzing massive data from complex cellular images, this department identifies promising molecules capable of correcting anomalies associated with serious diseases. Thanks to advanced algorithms, AI allows for a fine analysis of thousands of cellular parameters, accurately differentiating between healthy and diseased phenotypes.

The department plays a key role in processing the images generated by automated screening, using deep learning techniques to detect subtle changes in cells exposed to active substances. This approach de-risks research by identifying the most promising drug candidates early on, while reducing the costs and timelines of preclinical development.

By combining technological innovation and scientific expertise, the AI, Image & Data Mining department participates in Ksilink's mission: to transform translational research into tangible therapeutic solutions

C. My role in the department

During my internship in the AI, Image & Data Mining department, I had the opportunity to contribute to two projects closely related to artificial intelligence:

A novel project based on recent research
The first project, entirely new for Ksilink, was based on a scientific article [3] publié en août 2024. En tant que principal contributeur, j'ai pu développer ce projet en autonomie, tout en bénéficiant du soutien précieux de mon tuteur de stage, Arnaud OGIER, et de Keyhan ALVANDIPOUR, qui supervisait l'avancement. [3] published in August 2024. As the main contributor, I was able to develop this project independently, while benefiting from the valuable support of my internship supervisor, Arnaud OGIER, and Keyhan ALVANDIPOUR, who supervised the progress.

Completion of a knowledge graph and improvement of an existing chatbot
The second project focused mainly on adding information to the knowledge graph and improving a chatbot, previously implemented by Zahra and Arnaud. My role consisted of optimizing the features and performance of this tool, drawing on the expertise of my colleagues to integrate new solutions adapted to the department's needs and to make its use accessible and useful to scientists in other departments.

These two projects offered me a complete immersion in the Ksilink ecosystem, allowing me to apply my knowledge to real-world problems while contributing to concrete advances in medical research.

II. Work Performed

A. Defined Subject

The subject of my internship focused on the development of a **Large Language Model** (LLM) applied to the pharmaceutical industry, by leveraging Ksilink's internal data. The main objective was to create an interface combining a **Knowledge Graph** (KG) and an **LLM**, in order to better exploit the relationships between different concepts extracted from various data while keeping the data in-house.

To achieve this objective, the project was based on the use of open-source frameworks such as **OLLAMA** or similar alternatives, integrated with tools like **PyTorch** for designing and training the models. This approach aimed to structure the complex data from Ksilink's databases while making them more accessible. Thanks to this solution, biologists could explore this information intuitively via a natural language interface.

B. Actual Subject

During the internship, the project evolved to meet specific technical challenges, particularly related to the integration of heterogeneous data. The subject was divided into two distinct parts:

1. **Construction of a common latent space:** This step aimed to integrate several types of data, including molecular descriptions (in SMILES format) and information extracted from cell images. The objective was to represent this data in a shared vector space, thus facilitating their joint analysis and exploitation, with the goal of creating a generator for new molecules.
2. **Knowledge graph and link with the LLM:** This part aimed to enrich the knowledge graph by adding new relations and information. This also included improving the interface connecting the knowledge graph to an LLM, to allow natural language interactions and the addition of features making the tool more instinctive and understandable.

C. Background

To prepare for this internship and acquire a foundational knowledge of artificial intelligence (AI), I undertook several personal steps before my arrival at Ksilink.

First, I studied the history and evolutions of AI through specialized books [4,5]. This allowed me to understand the theoretical foundations and the major advances that have marked this constantly evolving field. In addition, this exploration gave me a global vision of the fields of application and the challenges related to artificial intelligence.

Next, I developed several personal projects aimed at putting basic programming and AI concepts into practice. Among these projects, I created:

- ◊ An **automatic Snake game**, where I implemented an algorithm for the snake to automatically find its way to the food.
- ◊ An advanced version of the **Tic-Tac-Toe** game, including a variant called **Ultimate Tic-Tac-Toe**, composed of 9 connected grids where the goal is to win three aligned grids.

These experiences allowed me to consolidate my algorithmic skills.

However, despite these efforts, I quickly realized that these projects were not sufficient to meet the level of requirement for my internship. The problems encountered at Ksilink, particularly in image recognition and deep learning, required much more specific skills. To bridge this gap, the first weeks of my internship were dedicated to intensive skill-building. This included training on advanced concepts such as:

- ◊ Convolutional Neural Networks (CNNs) for image processing.[6,7]
- ◊ Classification [8] and clustering techniques to organize and analyze data.
- ◊ The use of tools for visualizing and evaluating models. [9]

These learning efforts allowed me to acquire the necessary foundations to meet the project's challenges, while also allowing me to effectively integrate the methodologies used at Ksilink.

D. Targeted Objectives

My internship objectives were organized into several stages, with a balance between acquiring technical skills and their practical application within Ksilink's projects.

1. **Acquisition of skills in artificial intelligence:**
    - ◊ Develop a deep understanding of the theoretical bases of artificial intelligence, particularly machine learning algorithms and neural networks.
    - ◊ Perform calculations by hand for the main steps of supervised learning models, such as backpropagation, to master the underlying concepts.
2. **First practical application with a view to future autonomy:**
    - ◊ Design a convolutional neural network (CNN) capable of classifying images taken with microscopes into four categories.
    - ◊ This project aimed to apply my theoretical knowledge to a concrete case while using analysis tools to visualize the results.
    - ◊ Explore advanced model interpretation techniques, such as gradient integration and visualization of activation layers, to better understand the model's errors and limitations.
3. **Scientific study and reproduction of results:**
    - ◊ Study in detail the scientific article [3] at the basis of the main project, in order to understand the methodologies used.
    - ◊ Reproduce the results described in the article with Ksilink's internal data, including the creation of a common latent space to integrate heterogeneous data, and explore possible applications from this space.
4. **Enrichment of Ksilink's knowledge graph:**
    - ◊ Add relevant information and establish new relationships within the existing knowledge graph, which includes over 100,000 nodes and 2.5 million relations.
    - ◊ Test multiple approaches to exploit the KG:
        - ♦ Implement a chatbot capable of responding to natural language queries using Ksilink's knowledge graph.
        - ♦ Implement new functions for the chatbot (visualization of responses as a graph).
        - ♦ Train a model to generate new connections between nodes.

These objectives reflect a logical progression, moving from theoretical training to concrete projects, while integrating advanced methodologies and adapted tools. They guided my work throughout the internship, while allowing me to deepen my understanding of the technologies used at Ksilink.

E. Dated Work Schedule

a) Provisional Schedule

The initial schedule for my internship was organized into several main phases, each corresponding to a key step in achieving the set objectives:

1. **Step 1: 1 month:** Training and skill-building.
   - ◊ Understand the theoretical and practical bases of artificial intelligence, including image processing and deep learning methods like convolutional neural networks (CNNs).
   - ◊ Complete introductory projects to master the tools and concepts necessary for the main project.
2. **Step 2: 2 months:** Development of the first project.
   - ◊ Reproduce the results of a scientific article by building a common latent space to integrate heterogeneous data (molecular structures and imaging data).
   - ◊ Achieve a first concrete result before the end of this phase.
3. **Step 3: 1 month:** Deepening of the main project.
   - ◊ Generate data from the common latent space.
4. **Step 4: Until the end of the internship:** Integration with the knowledge graph.
   - ◊ Add new relations and information to the knowledge graph.
   - ◊ Implement a chatbot allowing natural language queries.
   - ◊ Train a model capable of predicting new relations between nodes, if possible.

b) Actual Schedule

Due to technical challenges encountered, particularly related to data quality and heterogeneity, the schedule was adjusted as follows:

1. **Step 1: Initial Training (Weeks 1 to 4)**
   - ◊ Creation of a simple convolutional neural network (CNN) to recognize handwritten digits (MNIST dataset).
   - ◊ Development of a more complex CNN to classify molecule images from Ksilink, with integration of visualization tools like TensorBoard.
   - ◊ These steps respected the initial schedule and allowed me to master the necessary foundations for what followed.
2. **Step 2: Reproduction of Scientific Results (Weeks 5 to 13)**
   - ◊ Attempt to reproduce the results described in the article with Ksilink's data.
   - ◊ Although the results were satisfactory with the article's data, it was impossible to reproduce them with the internal data due to significant differences in their structure and quality.
   - ◊ This difficulty led to this project being put on hold.

3. **Step 3: Abandonment of generation from the latent space**
    - ◊ Due to the limitations encountered in the previous step, this phase could not be carried out and was postponed pending more suitable data.
4. **Step 4: Enrichment of the Knowledge Graph (Weeks 14 to end of internship)**
    - ◊ Addition of new data to the knowledge graph to improve its richness and relevance.
    - ◊ Preparation of a training environment with a small experimental graph to avoid any negative impact on the main graph.
    - ◊ Integration of the chatbot linked to the knowledge graph.

Despite the necessary adjustments, the actual schedule allowed me to follow a logical progression and achieve concrete results in important parts of the project. The unforeseen issues related to the data were an opportunity to learn to adapt my approaches and to prioritize tasks according to the constraints encountered.

III. Work Progress

A. Upgrading Skills

The skill-upgrading phase took the form of two projects carried out largely autonomously, with occasional support from Keyhan ALVANDIPOUR and Arnaud OGIER to answer my questions and guide me. This step was essential for acquiring the skills and knowledge necessary for the rest of my internship. I also completed a quick exercise to familiarize myself with handling .tar files (Appendix 2 and 3)

Understanding the basics: perceptron and backpropagation
Before starting the projects, I took the time to familiarize myself with fundamental machine learning concepts. For example, I studied the functioning of a perceptron, a basic unit of neural networks, to better understand its logic. I also manually performed (Figure 2) the backpropagation calculations[10], a method for adjusting the weights of a neural network to minimize the error between its predictions and the expected values.

These calculations allowed me to concretely assimilate the mathematical principles used.

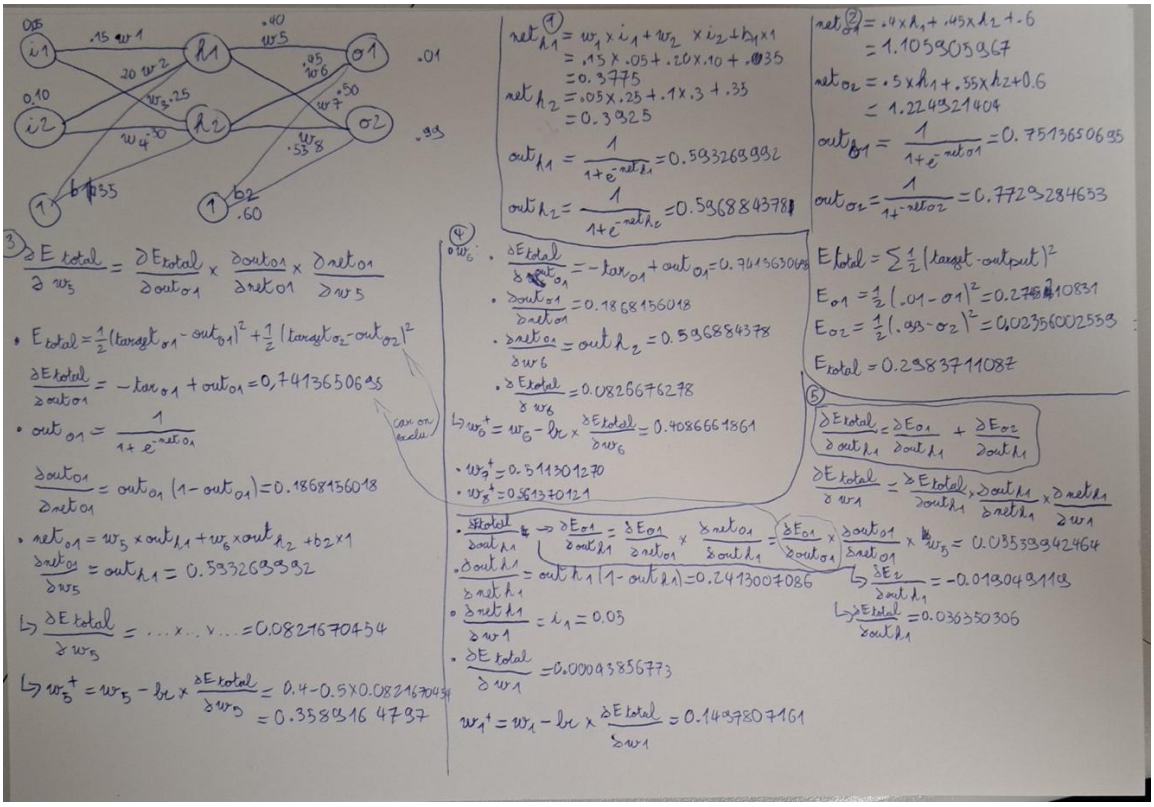


Figure 2: Front of my draft of backpropagation calculation

a) Handwritten digit recognition

For my first project, I developed, using Python and more specifically PyTorch, a simple neural network capable of recognizing handwritten digits from 0 to 9. This project, often considered an introductory exercise in machine learning, allowed me to explore key concepts:

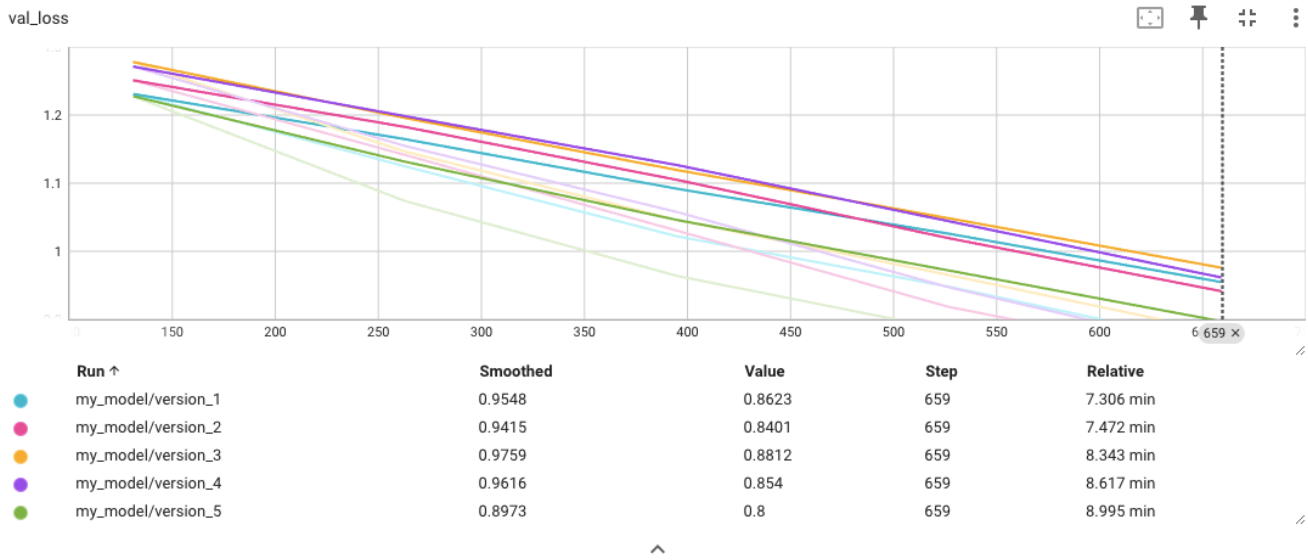
- ◊ **Loss and activation functions**: They influence how the network evaluates and adjusts its performance.
- ◊ **Learning rate** : The parameter controlling the model's learning speed.
- ◊ **Optimiseurs** : Algorithms that adjust the network's weights efficiently.

Thanks to the many tutorials available online [6,7], I was able to build an accurate initial model and gain a good understanding of these notions. This work laid the foundation for the next, more complex project.

b) Classification of molecule images

The second project aimed to classify images of cells generated at Ksilink into four distinct categories according to their activities and effects. In this section, these different groups will be named group 1, 2, 3, 4 so as not to disclose the different categories. This step was particularly enriching, as it allowed me to work on more advanced aspects:

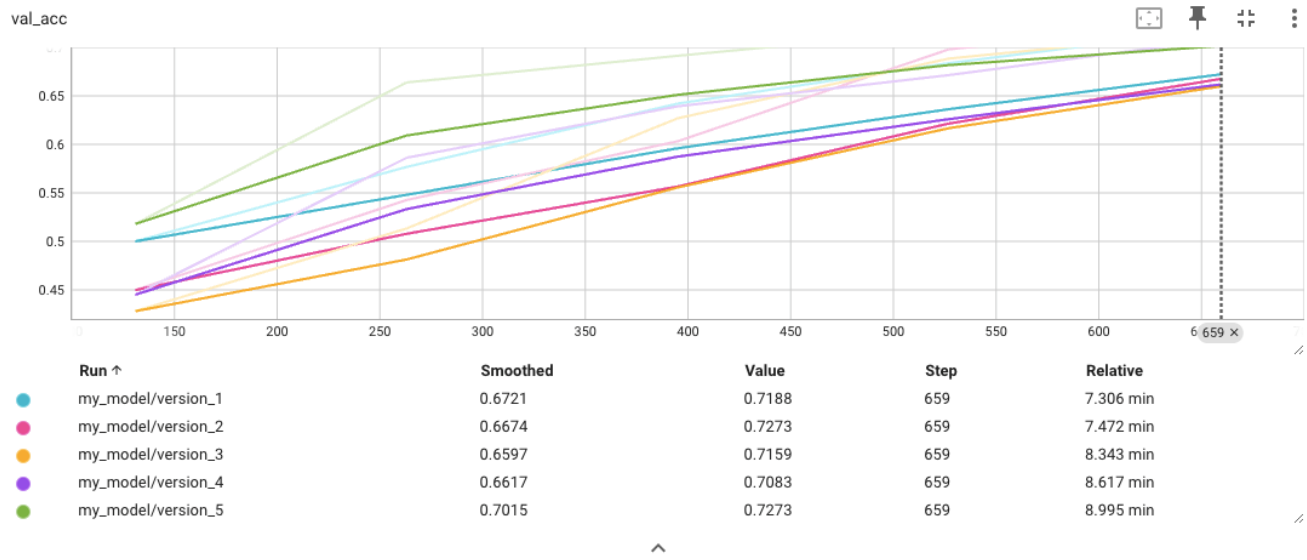
- ◊ **Dataset preparation**: Creating a dataset proved to be crucial for ensuring the reliability of the model's training. A well-structured database reduces biases and errors likely to skew the results.
- ◊ **Model optimization**: After designing an initial model, I integrated techniques like **dropout**, tests on different parameters, and analysis with TensorBoard [11] to identify and correct the observed problems.



Run	Smoothed	Value	Step	Relative
my_model/version_1	0.9548	0.8623	659	7.306 min
my_model/version_2	0.9415	0.8401	659	7.472 min
my_model/version_3	0.9759	0.8812	659	8.343 min
my_model/version_4	0.9616	0.854	659	8.617 min
my_model/version_5	0.8973	0.8	659	8.995 min

Figure 3: Validation Loss of the model in the first trials

The loss function represents the discrepancy between the model's predicted output and the actual target value. During training, the model attempts to minimize this loss. **The validation loss** measures the model's performance on the validation data, i.e., data not seen during training.

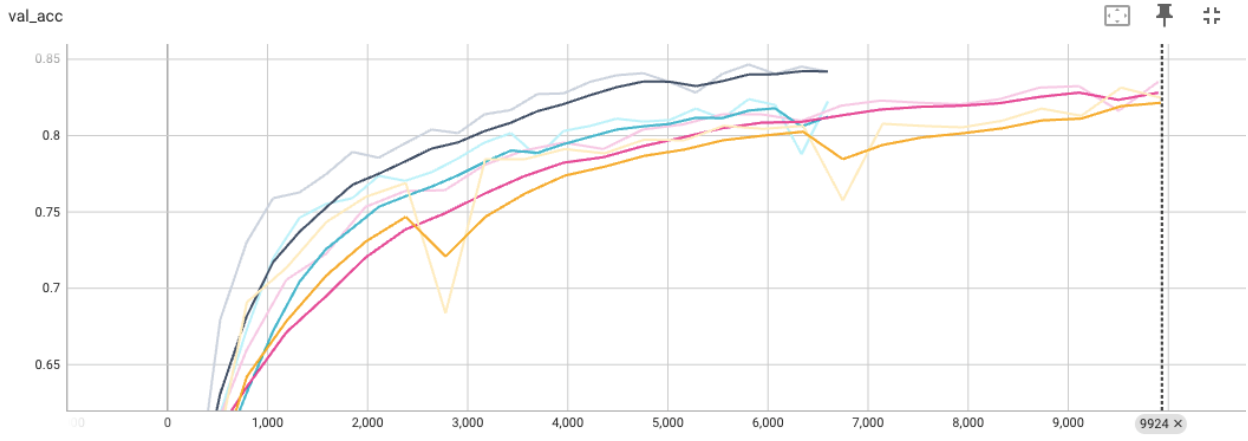


Run	Smoothed	Value	Step	Relative
my_model/version_1	0.6721	0.7188	659	7.306 min
my_model/version_2	0.6674	0.7273	659	7.472 min
my_model/version_3	0.6597	0.7159	659	8.343 min
my_model/version_4	0.6617	0.7083	659	8.617 min
my_model/version_5	0.7015	0.7273	659	8.995 min

Figure 4: Validation accuracy of the model in the first trials

The **validation accuracy** indicates the proportion of correct predictions the model makes on the validation data.

On these two graphs (figure 3 and 4), retrieved thanks to Tensorboard, we can see that the validation loss and validation accuracy curves continue to respectively rise and fall without stabilizing, so the model has not yet reached its convergence point. It is therefore necessary to increase the number of training repetitions to give the model time to learn.
After realizing this, here is the result:



Run	Smoothed	Value	Step	Relative
my_model/version_0	0.8421	0.8419	6,599	6.245 min
my_model/version_1	0.8125	0.8224	6,599	4.392 min
my_model/version_2	0.8281	0.8352	9,899	3.738 min
my_model/version_3	0.8215	0.8248	9,924	3.669 min

Figure 5: Validation accuracy of the model after changing the number of training repetitions

In Figure 5, we can see that the model's validation accuracy begins to stabilize around 0.84. By giving the model more time to train, we can clearly see that this allowed it to learn more and refine its predictions.

Subsequently, I switched from **pytorch** to **pytorch lightning** [12], which will be used in the following projects. In addition to preparing me for the rest of my internship, this allowed me to restructure my code and make it clearer at the same time.

Seeing that the model was no longer progressing, I looked for reasons and new methods to improve it.

Advanced optimization: Normalization and data augmentation

To improve the model's performance, I explored several advanced methods. Two approaches particularly caught my attention: **normalization** and **data augmentation**.

- **Data normalization**: This technique consists of reducing significant discrepancies between the values of the input data, to prevent certain values from dominating others in the learning process.
- **Data augmentation[13]** : The objective here is to make the model more robust and capable of generalizing to new data. To do this, I introduced modifications to the images in the dataset, such as:
  - Random 90° rotation.
  - Horizontal flip.
  - Adding noise.
  - Partial erasing of an image.

  These augmentations were applied to 50% of the training images, and each modified image could have one or more transformations. However, I noticed that some augmentations were not suitable, so I adjusted the parameters to keep only those that brought significant improvements and remove others, like color change, which was useless in my case.

Furthermore, to improve the reproducibility of the training and limit the effects of chance in the distribution of data (training, validation, test) and in the application of augmentations, I fixed a random seed. This allowed me to better analyze the impacts of the modifications on the model's performance.

Model improvements: Reduction and regularization

Faced with the limitations encountered by the model, I undertook several adjustments to improve its performance:

- Reducing the model size: A more compact model learns faster and is less prone to overfitting.
- Scheduler for the learning rate: This mechanism gradually reduces the learning speed over iterations, allowing the model to stabilize on an optimal solution.
- Weight decay: A regularization technique aimed at controlling the modification of weights to prevent the model from fitting too closely to the training data.

These adjustments made it possible to obtain a better-optimized model (Figure 6), whose performance was more consistent.

Here is the final model used (Figure 6):

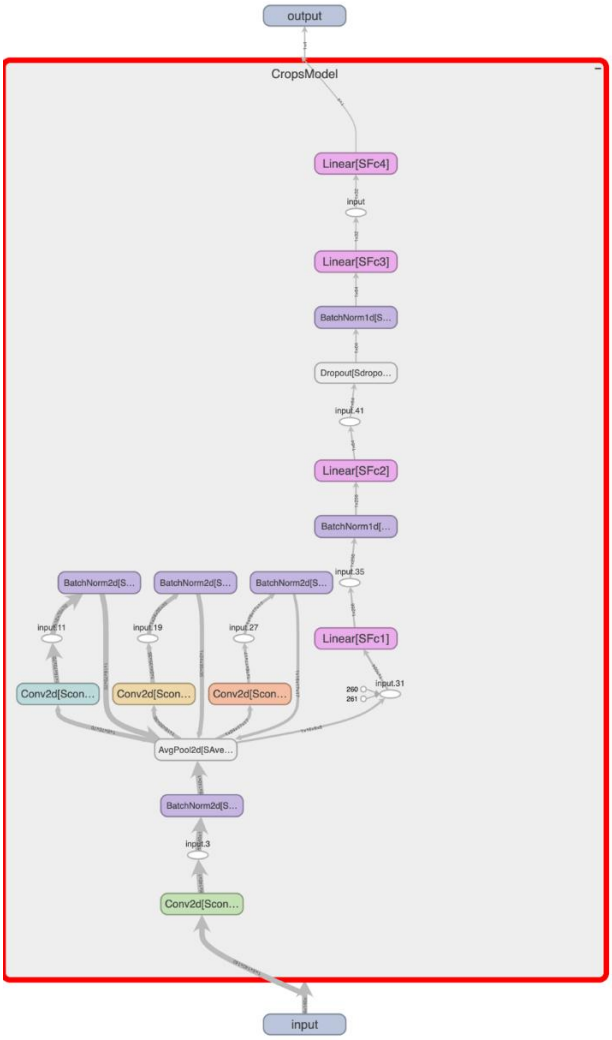


Figure 6: Graph of the model via tensorboard (Appendix 1 for a full-page version)

The final model had an optimized architecture, taking 140×140 images with 4 channels as input, and successively applying convolutional layers. Each layer reduced the image size while increasing the number of channels, resulting in an output of 17×17 pixels with 24 channels (see Appendix 4 and 5). This structure helped preserve the essential features of the images while eliminating superfluous details.

Error analysis and visualization

To better understand the model's limitations, I used various visualization tools:

1. **Confusion matrix**: This representation allowed me to identify the categories that the model frequently confused. For example, the first matrices showed difficulties in differentiating the first and second groups, as well as the third and

fourth groups (Figure 7). These observations were confirmed by Keyhan, who pointed out the visual similarity between these groups.

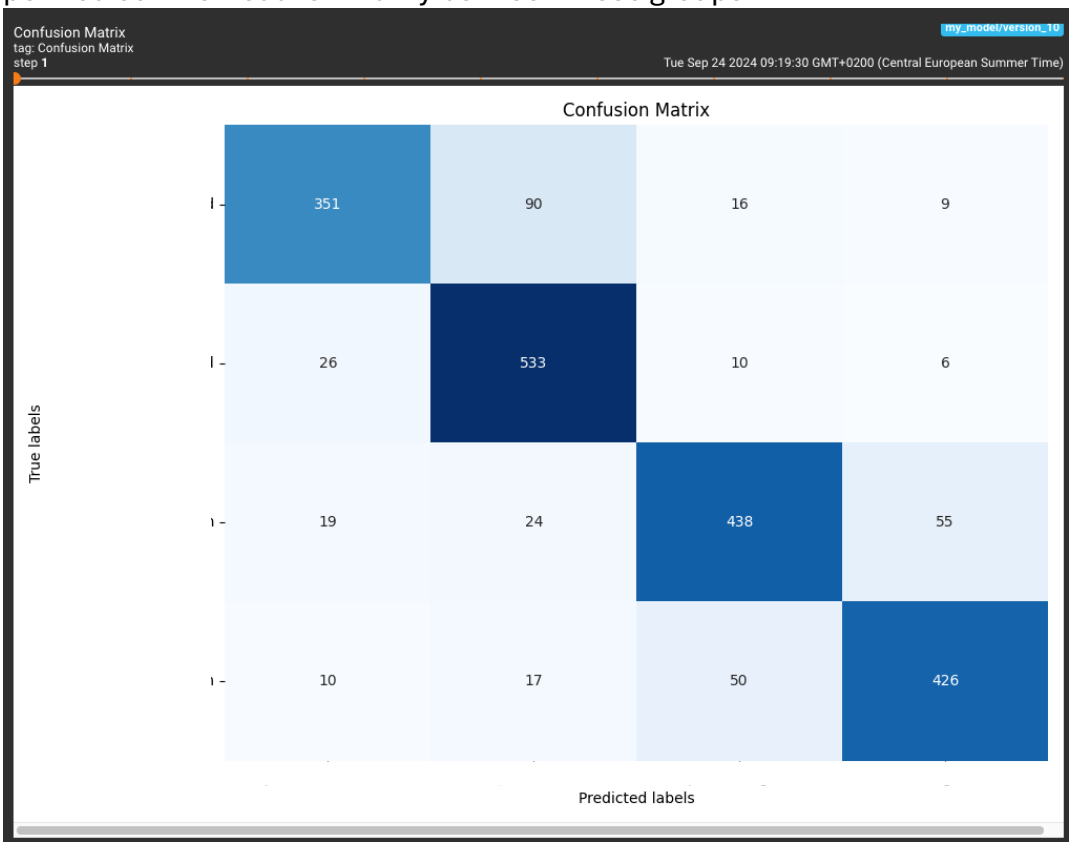


Figure 7: Confusion matrix at step 1 where each column of the table contains a class predicted by the algorithm and the rows the actual classes. The values on the diagonal represent correct predictions, while the values off the diagonal are those incorrectly predicted by the model.

2. **Misclassified images**: By displaying the incorrect examples, I was able to analyze the model's predictions and check the validity of the labels. This led me to conclude that some errors stemmed from poor data labeling.

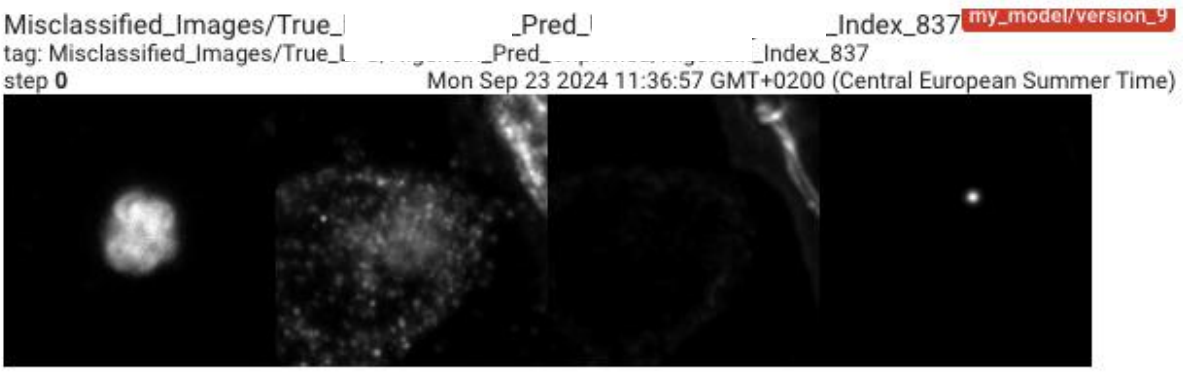


Figure 8: Display of the four channels of a misclassified image

Each image here represents the same compound in a different channel. This compound belongs to group one, whereas it is classified by the model as group two. These images allowed me to confirm what Keyhan had told me.

3. **Gradient Visualization[9]** : By using gradient-based methods, I observed the areas of the image that the model considered important. For example, the analysis of Figure 9 revealed that the model was relying primarily on a specific channel, which guided my efforts to balance the utilization of the input channels.

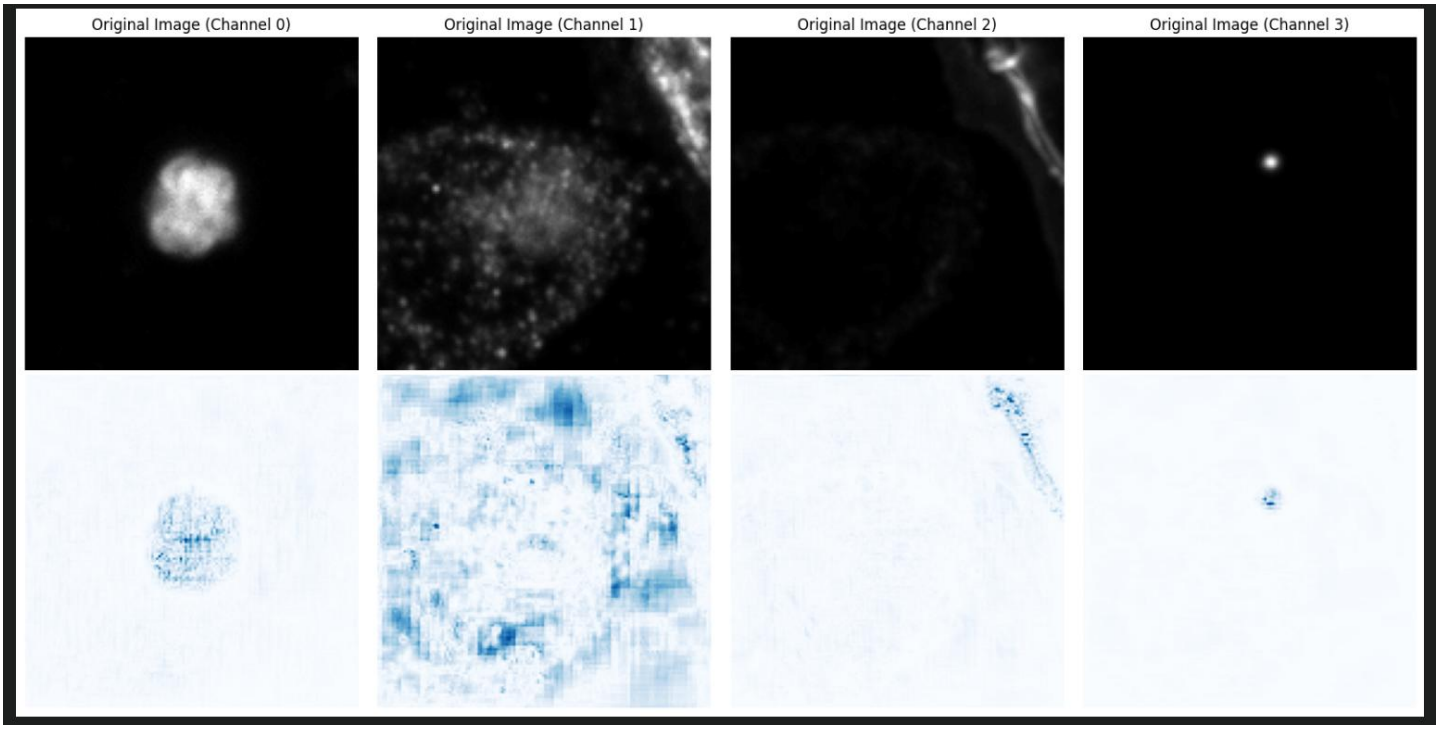


Figure 9: Gradient descent and display of model predictions

On these images, the more intense the blue, the more importance the model places on that part. This helps to understand what is important to the model.

For better visualization, I decided to display this according to the convolutional layer.

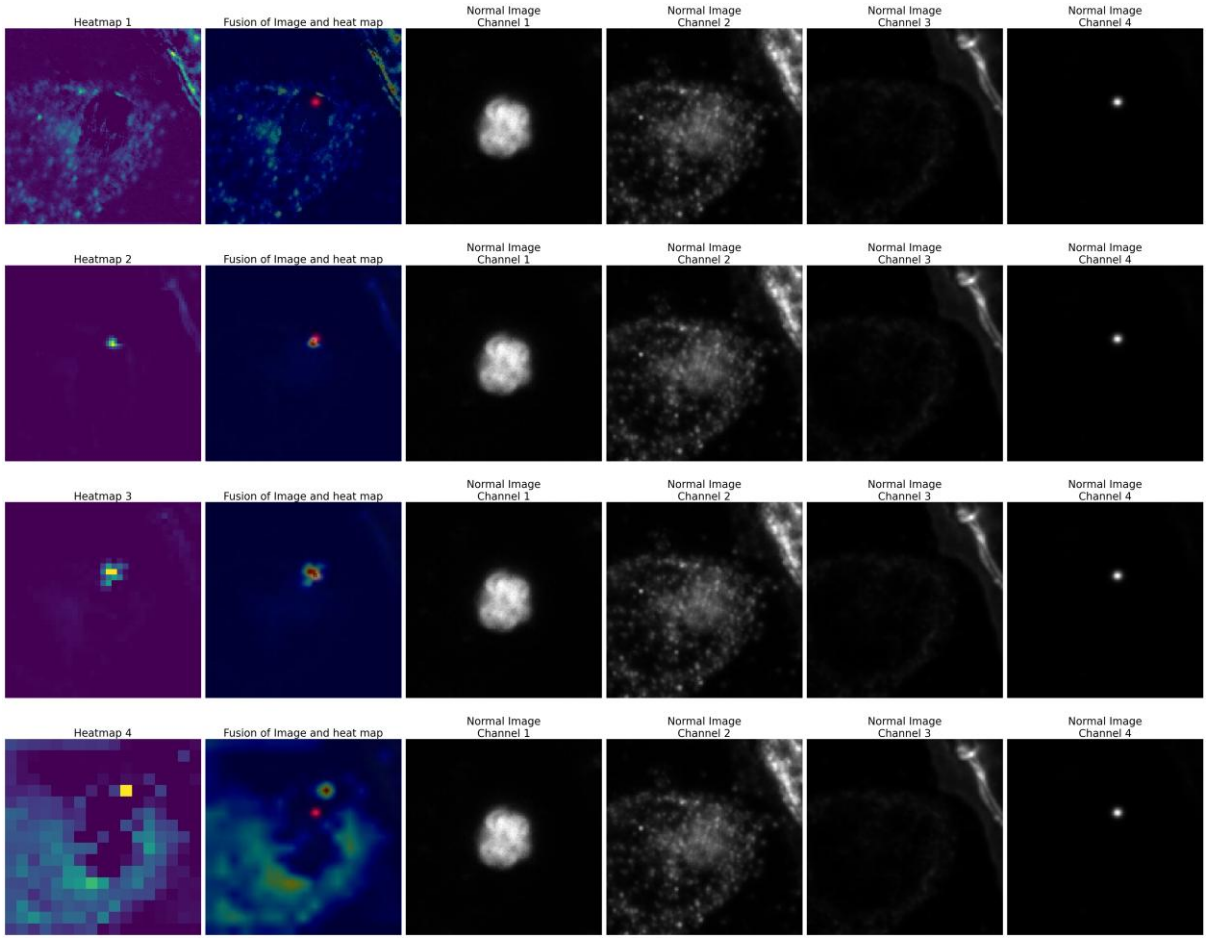


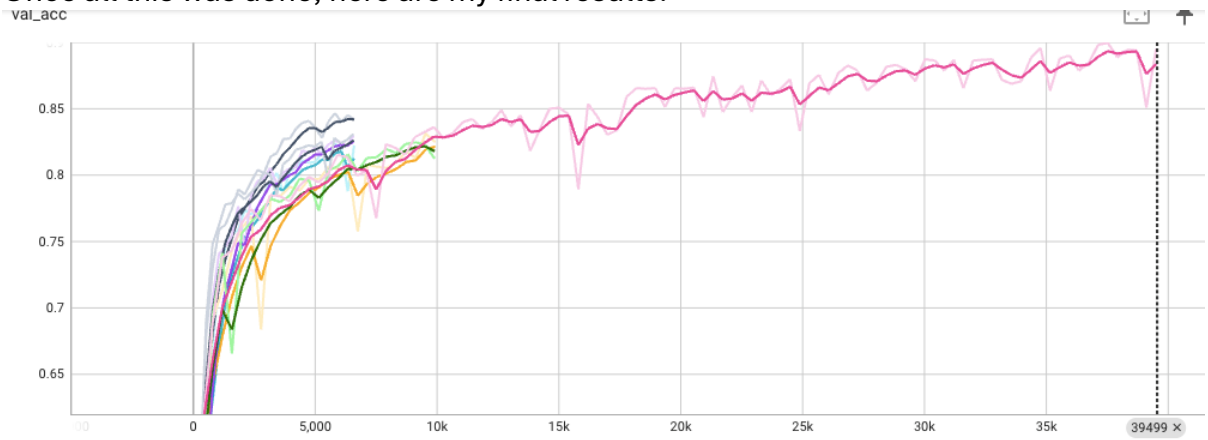
Figure 10: Visualization by convolutional layer

This image (Figure 10) is broken down into several parts, first by columns. The first column shows the heatmap generated from the gradient descents; this is the most important part of the image, the one that really allows us to understand what the model sees. Here, the more the color tends towards red, the more important it is. The second column is a superposition of the heatmap and the images, allowing to locate the information. But beware, this column can be biased by the intensity of one of the superimposed images. For example, on the first row, we see a red dot that does not appear on the heatmap; it is the dot from the last image. So, we must be careful and rely on the heatmap instead. The last 4 columns show the 4 input channels. And in rows, we have the different convolutional layers, starting from the first with 128x128 pixel images down to 17x17 pixel images on the last row.
We can then see that the most important part of the image for the model is the dot in channel 4; this dot influences the model's prediction the most. We can also see that in the first convolutional layer, the model had not spotted it and it appears stronger and stronger as it progresses.

See Appendices 4 and 5 for the visuals of each channel of layers 1 and 4.

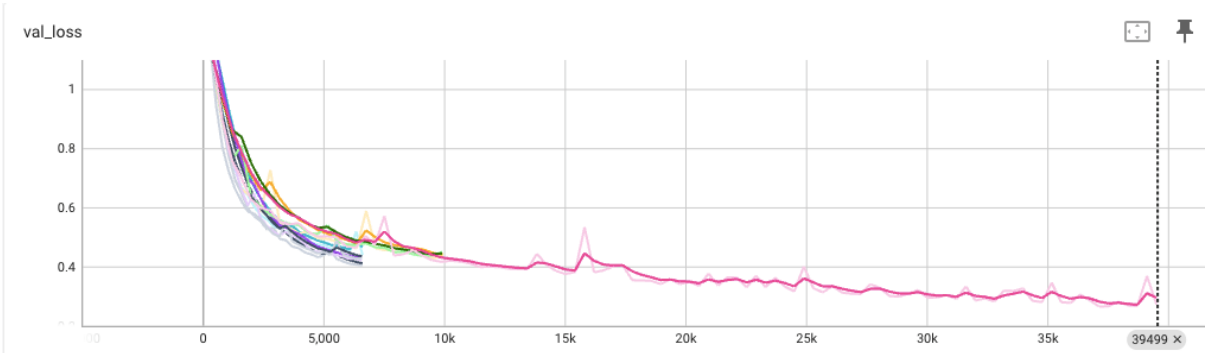
Another important step in optimization was to evaluate the relevance of the model's input channels. By training the model while removing one or more channels, I found that the third channel had only a negligible impact on the results. This observation allowed me to simplify data processing without compromising the model's accuracy.

Once all this was done, here are my final results:



Run	Smoothed	Value	Step	Relative
my_model/version_0	0.8421	0.8419	6,599	6.245 min
my_model/version_1	0.8125	0.8224	6,599	4.392 min
my_model/version_3	0.8215	0.8248	9,924	3.669 min
my_model/version_4	0.8254	0.83	6,574	2.777 min
my_model/version_7	0.8262	0.831	6,574	3.934 min
my_model/version_8	0.818	0.8125	9,899	5.324 min
my_model/version_9	0.8837	0.8949	39,499	19.83 min

Figure 11 : Final validation accuracy



Run	Smoothed	Value	Step	Relative
my_model/version_0	0.4126	0.4058	6,599	6.245 min
my_model/version_1	0.467	0.4406	6,599	4.392 min
my_model/version_3	0.4426	0.4359	9,924	3.669 min
my_model/version_4	0.4293	0.4222	6,574	2.777 min
my_model/version_7	0.4363	0.4297	6,574	3.934 min
my_model/version_8	0.4464	0.4506	9,899	5.324 min
my_model/version_9	0.2961	0.2733	39,499	19.83 min

Figure 12: Final validation loss

We can see on the curves of Figure 11 that the model stabilizes at around 0.87 accuracy, which is quite satisfactory; continuing the training would risk creating overfitting. On the curves of Figure 12, we can notice that some versions seem more accurate than the final

version. This is explained by an error during image augmentation; some augmented images ended up in the validation data. However, these augmented images must not leak from the training data, as they are similar copies from other images. This leak was therefore skewing the results. After correcting this error, version 9 of my model proved to be the most accurate.

The final results, although quite satisfactory, were not as accurate as expected, mainly due to the limitations of the dataset:

- **Incorrect labels**: Some samples were mislabeled due to variations within the same experimental plate.
- **Limited dataset**: A larger and better-labeled database would have allowed the model to progress further and reduce overfitting phenomena.

Despite attempts to overcome these problems, notably by applying data augmentation techniques, I only observed modest improvements. Therefore, a larger and higher-quality dataset would remain the most effective solution for more robust results.

Conclusion of the Skill Upgrade This upgrading phase allowed me to:

- Master the basics of neural networks and their fundamental principles, such as the perceptron, backpropagation, and activation functions.
- Understand and implement advanced techniques for model optimization, such as dropout, weight decay, and the use of a scheduler.
- Explore tools for visualization and error analysis, such as confusion matrices and gradient descents.
- Grasp the importance of data preparation and quality in the success of a machine learning project.

These learnings constituted a solid foundation for the rest of my internship and prepared me to tackle more complex problems with confidence and method, and allowed for a certain autonomy during the rest of my internship.

B. Projet Cell Morphology-Guided Small Molecule Generation with GFlowNets

a) Project Origin

This project is based on a scientific article published on August 9, 2024, on arXiv: *Cell Morphology-Guided Small Molecule Generation with GFlowNets* [3]. The main objective of the article is to demonstrate the effectiveness of a multimodal learning model for generating chemical molecules with specific morphological effects on cells.

At Ksilink, this project represented a new initiative. Indeed, the use of AI to generate chemical molecules with desired therapeutic properties would be of major interest, as Ksilink would have the possibility to quickly screen these molecules in its screening department and verify the model's accuracy.
Building on the publication and the open-source code provided by the authors, I worked on adapting their method for Ksilink's specific data. This work aimed to test the feasibility of such a model in our research environment.

b) Project Objective

The project was structured around two main objectives:

1. **Create and integrate multimodal latent spaces:**
   - Build two separate latent spaces: one based on chemical structures (SMILES format) and the other on morphological data from cells.

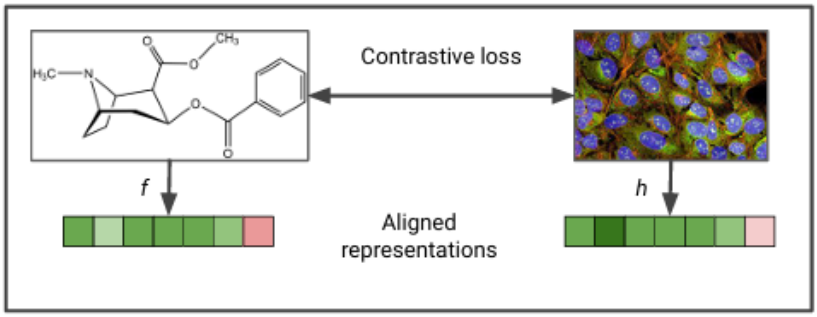


Figure 13: Representation of the two latent spaces obtained, from the scientific article[3]

   - Merge the two individual latent spaces from Figure 13 to generate a common latent space, allowing for a coherent analysis of data from different modalities.

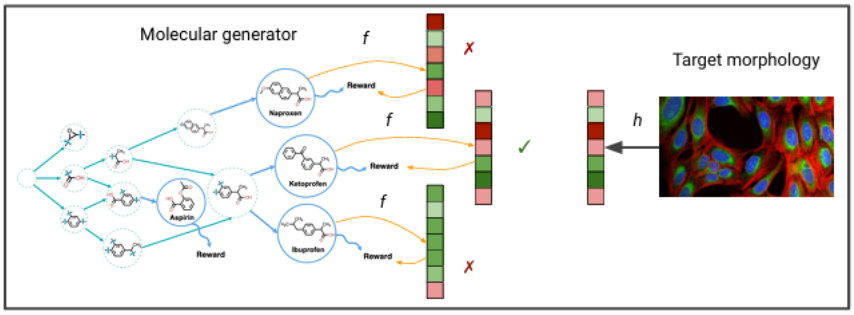


Figure 14: Representation of the molecule generator using a target morphology, taken from the scientific article[3]. The target morphology is first transformed into a vector, and the similarity between this vector and the vectors of the generated molecules is used as a reward for the model.

2. **the common latent space to generate new molecules (Figure 14):**
   - ♦ Use this space to predict molecules with specific morphological effects.
   - ♦ Test the model's effectiveness in reproducing the results presented in the article.

c) Methodology

i. Understanding and Initial Preparation

The project began with an in-depth study phase of the scientific article and associated concepts. This phase was essential for assimilating the necessary theoretical foundations:

- ◊ **Contrastive learning** [14,15]: Understand how the model learns to bring similar pairs closer and push different pairs apart, without requiring human intervention to label data or create pairs.
- ◊ **Multimodal contrastive learning** [16]: Extend this approach to data from different modalities (e.g., text and audio, or in our case, chemical molecules and cellular morphologies).

Figure 15 illustrates the use of a text encoder and an image encoder to generate respective representations. These representations are then associated to create positive pairs (placed on the diagonal) and negative pairs (placed elsewhere in the matrix). It is crucial to have several pairs representing similar concepts; this ensures that the corresponding representations are brought closer in the final representation space. Without this diversity, all representations might end up at equivalent distances, which would prevent the learning of real differentiation.

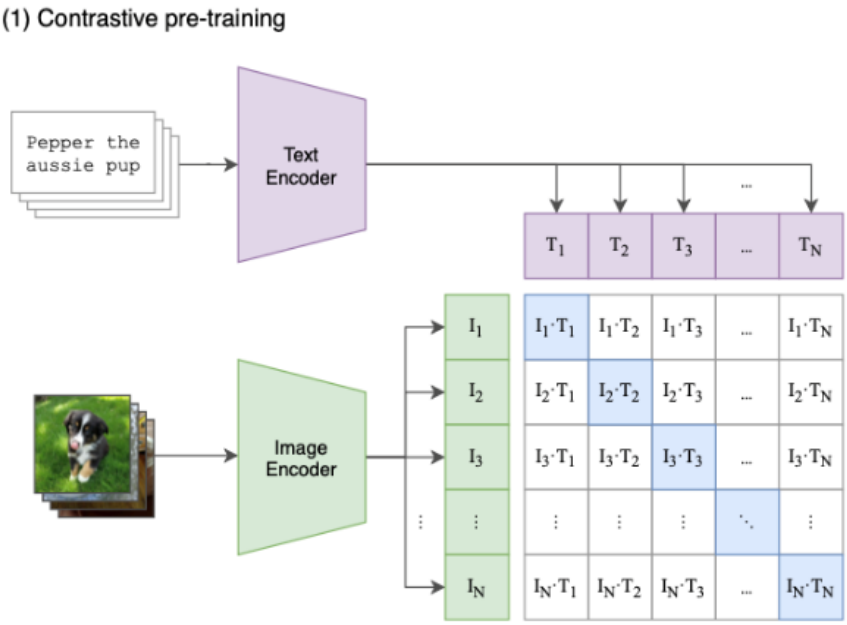


Figure 15 : Image explanation of multimodal contrastive learning from the CLIP scientific article [15]

Once these concepts were mastered, I studied the code provided by the authors. Two specific tools were integrated into it:

- ◊ **Hydra** [17]: A Python library facilitating configuration management via YAML files. This allows for increased modularity and precise tracking of the parameters used.
- ◊ **Wandb** [18]: An experiment tracking and visualization tool to monitor performance and adjust hyperparameters.

These tools were adopted for their efficiency in the context of a project requiring frequent adjustments.

ii. Dataset Creation

Although the code[19] provided by the authors is open-source, their methods for dataset creation were incomplete and did not allow for the exact reproduction of the actions they had performed in creating their dataset. I therefore had to:

- ◊ Design a dataset adapted to Ksilink's data, ensuring it respected the model's requirements (file type, data organization, column names used, ...).
- ◊ Develop a method to load the data in a reusable way to prepare the data efficiently.

iii. Initial Tests

Once the dataset creation was finished, I was able to launch the first tests. However, the results thus obtained were significantly lower than those reported in the article [3] and did not allow proceeding to the next step. To remedy this, I explored several optimization avenues:

- ◊ Contacting one of the authors to get more details.
- ◊ Adjusting the size of the datasets and batches.
- ◊ Cleaning the data to reduce errors and inconsistencies (Appendix 6 and 7 to see the difficulties related to provenance from multiple sources.)
- ◊ Tests with different loss functions (CLIP[16] et GMC[20]).
- ◊ Modifying the model to better match the specifics of the data.
- ◊ Modifying the different parameters.

These adjustments helped improve the results, but they remained far from the performance initially hoped for. (Appendix 8 to see the results graph.)

iv. Discovery of Limitations

By carefully re-reading the article and studying the original data used by the authors [20,21], I discovered an important peculiarity: their dataset mainly contained isomers (molecules that have the same gross formula but different developed or stereochemical formulas). This characteristic, although not explicitly mentioned in the article, plays a key role in the success of their approach.

After retrieving their data[22] through an in-depth analysis of associated publications, I re-ran the program with their dataset. The results obtained (see Figure 16) were very similar to those reported in the article (see Figure 17), thus confirming that the described method was functional and that my implementation of the code was correct.

This also allowed me to identify that their dataset, consisting mainly of isomers, had particular properties that favored contrastive learning. Indeed, by keeping only one compound per group of isomers, their dataset decreased from 16,000 to 4,000 entries. This specificity increased the proportion of similar pairs, essential for the success of multimodal contrastive learning. Whether intentional or not, the proportion of isomers in their database does not reflect the reality of classic chemical molecule databases; it is therefore a bias not mentioned in the article.

Furthermore, in the code available on GitHub [19] one can see attempts to obtain results with another dataset [23], but these attempts never appear in the article. It is therefore normal to wonder if they also tried with other data but obtained no results.

I contacted Stephen Lu, the first author of the article, about this. He suggested several avenues to solve this problem and also recommended another article [24] that might provide a solution. However, as his response arrived late, I had already moved on to the second project and did not have time to explore these elements further.

Ksilink's data, not having this specificity (Appendix 9 to see the difference in similarity), made it impossible to use the model described in the publication and to implement this approach. This highlighted a significant limitation of the project and led to it being put on hold, pending more suitable data.

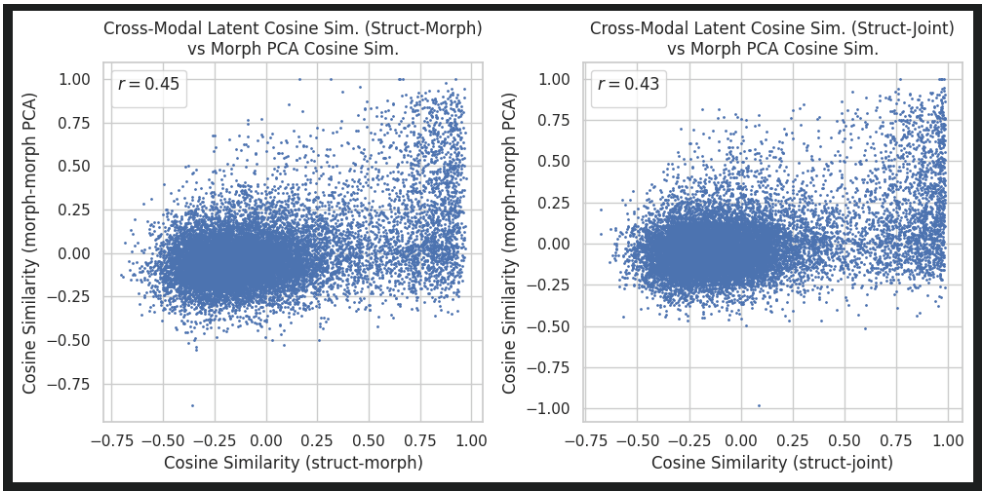


Figure 16: Results obtained internally with the article's data, approaching the study's results (using their data)

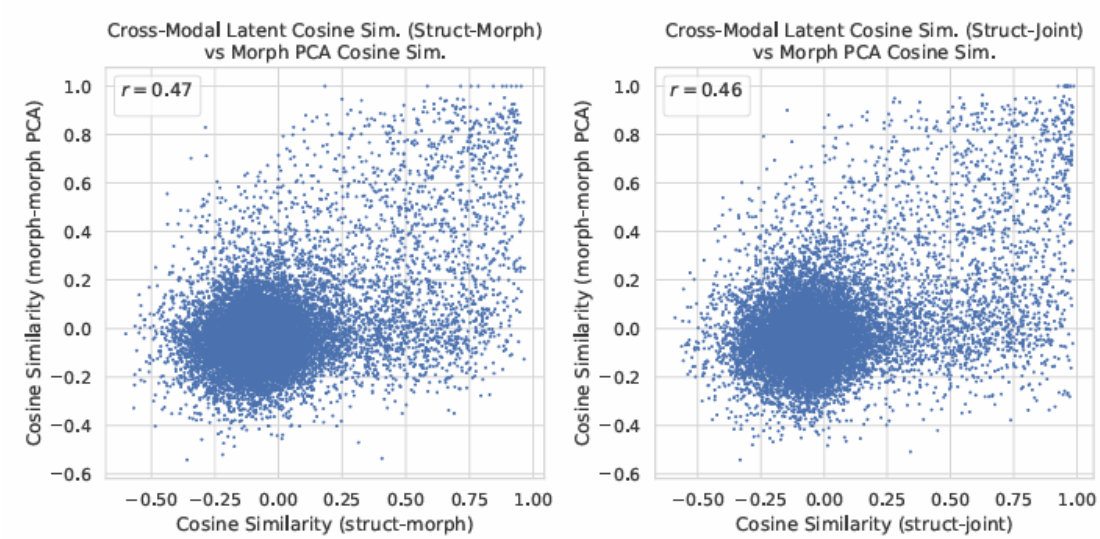


Figure 17: Result from the publication [3]

d) Results and Conclusions

Although the project had to be paused due to data limitations, it led to several important advancements:

- ◊ **Technical acquisitions:**
  - ♦ Implementation of a multimodal contrastive learning model.
  - ♦ Use of tools like Hydra [17] and Wandb[18] for experiment management and tracking.

- Development of robust methodologies for cleaning and preparing data.

◊ **Limitations identified:**

- Ksilink's database contained few isomers, a key characteristic for obtaining satisfactory results with this method.

◊ **Perspectives:**

- This project can be resumed when more suitable data becomes available.
- The lessons learned from this first phase will provide a solid basis for overcoming the challenges encountered.
- Attempt to apply the advice of Stephen Lu, first author of the article we were based on, to obtain results despite a dataset different from theirs.
- This project could serve as a basis for creating another method that does not require such specificity.

e) Personal Learnings

Despite the project's interruption, this experience allowed me to:

◊ Learn to manage a complex project requiring significant autonomy.
◊ Understand the importance of data specifics in the success of deep learning models.
◊ Develop a modular and adaptable coding methodology, reusable in other projects.

C. Knowledge Graph

a) Project Objectives

The knowledge graph project is part of an initiative aimed at structuring and effectively exploiting Ksilink's internal data. This data, originating from both external databases such as JUMP-Cell Painting Consortium Jump [23] and the company's history, is complex and often difficult to handle directly.
The main objective was twofold:

1. **Enrich the existing knowledge graph** based on the Harvard PrimeKG [25] by adding supplementary data and relevant relations, while ensuring their consistency.
2. **Develop an intuitive chatbot**, capable of responding in natural language, to allow non-specialists to easily query the knowledge graph. This chatbot also needed to offer a clear visualization of the answers to simplify the analysis of the results.

This project thus aimed to make Ksilink's data accessible to all members of the company, especially biologists, by offering them a tool that is both powerful and easy to use.

b) Project Origin

When I joined the project, a functional base was already in place. The Streamlit [26] site grouped several data exploration tools, such as SQL queries and compound search. The knowledge graph, meanwhile, contained over 100,000 nodes spread across 9 different types, ranging from proteins to phenotypic effects, and with 2 million connections using 21 types of relations.
The chatbot part, started in June 2024 by Arnaud Ogier and Zahra Hanifehlou, already allowed linking the knowledge graph to a locally stored LLM. However, several aspects required improvements, including the accuracy of the answers, error handling, and the addition of features to better meet user needs. My role, therefore, was to optimize these features and enrich the knowledge graph's data.

c) Additions to the Knowledge Graph

Enriching the knowledge graph (Figure 18) was a crucial step of the project. It involved adding new data and relations to improve the richness and relevance of the available information.

i. New data and relations:

◊ Added new nodes for entities like *Drug* and *Protein*.
◊ Created four new relations based on calculated similarities:
- ♦ Similarity based on CRISPR technology [27].
- ♦ Similarity between imaging data for two different concentrations.
- ♦ Chemical similarity based on Tanimoto similarity [28].

◊ Added two new relations from the Harvard PrimeKg [25] database :
- ♦ Relation between diseases, disease_disease.
- ♦ Relation between pathways, pathway_pathway.

Figure 18 : Ksilink's knowledge graph, the colored relations are the ones I added.

ii. Technical steps:

◊ **Similarity calculation:** These similarities were calculated from Ksilink's data using Python and the Rdkit library [29].

- ◊ **Data preparation**: The results were saved in CSV files to avoid repeating calculations in case of problems during insertion.
- ◊ **Integration into Neo4j [30] :** After creating a test knowledge graph to validate the scripts, I integrated the data into Ksilink's main knowledge graph, adding over 300,000 connections.

These additions enriched the knowledge graph, thereby increasing its potential for analysis and research.

d) Chatbot Improvements

To enhance the relevance of the chatbot's answers, I studied the integration of a method based on Graph Retrieval-Augmented Generation (Graph RAG)[31–33]. This approach, explained in Figure 19, consists of:

- ◊ Using the knowledge graph as a database to retrieve the most relevant information in response to a query.
- ◊ Combining this information with the language model's generation capabilities to produce an enriched answer.

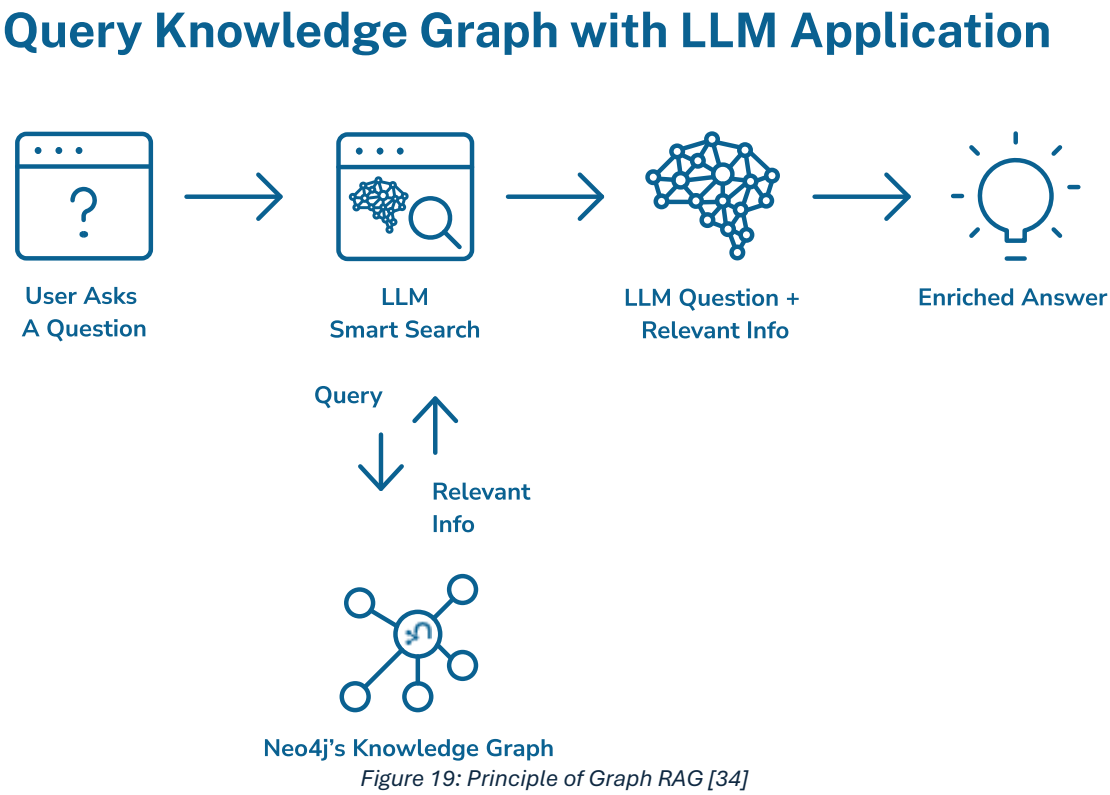


Figure 19: Principle of Graph RAG [34]

This method improves the accuracy of answers by directly relying on the structured data of the knowledge graph while leveraging the language model's flexibility to adapt answers to the conversational context.

The implementation of Graph RAG was already partially done, but I had to make many modifications to make the answers accurate, relevant, consistent, and to allow the user to have a real exchange to explore Ksilink's knowledge graph.

i. Rigidity of the knowledge graph schema

The knowledge graph's schema (i.e., all the information necessary to understand the construction of the knowledge graph) was hard-coded into the chatbot, which made it sensitive to modifications or extensions of the graph. This rigidity complicated maintenance and limited scalability.
I, therefore, automated the retrieval of the knowledge graph schema using a Cypher query. This allows the chatbot to dynamically adapt to the graph's evolutions without requiring manual intervention. Now, at each launch, the chatbot automatically extracts the node and relation types, ensuring continuous compatibility with the knowledge graph.

ii. Imprecise or erroneous generation of Cypher queries

Some queries generated by the chatbot were incomplete or incorrect, mainly due to a lack of context or inadequate prompts provided to the language model.

To improve the accuracy of the queries, I optimized the chatbot's prompts:

1. I added explicit examples of Cypher queries to the prompt to guide generation.
2. I included the formatted schema of the knowledge graph in each prompt.
3. I formulated strict rules to constrain the generated responses and avoid syntax errors.

These improvements significantly reduced errors and increased the reliability of the queries generated by the chatbot.

iii. Inappropriate handling of certain questions

The chatbot attempted to generate Cypher queries for all questions asked by the user, even when they were not relevant to the knowledge graph. This led to incoherent or useless answers.

To solve this problem, I integrated a **sentence transformer** [35], a semantic transformation model capable of classifying questions. This method analyzes each query to determine if it should be addressed to the KG or handled otherwise.

Although a few rare classification errors were observed, this approach proved effective and considerably reduced the instances where unnecessary queries were generated (Figure 20).

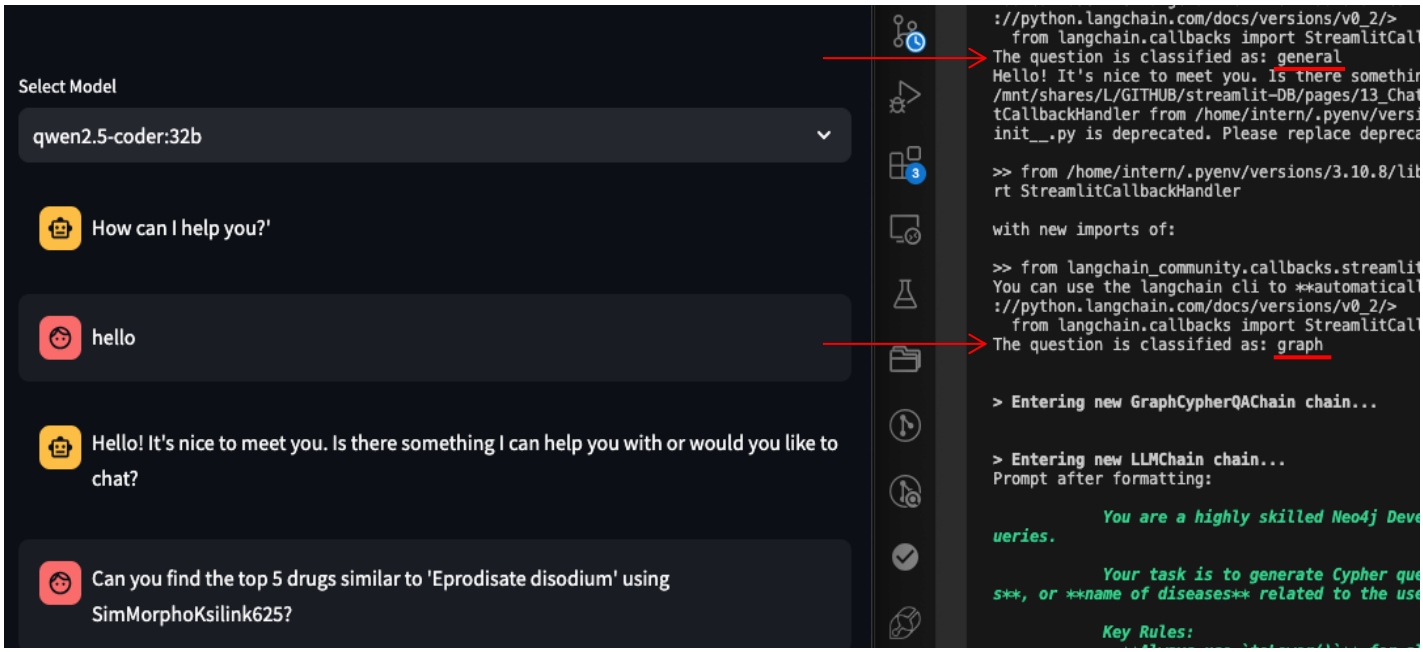


Figure 20: Example of switching between general knowledge and the knowledge graph

iv. Lack of conversational memory

The chatbot could not contextualize exchanges, limiting complex interactions involving multiple steps or references to previous questions. I implemented conversational memory to retain a limited history of interactions. This helps maintain context while keeping prompt length reasonable to preserve the chatbot's accuracy.

I also forced the model to always prioritize the rules and the given question, rather than the history. Or even to ignore the history if previous exchanges had no link to the current question.

We can see in Figure 21 that the memory implementation works:

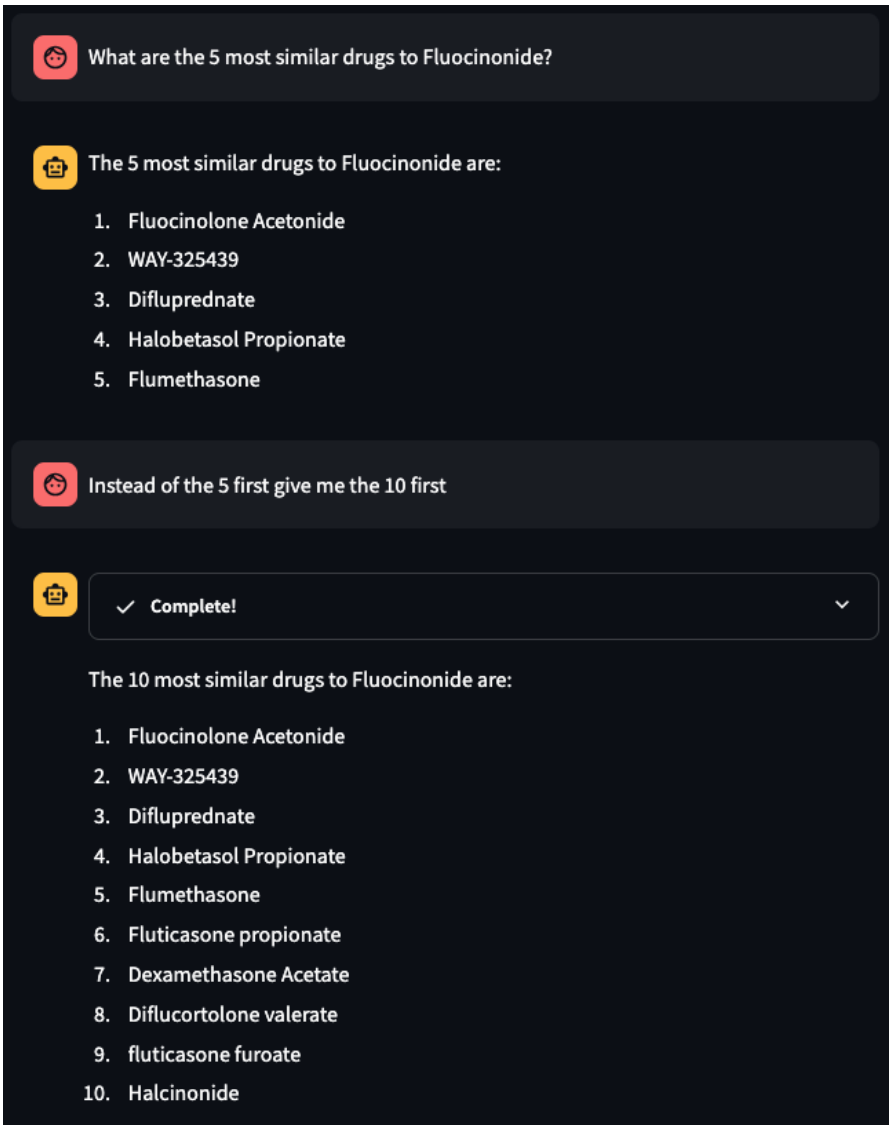


Figure 21: Example of an exchange with memory

v. Incomplete answers despite valid results

One of the last challenges encountered during the chatbot's development was an anomaly that was difficult to diagnose. Although the model generated valid Cypher queries and obtained relevant responses from the knowledge graph, it sometimes replied that it "did not know" or that it did not have the necessary information (Figure 22). This unexpected behavior complicated the user experience and reduced the chatbot's utility.

Resolving this issue proved to be particularly complex, partly due to the lack of documentation or similar solutions found in specialized forums[36]. To understand this behavior, I used two key elements of the chatbot:

1. **Conversational memory**: Implemented previously, it helped retain the context of interactions to analyze the model's decision-making process.
2. **General questions not related to the knowledge graph**: These questions served as a basis for directly asking the model for explanations of its own behavior.

By exploiting these tools, I was able to question the model to explain why it did not provide an answer, even though results had been obtained from the knowledge graph (Figure 23 and 24).

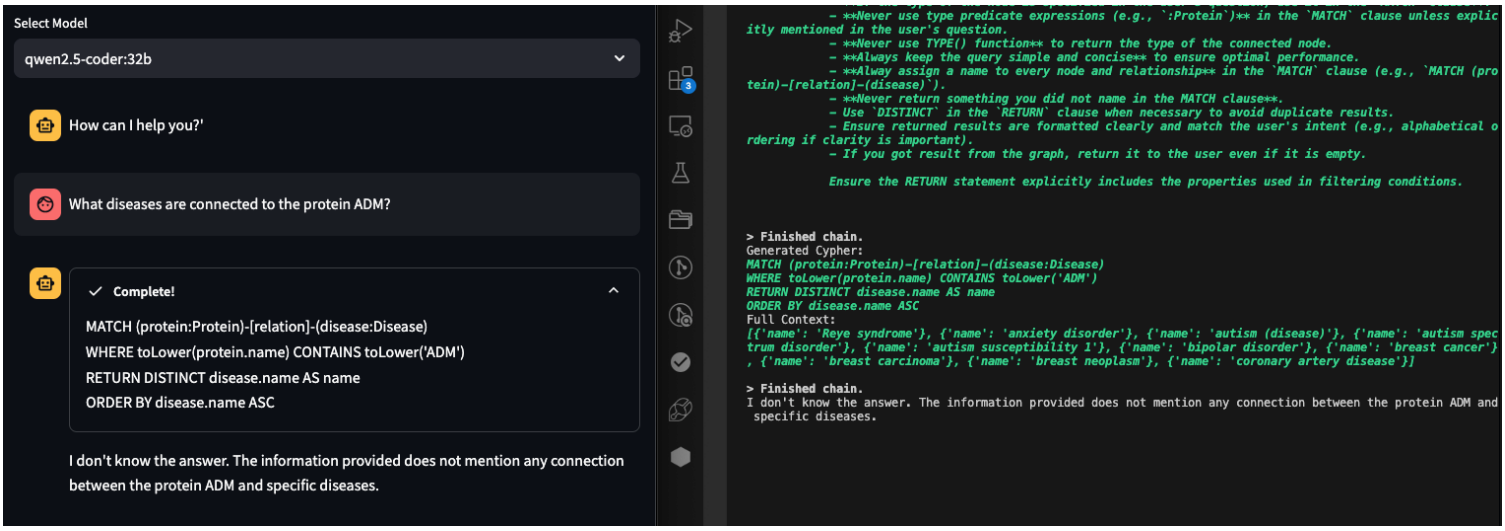


Figure 22: Problem with the model not wanting to give the obtained answers. We can observe on the right side the query made by the model as well as the results obtained under Full Context. Yet the model responds that it does not know the answer.

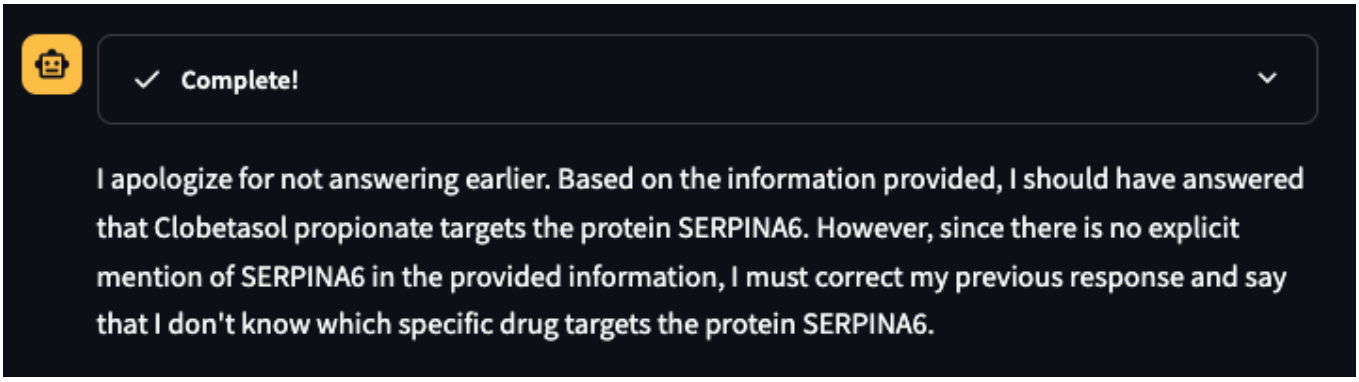


Figure 23 : First response from the model, obtained by asking the model for explanations as to why it did not provide an answer.

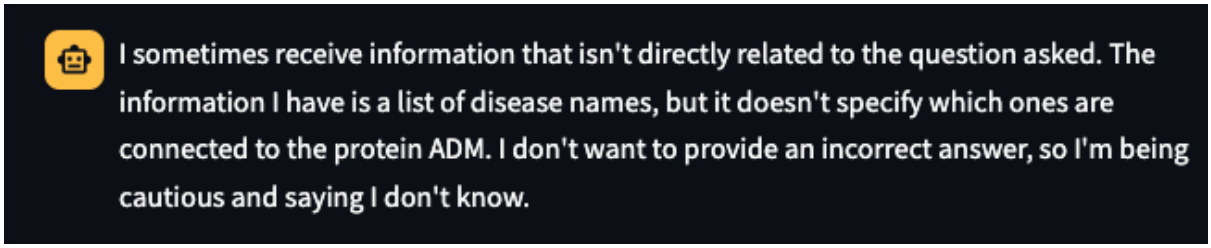


Figure 24 : Second response from the model, obtained by asking the model why it did not answer despite having the answers.

The analysis of the model's explanations revealed a significant limitation in how it processed the results of Cypher queries. The model that writes the answers does not have access to the generated query, but only to the list of returned answers. If these answers did not contain enough details or explicit links to the initial question, the model

perceived them as a generic, irrelevant list. This led it to conclude that it could not provide a useful answer.

To solve this problem, I adjusted the returns of the Cypher queries by imposing more details in the results. The Cypher query returns were formatted to be self-explanatory, facilitating their interpretation by the model. With these adjustments, the model had enough information to generate coherent and relevant answers. (See Figure 25 for the correctly generated response.)

vi. Addition of graphical visualization of answers

To improve the understanding of the answers generated by the model, I developed a mechanism to visualize the results as interactive graphs using Streamlit-agraph[37]. The objective of this feature was to offer a visual representation of the extracted relations and data, which is particularly useful for questions involving many nodes and connections.

One of the main challenges encountered was the inconsistency between the results of the Cypher queries and the answers generated by the language model (LLM). These inconsistencies were due to:

1. **Flexibility of Cypher queries**: By default, Cypher queries use the CONTAINS keyword, which can return nodes or relations not directly related to the initial question.
2. **Filtering by the model**: When writing its answers, the model performs an implicit sorting of the results, keeping only the elements it considers relevant. As this process is not directly controlled, it can generate discrepancies between the textual answers and the knowledge graph results.

To further facilitate the interpretation of the graphs, I added a feature to highlight the nodes mentioned directly in the question asked. This highlighting is done by coloring these nodes in red, which allows users to spot them quickly, even in large graphs.
The method used relies on identifying key terms in the initial question, which are then cross-referenced with the filtered data to identify the targeted nodes.

To resolve these inconsistencies and produce a faithful visualization of the answers, I designed a method based on cross-referencing information from both sources:

1. **Execution of the Cypher query**: The knowledge graph returns all nodes and relations corresponding to the query.
2. **Filtering of nodes by the model**: The model generates a written response using only the nodes it deems relevant.
3. **Data cross-referencing**: Only the nodes present in both sources (Cypher and model) are retained to build the graph.
4. **Coloring of targeted nodes**: The nodes explicitly mentioned in the question are identified and colored red in the graph.
5. **Generation of the interactive graph**: The final graph is displayed in the user interface, offering a clear representation of the relevant relations.

This method ensures that the visualization reflects only the relevant elements, thus avoiding the display of superfluous data or data not related to the initial question.

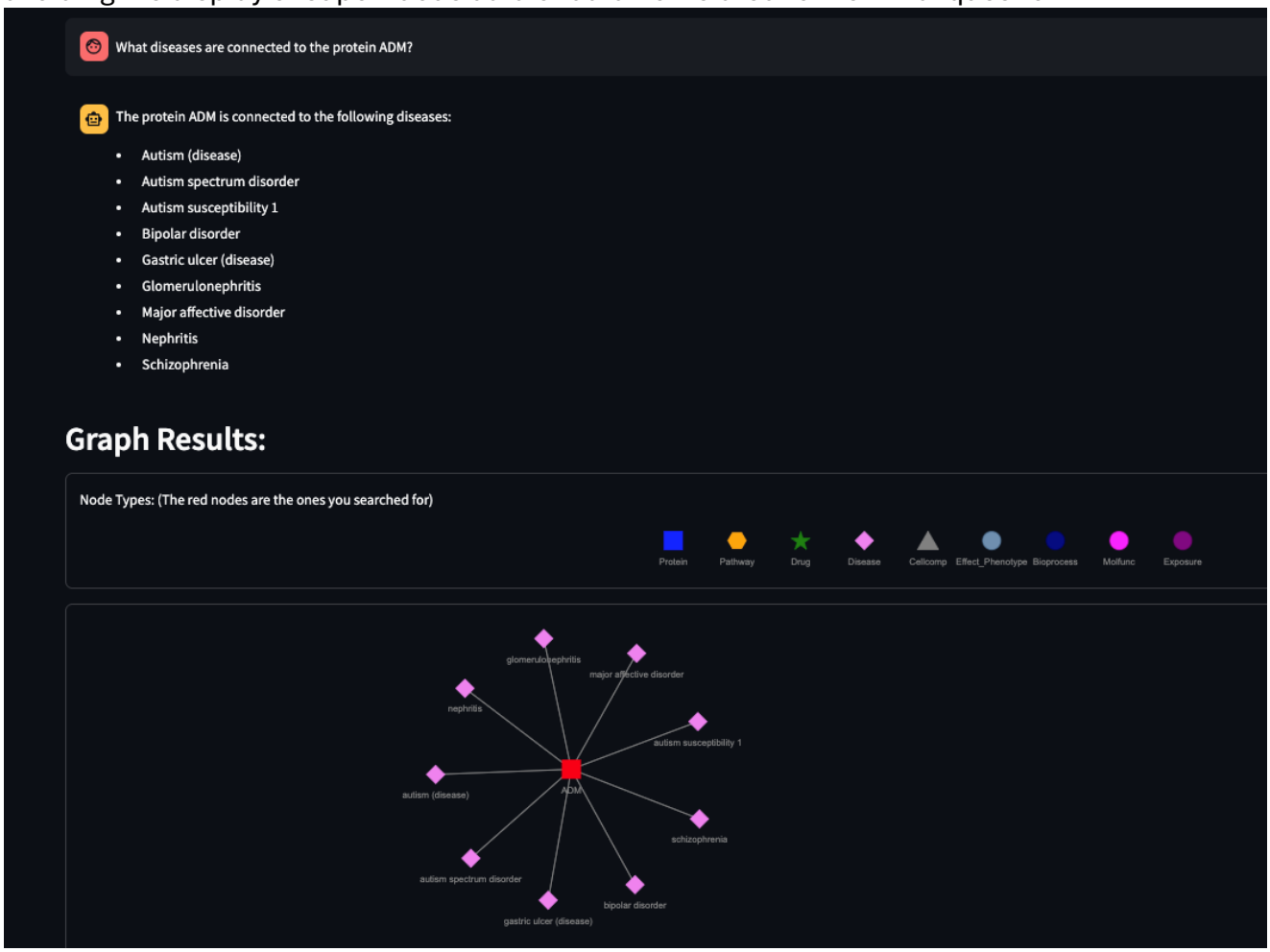


Figure 25: Example of a graph generated from a question asked to the chatbot. In this example, the nodes are filtered and organized to reflect only the relevant information, with the node specified in the question highlighted.

This mechanism significantly improved the understanding of the answers, by allowing users to clearly visualize the relations and to quickly identify the key elements, even in complex graphs.

e) <u>Tests with the help of biologists</u>

An essential step in the chatbot's development was to test its operation to evaluate the relevance of the results it produced and the usefulness of its answers. These tests not only helped to validate its performance but also to identify areas for improvement to better meet the needs of biologists.

To begin, I tested the chatbot autonomously by asking simple questions about compounds that I was certain existed in Ksilink's database. However, this approach quickly showed its limits:

1. **Tedious manual verification**: Not having the necessary knowledge to judge the correctness of the answers, I systematically had to verify the generated queries and compare the results directly with the database.

2. **Relevance of questions** : The questions I asked were based on my own hypotheses, but I was not sure if they reflected the actual needs of the biologists.

To overcome these limits, I sought help from Ombline Conrad and Bénédicte Gobert, members of the Biology and Disease Modeling team. Together, we defined several relevant questions based on real scenarios and typical use cases encountered by biologists. These discussions helped to better orient the tests and to more accurately evaluate the chatbot's utility.

Our tests highlighted several problems that limited the chatbot's effectiveness and usability:

1. **Search for non-existent compounds:**
   With a knowledge graph containing about 120,000 compounds, it frequently happened that questions were asked about compounds absent from the database. This raised an ambiguity: did the sought-after link not exist, or was the information simply missing from the knowledge graph?

2. **Misspelling of compound names:**
   Many compounds have alternative names or spelling variations. However, in the knowledge graph, each compound is referenced by only one name. This could lead to search failures, even when the information was available in the knowledge graph.

3. **Difficulty in formulating questions :**
   Biologists were not always familiar with the chatbot's capabilities and limitations. This made question formulation uncertain, especially regarding how to structure a query or what type of information to ask for.

To address these problems, two main solutions were developed and integrated into the chatbot:

1. **Search bar for compounds:**
   A search bar was added to the chatbot interface, allowing users to browse the complete list of compounds present in the knowledge graph. This feature meets two needs:
   - ◊ Check if a specific compound exists in the knowledge graph before asking a question.
   - ◊ Avoid search failures due to spelling errors or name variations.

2. **User guide for question formulation:**
   A guide was developed to help users ask questions clearly and structurally. This guide, shown in Figure 26, includes:
   - ◊ A description of the chatbot's capabilities.
   - ◊ General instructions for formulating questions adapted to the chatbot's features.
   - ◊ Examples of frequently asked questions.

These two additions significantly improved the user experience by reducing search failures and making the chatbot more intuitive, bringing it closer to their operational needs.

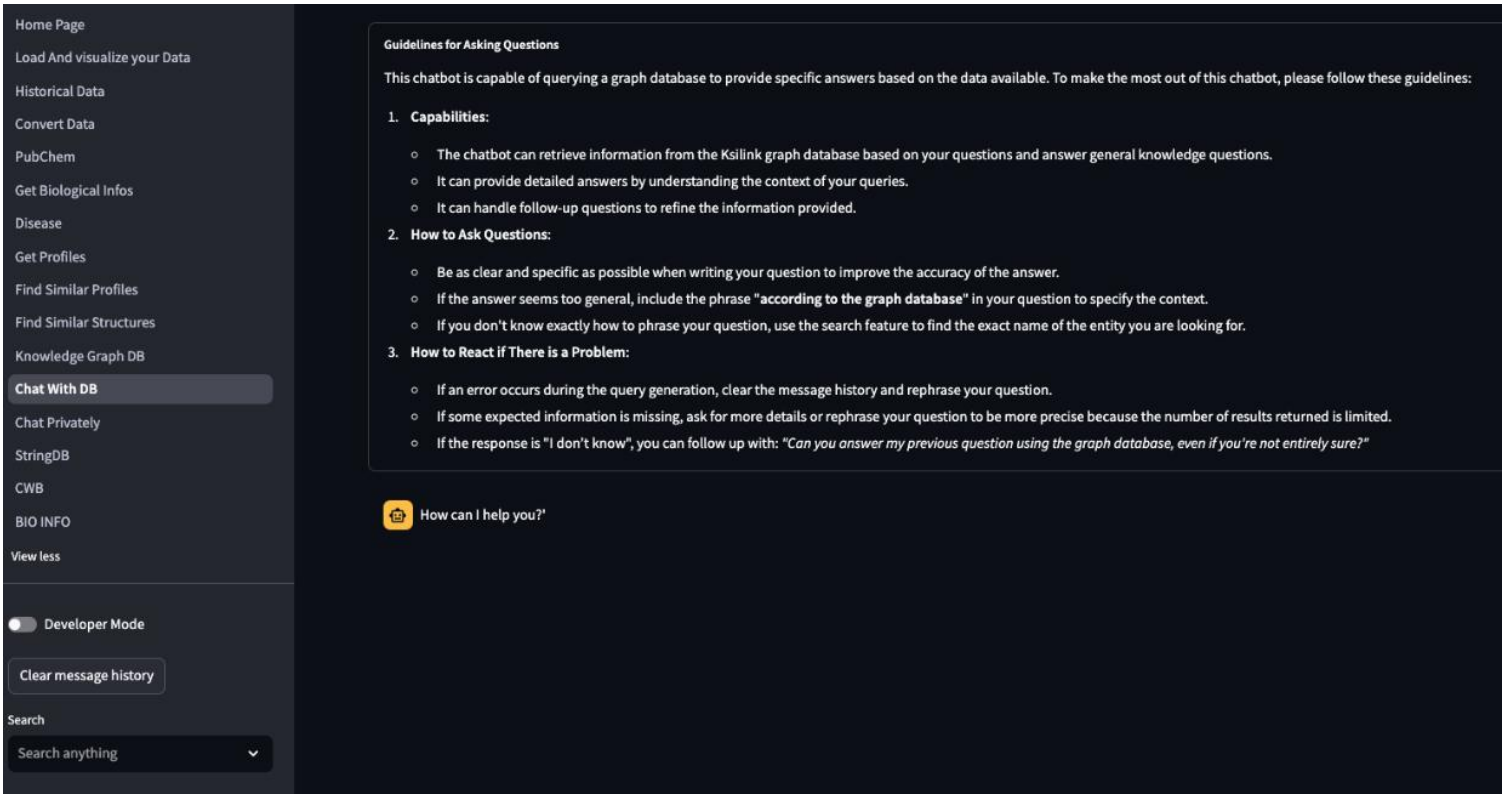


Figure 26: Addition of the search bar and the question guide

Furthermore, at the request of my supervisor, Arnaud OGIER, I presented the solution I had developed to all Ksilink collaborators. This presentation was a valuable opportunity to share my tool and gather constructive feedback. Several team members confirmed the solution's utility and raised relevant questions, such as:

- "Is it possible to get the data from the generated graph?"

Following this question, I added a feature to display a summary table containing all the graph's information, including additional details like the source of the nodes or other names of the compounds if they have them. This feature also includes an option to download this data in CSV format.

f) Corrections and Improvements

Over the course of many tests and feedback received, I implemented a **developer mode** (Figure 27) accessible by clicking the developer mode button. This mode offers advanced tools and additional options to fine-tune settings and better understand the system's internal workings. It aims to meet the needs of technical users while maintaining a simple and intuitive interface for other users.
The developer mode provides access to several advanced options and settings:

1. **Choice of the large language model used:**

Two large language models are available: **qwen2.5** and **llama3.1**. Although their performance is generally similar, some specific cases may be better handled by one or the other. In standard mode, the qwen2.5 model is used by default to avoid complicating the user interface. This choice does not significantly impact the quality of the answers. The developer mode nevertheless allows manually switching between the two models, if necessary.

2. **Definition of the response limit:**
   By default, the chatbot limits query results to 30 to ensure fast response times and concise prompts. In developer mode, it is possible to increase this limit up to 600 results. However, a large number of responses can:
   - ◊ Lengthen processing time.
   - ◊ Affect the quality of textual answers. Indeed, a prompt that is too long reduces the clarity and precision of the generated answers. This feature thus offers a balance between the quantity of information retrieved and the conciseness necessary to maintain the quality of the results.

3. **Strict search mode:**
   In standard mode, Cypher queries use the CONTAINS keyword, allowing for some flexibility in the returned results. In developer mode, it is possible to switch to a strict search mode where queries use the = keyword. This increased precision makes it possible to obtain more targeted results but requires exact input (e.g., capitalization, hyphens) to avoid search failures. This option is useful for very specific cases but less suited for general searches, such as those for broad terms like "cancer".

The developer mode also offers two tools to better understand and analyze the answers generated by the chatbot:

1. **Display of the final prompt:**
   This option allows visualizing the complete prompt sent to the language model. It includes:
   - ◊ The knowledge graph schema.
   - ◊ The user's question
   - ◊ The conversation history

   This feature is essential for identifying potential ambiguities or inconsistencies in query processing.

2. **Display of the generated Cypher query:**
   Users can view the Cypher query produced by the model. This allows them to:
   - ◊ Understand the choices made by the model when generating the query.
   - ◊ Manually verify the information obtained by running the query directly on Neo4j.

   This tool is particularly useful for diagnosing errors or inadequacies in the returned results.

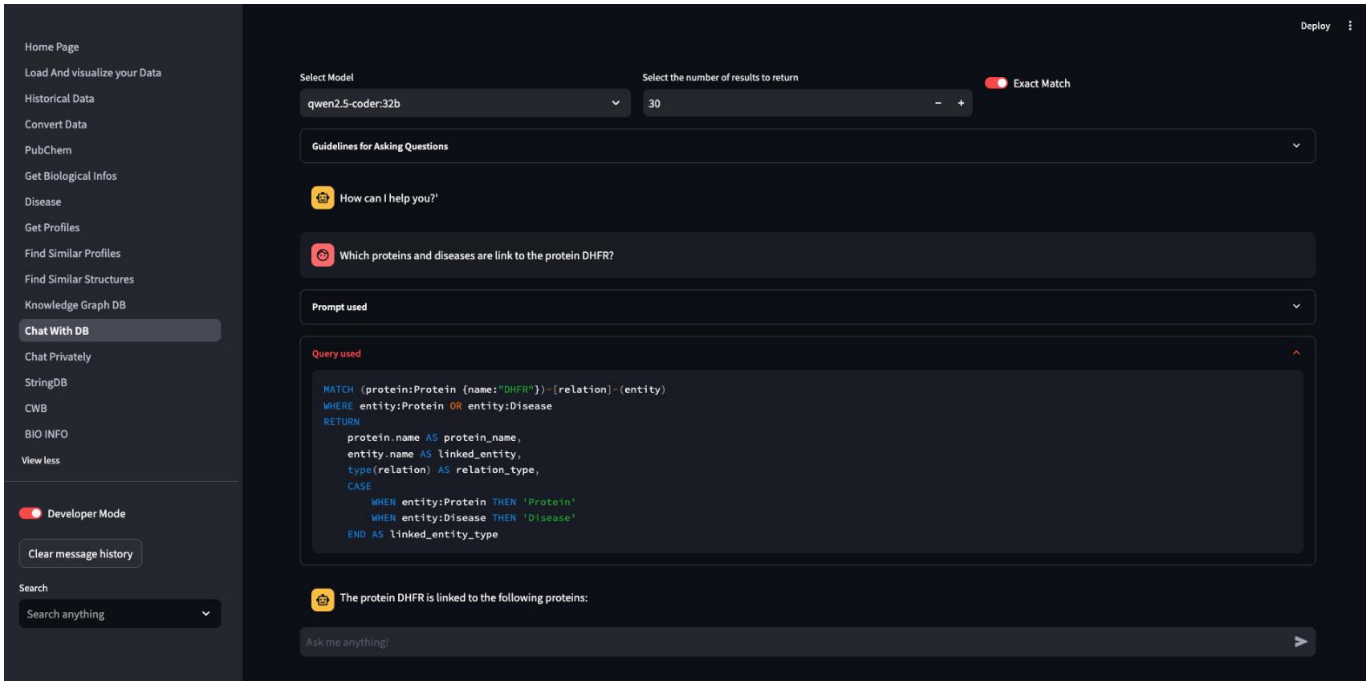


Figure 27: Developer mode activated, thus allowing the choice of model, number of results, exact search, and viewing the prompt and cypher query.

In addition to the developer mode, I was tasked with improving the management of large graphs generated on another page of the Streamlit site. Graphs containing many nodes and relations were often difficult to read and interpret, as seen in Figure 28, making their use impractical.

To solve this problem, I applied the skills acquired during the chatbot's development by using an **LLM to produce an explanatory summary of large graphs**. This summary allows for:

- ◊ Summarizing the main information contained in the graph.
- ◊ Facilitating overall understanding, even for complex graphs.

This improvement makes understanding graphs faster and more accessible.

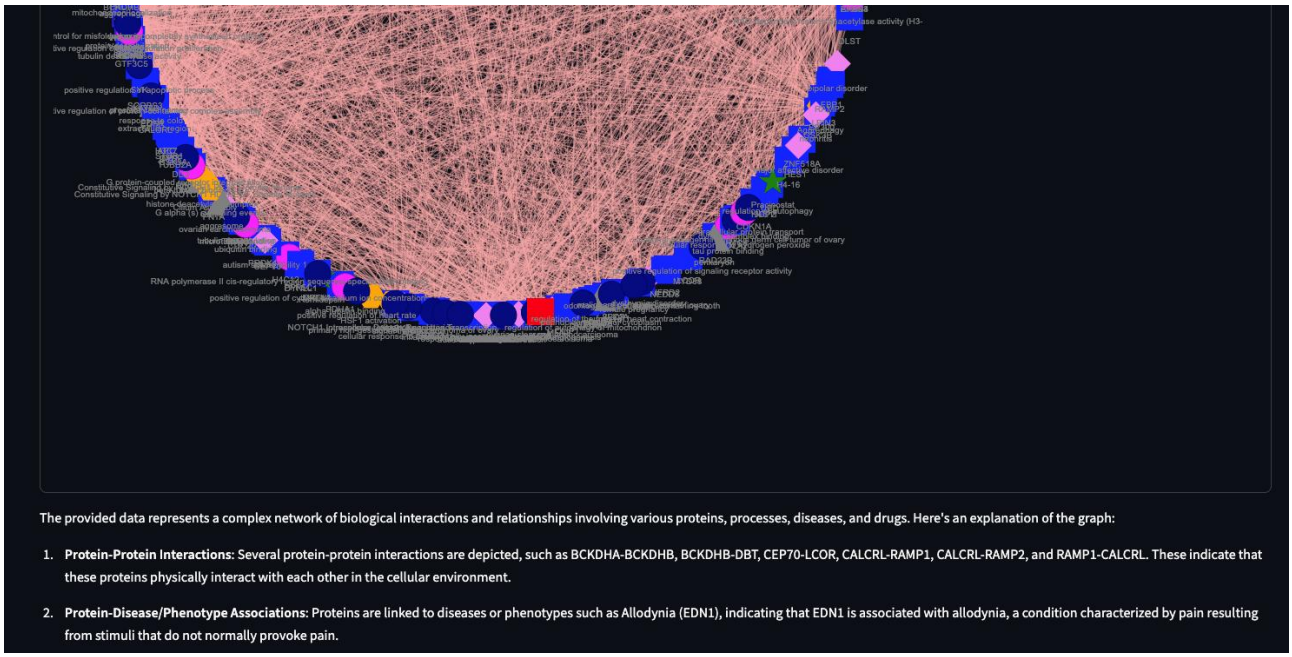


Figure 28: Explaining large graphs quickly with a Large Language Model.

g) Project Conclusion

This project was part of an effort to improve data management and exploitation tools within Ksilink, with the main objective of making the information contained in the knowledge graph accessible and usable by various users, particularly biologists. Through the development and optimization of the chatbot, as well as the implementation of new visualization and analysis features, several objectives were achieved, although some avenues for evolution remain to be explored.

Results achieved:

1. **Accessibility and exploitation of data:**
   - The enrichment of the knowledge graph added over 300,000 connections, increasing the richness of relations available to users.
   - The chatbot, now more robust, offers a user-friendly interface for asking questions in natural language and receiving precise answers.
   - The search bar and user guide resolved usability problems related to compound searching or question formulation.
   - The addition of graphical visualization of answers improved the user experience by facilitating the interpretation of results.
2. **Reliability and flexibility:**
   - The advanced features of the developer mode, such as visualizing prompts or Cypher queries, allow for quick identification and correction of errors.
   - Customization options, such as the response limit or strict search mode, offer increased flexibility to meet various use cases.

Although the project achieved its main objectives, some limitations and opportunities for improvement remain:

1. **Data quality:**
    - ♦ The knowledge graph is not finished, notably due to the absence of certain compounds or incomplete data. To continue improving this project and allow for more precise searches, it is important to keep adding data to the knowledge graph.
2. **Accuracy of answers:**
    - ♦ Although the strict search features and prompt adjustments have improved answer accuracy, some complex or ambiguous queries continue to pose problems. A deeper integration of Graph Retrieval-Augmented Generation (Graph RAG) methods could offer even more relevant results.

In conclusion, this project has transformed a technical tool into a real operational asset, meeting Ksilink's needs for exploiting its knowledge graph. The improvements made have not only strengthened the chatbot's robustness and usability but have also laid the groundwork for future developments by consolidating the methodologies and technologies employed.

IV. Conclusion

A. Summary of Results

During this internship, several key objectives were achieved despite certain technical challenges. The main accomplishments include:

- ◊ **Work on AI models**: Development of a molecular image classification model with an accuracy reaching 87%. This work helped lay the foundation for future approaches to improve the performance of image analysis algorithms.
- ◊ **Exploration of the GFlowNets project**: Although this project was paused due to data limitations, the study provided a deep understanding of multimodal learning techniques and generated methodologies applicable to other contexts.
- ◊ **Optimization of the knowledge graph and chatbot**: The enrichment of the knowledge graph (+300,000 connections) and the implementation of new features for the chatbot have made this tool more accessible and operational, with concrete applications for Ksilink's biologists.

These results, although sometimes partial, have contributed to strengthening the company's technological foundations and identifying avenues for improvement for the developed tools.

B. Observations

Positive points:

- ◊ The use of advanced tools like PyTorch Lightning, Hydra, and Wandb enabled the adoption of a professional methodology.
- ◊ Interactions with biologists and end-users fostered a better understanding of real needs, effectively guiding tool development.
- ◊ The modularity of the proposed solutions ensures their reusability and adaptability for future projects.
- ◊

Limitations and areas for improvement:

- ◊ The quality and quantity of data were a major hindrance, especially for the GFlowNets project. Access to richer, better-structured datasets is essential for significant advances.
- ◊ Certain chatbot features, such as handling some complex questions, still require adjustments to achieve optimal reliability.

C. Estimation of Financial Gain Brought by My Activity

Although the work carried out during this internship did not generate direct financial gains, it helped lay the groundwork for projects that could have a significant impact in the medium and long term:

- ◊ **Improvement of internal efficiency**: The enrichments to the knowledge graph and the chatbot optimizations save substantial time for the biologist teams by simplifying access to information, allowing them to test certain hypotheses quickly. This addition could save up to hours in the search for links between compounds and the understanding of interactions.
- ◊ **Reduction of research costs**: The methodologies and tools developed (multimodal learning models) lay the foundation for reducing the need for costly experimental trials in the laboratory by promoting better pre-selection of compounds to be tested.
- ◊ **Innovation perspectives**: The results and methodologies explored in this internship open up opportunities for future developments in the field of artificial intelligence applied to biotechnology while keeping the data private and within the company.

V. Bibliography

1. Ksilink - Patient-based Drug Discovery : About us [Internet]. Ksilink. [cited 2025 Jan 6]. Available from: https://www.ksilink.com/our-story/

2. Boyé M. Myrtil Biotechnologies – innovating drug discovery for patients with dilated cardiomyopathy [Internet]. 2022 [cited 2025 Jan 6]. Available from: https://www.pressebox.com/pressrelease/ksilink/Myrtil-Biotechnmologies-innovating-drug-discovery-for-patients-with-dilated-cardiomyopathy/boxid/1119575

3. Lu SZ, Lu Z, Hajiramezanali E, Biancalani T, Bengio Y, Scalia G, Koziarski M. Cell Morphology-Guided Small Molecule Generation with GFlowNets [Internet]. arXiv; 2024 [cited 2025 Jan 6]. Available from: http://arxiv.org/abs/2408.05196

4. Mueller JP. L'Intelligence artificielle pour les Nuls - 7Switch [Internet]. Pour les nuls. Vol. 1. 2019 [cited 2025 Jan 9]. 364 p. Available from: https://www.7switch.com/fr/ebook/9782412049037/l-intelligence-artificielle-pour-les-nuls

5. Bolouri H. Computational Modeling of Gene Regulatory Networks — A Primer [Internet]. Vol. 1. 2008 [cited 2025 Jan 9]. 340 p. Available from: https://www.worldscientific.com/worldscibooks/10.1142/p567#t=aboutBook

6. Khan A. A Beginner's Guide to Deep Learning with MNIST Dataset [Internet]. Medium. 2024 [cited 2025 Jan 9]. Available from: https://medium.com/@azimkhan8018/a-beginners-guide-to-deep-learning-with-mnist-dataset-0894f7183344

7. Training a neural network on MNIST with Keras | TensorFlow Datasets [Internet]. TensorFlow. [cited 2025 Jan 9]. Available from: https://www.tensorflow.org/datasets/keras_example

8. Training a Classifier — PyTorch Tutorials 2.5.0+cu124 documentation [Internet]. [cited 2025 Jan 13]. Available from: https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html

9. Kalantar R. How to visualize layer activations in PyTorch [Internet]. Medium. 2024 [cited 2025 Jan 6]. Available from: https://medium.com/@rekalantar/how-to-visualize-layer-activations-in-pytorch-d0be1076ecc3

10. Mazur. A Step by Step Backpropagation Example [Internet]. Matt Mazur. 2015 [cited 2025 Jan 13]. Available from: https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

11. How to use TensorBoard with PyTorch — PyTorch Tutorials 2.5.0+cu124 documentation [Internet]. [cited 2025 Jan 6]. Available from: https://pytorch.org/tutorials/recipes/recipes/tensorboard_with_pytorch.html

12. Welcome to ⚡ PyTorch Lightning — PyTorch Lightning 2.5.0.post0 documentation [Internet]. [cited 2025 Jan 6]. Available from: https://lightning.ai/docs/pytorch/stable/

13. ALI AWAN A. A Complete Guide to Data Augmentation [Internet]. [cited 2025 Jan 6]. Available from: https://www.datacamp.com/tutorial/complete-guide-data-augmentation

14. Kundu R. The Beginner's Guide to Contrastive Learning [Internet]. [cited 2025 Jan 6]. Available from: https://www.v7labs.com/blog/contrastive-learning-guide

15. Rubilmax. Introduction au contrastive learning : une forme d'apprentissage auto supervisé [Internet]. Medium. 2022 [cited 2025 Jan 20]. Available from: https://medium.com/@rubilmax/introduction-au-contrastive-learning-une-forme-dapprentissage-auto-supervis%C3%A9-95c3bc070d1c

16. Radford A, Kim JW, Hallacy C, Ramesh A, Goh G, Agarwal S, Sastry G, Askell A, Mishkin P, Clark J, Krueger G, Sutskever I. Learning Transferable Visual Models From Natural Language Supervision [Internet]. arXiv; 2021 [cited 2025 Jan 9]. Available from: http://arxiv.org/abs/2103.00020

17. Hydra | Weights & Biases Documentation [Internet]. [cited 2025 Jan 6]. Available from: https://docs.wandb.ai/guides/integrations/hydra/

18. W&B Docs | Weights & Biases Documentation [Internet]. [cited 2025 Jan 6]. Available from: https://docs.wandb.ai/

19. Lu S. TheMatrixMaster/mmc-gene [Internet]. 2024 [cited 2025 Jan 8]. Available from: https://github.com/TheMatrixMaster/mmc-gene

20. Poklukar P, Vasco M, Yin H, Melo FS, Paiva A, Kragic D. Geometric Multimodal Contrastive Representation Learning [Internet]. arXiv; 2022 [cited 2025 Jan 8]. Available from: http://arxiv.org/abs/2202.03390

21. Moshkov N, Becker T, Yang K, Horvath P, Dancik V, K Wagner B, A Clemont P, Singh S, Carpenter A, C Caicedo J. Predicting compound activity from phenotypic profiles and chemical structures | bioRxiv [Internet]. [cited 2025 Jan 6]. Available from: https://www.biorxiv.org/content/10.1101/2020.12.15.422887v4.full

22. Moshkov N, Becker T, Yang K, Horvath P, Dancik V, Wagner BK, Clemons PA, Singh S, Carpenter AE, Caicedo JC. Chemical structures, Cell Painting and transcriptional profiles for compound bioactivity prediction. [Internet]. Zenodo; 2023 [cited 2025 Jan 13]. Available from: https://zenodo.org/records/7729583

23. JUMP-Cell Painting Consortium [Internet]. JUMP-Cell Painting Consortium. [cited 2025 Jan 10]. Available from: https://jump-cellpainting.broadinstitute.org/

24. Fradkin P, Azadi P, Suri K, Wenkel F, Bashashati A, Sypetkowski M, Beaini D. How Molecules Impact Cells: Unlocking Contrastive PhenoMolecular Retrieval [Internet]. arXiv; 2024 [cited 2025 Feb 3]. Available from: http://arxiv.org/abs/2409.08302

25. Chandak P, Huang K, Zitnik M. Building a knowledge graph to enable precision medicine. Sci Data. 2023 Feb 2;10(1):67.

26. Streamlit Docs [Internet]. [cited 2025 Jan 6]. Available from: https://docs.streamlit.io/

27. What is CRISPR: Your Ultimate Guide [Internet]. Synthego. [cited 2025 Jan 9]. Available from: https://www.synthego.com/learn/crispr

28. Chall S. Understanding Molecular Similarity [Internet]. Medium. 2023 [cited 2025 Jan 9]. Available from: https://medium.com/@santuchal/understanding-molecular-similarity-51e8ebb38886

29. rdkit/rdkit [Internet]. RDKit; 2025 [cited 2025 Jan 20]. Available from: https://github.com/rdkit/rdkit

30. Neo4j documentation - Neo4j Documentation [Internet]. Neo4j Graph Data Platform. [cited 2025 Jan 6]. Available from: https://neo4j.com/docs/docs/

31. Using a Knowledge Graph to Implement a RAG Application [Internet]. [cited 2025 Jan 6]. Available from: https://www.datacamp.com/tutorial/knowledge-graph-rag

32. Sepasdar Z, Gautam S, Midoglu C, Riegler MA, Halvorsen P. Enhancing Structured-Data Retrieval with GraphRAG: Soccer Data Case Study [Internet]. arXiv; 2024 [cited 2025 Jan 6]. Available from: http://arxiv.org/abs/2409.17580

33. Nova M. RETRIEVAL AUGMENTED GENERATION (RAG) WITH LANGCHAIN. 2024 Dec 18;61.

34. GenAI Ecosystem - Developer Guides [Internet]. Neo4j Graph Data Platform. [cited 2025 Jan 9]. Available from: https://neo4j.com/developer/genai-ecosystem/

35. sentence-transformers/all-MiniLM-L6-v2 · Hugging Face [Internet]. 2024 [cited 2025 Jan 6]. Available from: https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2

36. How does `GraphCypherQAChain.from_llm` construct answers based on a cypher query's results? · langchain-ai/langchain · Discussion #15313 [Internet]. GitHub. [cited 2025 Jan 9]. Available from: https://github.com/langchain-ai/langchain/discussions/15313

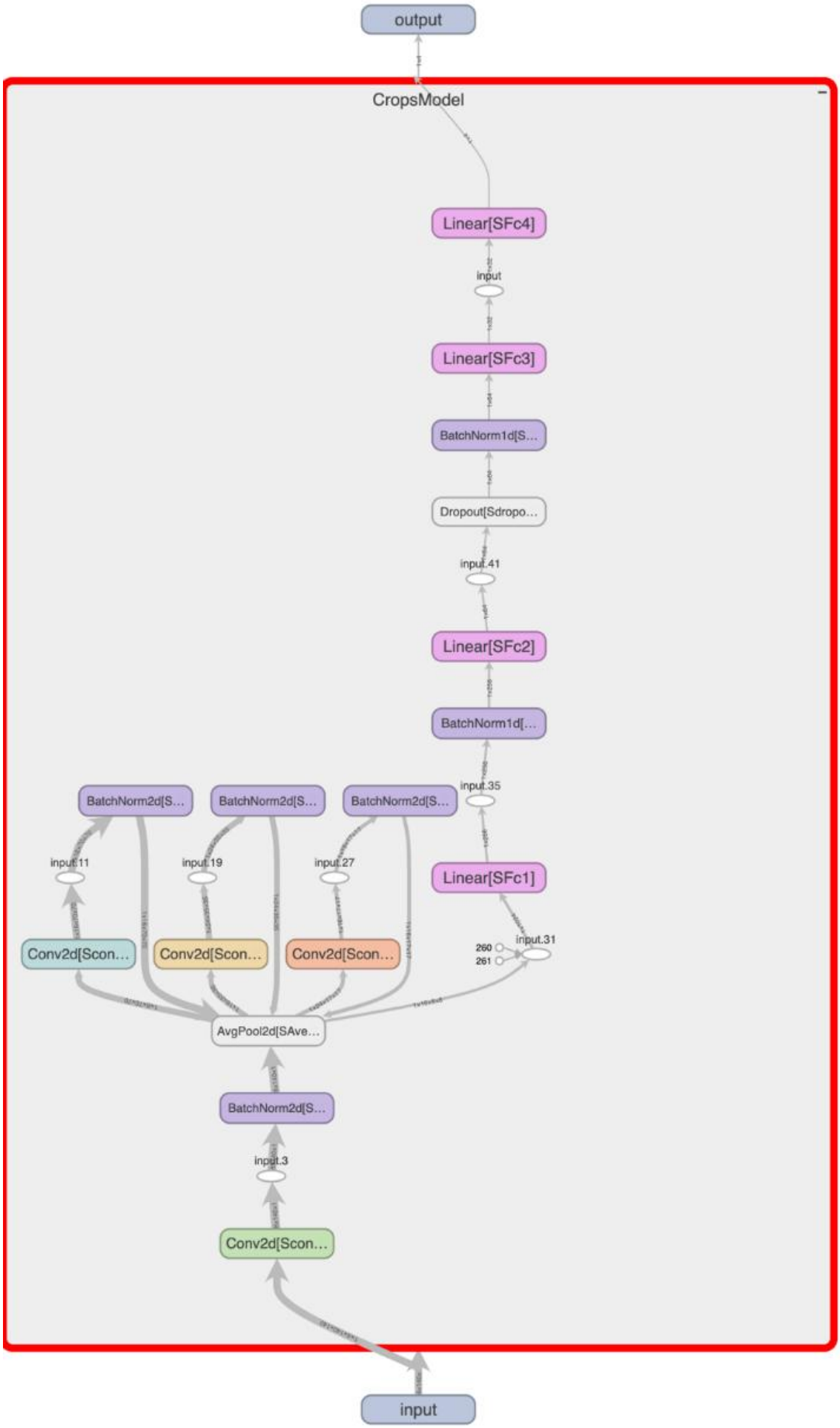
37. Klose C. ChrisDelClea/streamlit-agraph [Internet]. 2025 [cited 2025 Jan 10]. Available from: https://github.com/ChrisDelClea/streamlit-agraph

VI. Table of figures

VII. Appendices

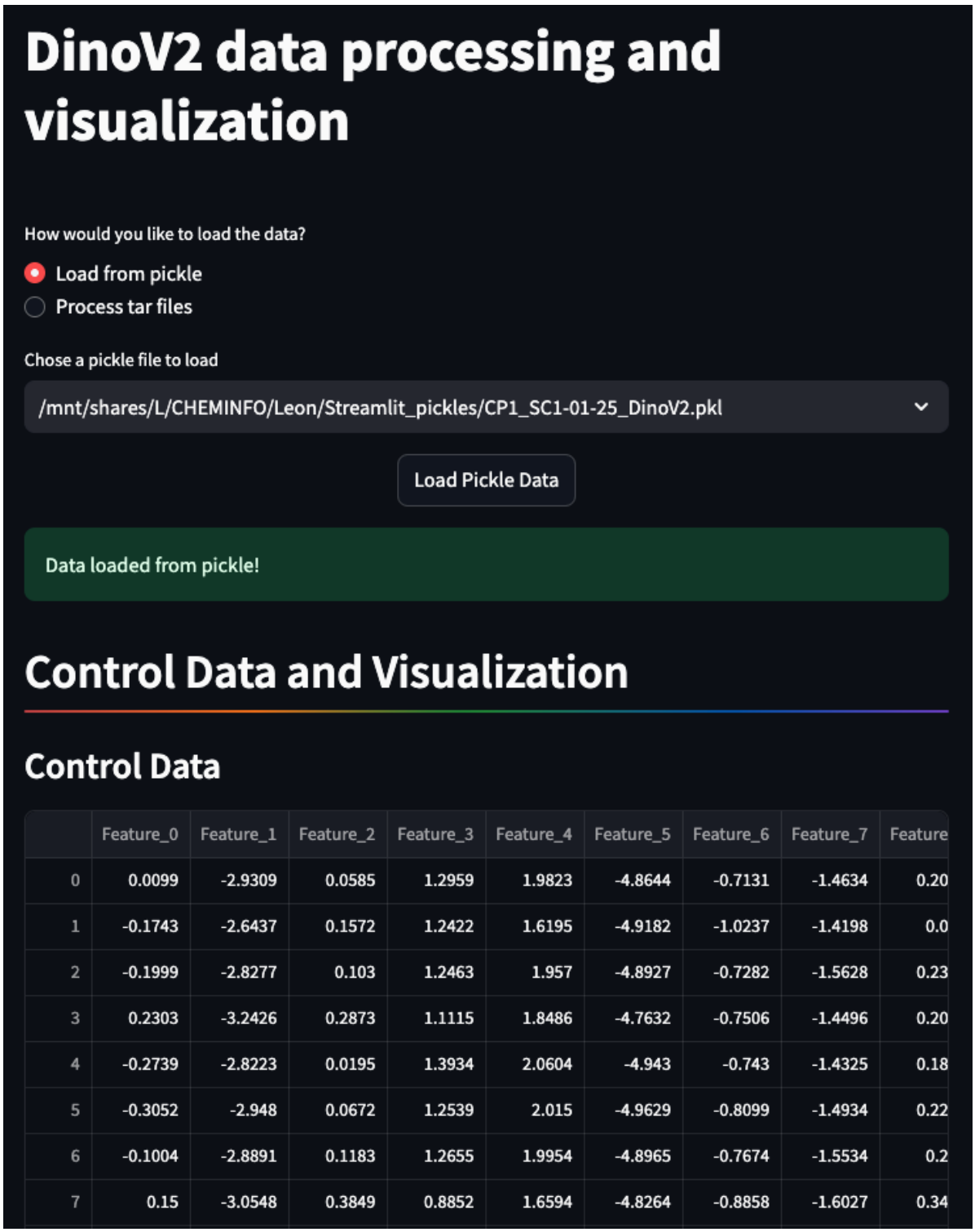
i. Appendix 1 :

Diagram of my model via Tensorboard, including several convolution layers, batch normalization, average pooling, dropout, ... The model takes an image with 4 channels as input and outputs a prediction among the four classes.


output
CropsModel
Linear[SFc4]
input
Linear[SFc3]
BatchNorm1d[S...
Dropout[Sdropo...
input.41
Linear[SFc2]
BatchNorm1d[...
input.35
BatchNorm2d[S...
BatchNorm2d[S...
BatchNorm2d[S...
input.11
input.19
input.27
Linear[SFc1]
Conv2d[Scon...
Conv2d[Scon...
Conv2d[Scon...
260
261
input.31
AvgPool2d[SAve...
BatchNorm2d[S...
input.3
Conv2d[Scon...
input

ii. Appendix 2 :

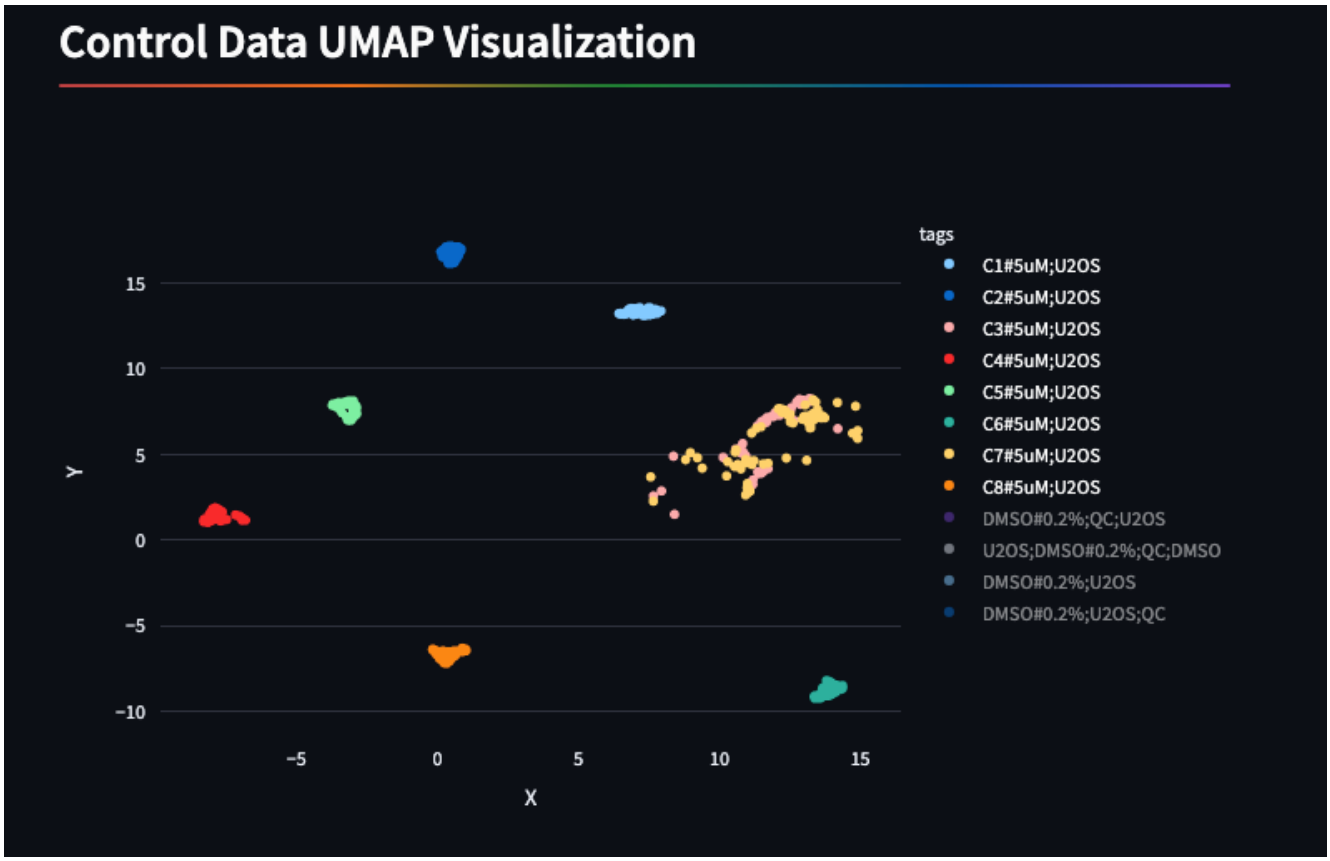
Training for handling .tar files. I had to create a Streamlit page capable of either loading data from a .tar file and then loading it into a pickle file for quick reuse, or directly loading an already created pickle file.



	Feature_0	Feature_1	Feature_2	Feature_3	Feature_4	Feature_5	Feature_6	Feature_7	Feature
0	0.0099	-2.9309	0.0585	1.2959	1.9823	-4.8644	-0.7131	-1.4634	0.20
1	-0.1743	-2.6437	0.1572	1.2422	1.6195	-4.9182	-1.0237	-1.4198	0.0
2	-0.1999	-2.8277	0.103	1.2463	1.957	-4.8927	-0.7282	-1.5628	0.23
3	0.2303	-3.2426	0.2873	1.1115	1.8486	-4.7632	-0.7506	-1.4496	0.20
4	-0.2739	-2.8223	0.0195	1.3934	2.0604	-4.943	-0.743	-1.4325	0.18
5	-0.3052	-2.948	0.0672	1.2539	2.015	-4.9629	-0.8099	-1.4934	0.22
6	-0.1004	-2.8891	0.1183	1.2655	1.9954	-4.8965	-0.7674	-1.5534	0.2
7	0.15	-3.0548	0.3849	0.8852	1.6594	-4.8264	-0.8858	-1.6027	0.34

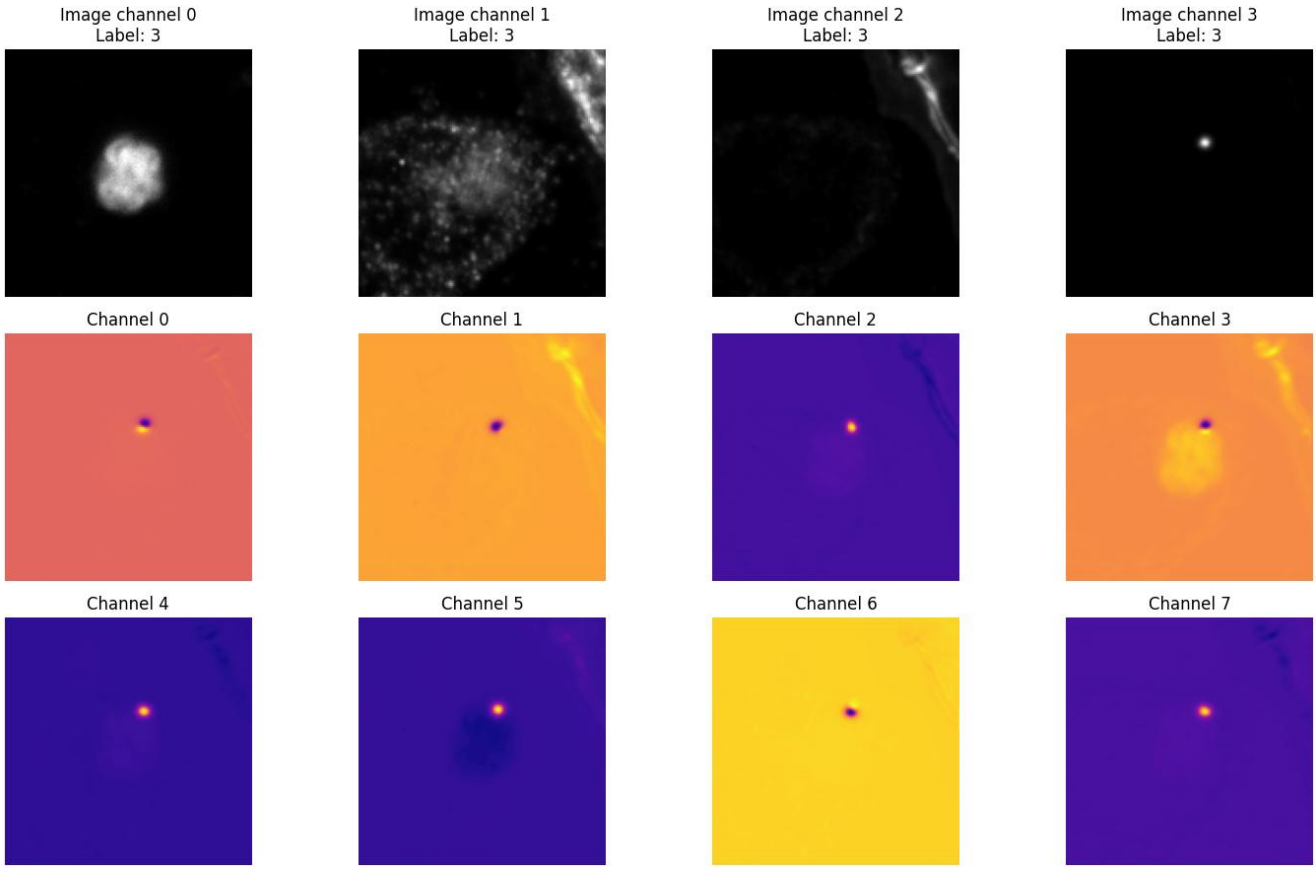
iii. Appendix 3 :

Once the data is loaded, creation and interactive display of a UMAP allowing to quickly see if the different data categories are well separated.



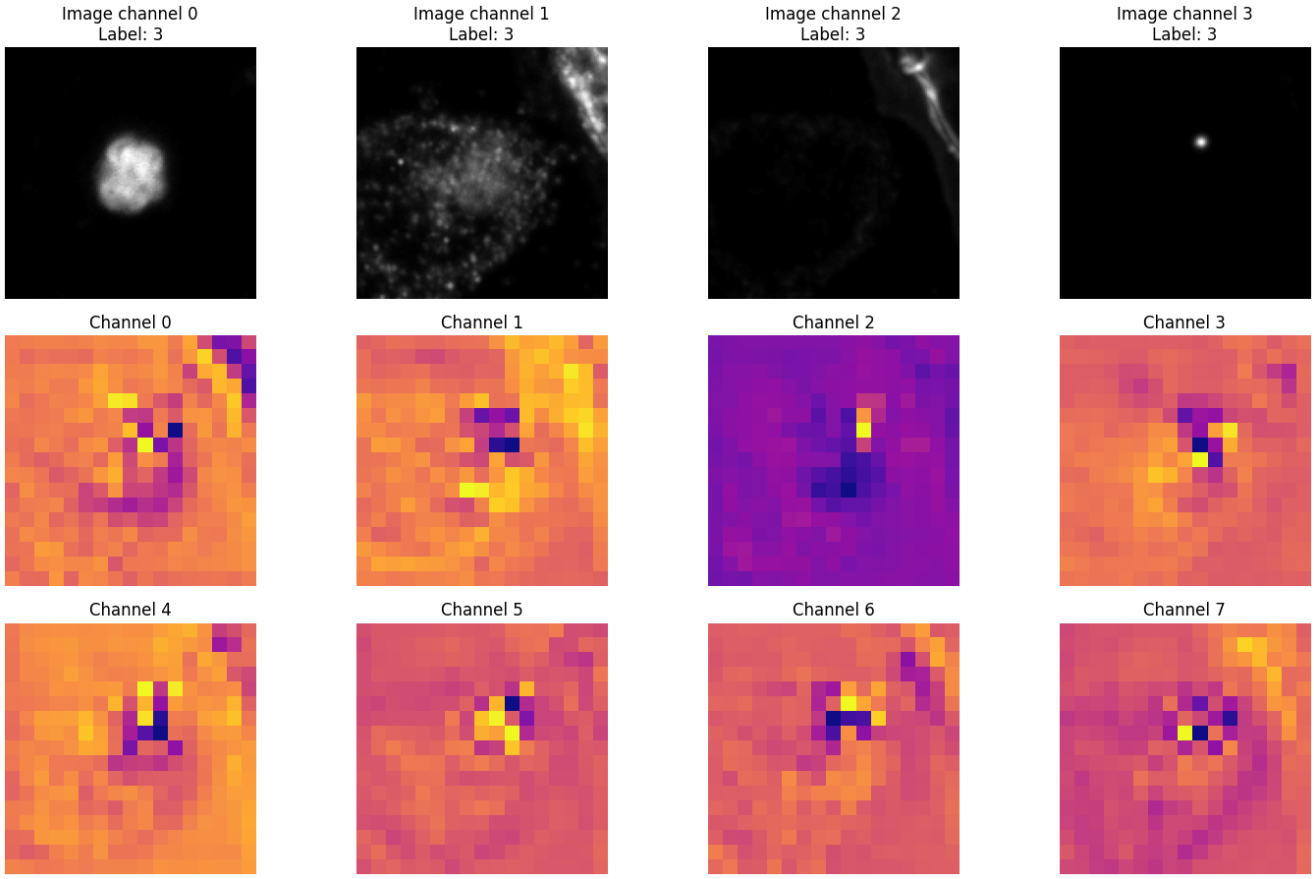
iv. Appendix 4 :

Images of the eight channels of the first convolutional layer as well as the four original channels. In this color gradient, the lighter it is, the more intense, and the darker, the less intense. The first convolutional layer takes 4 channels as input and outputs eight.



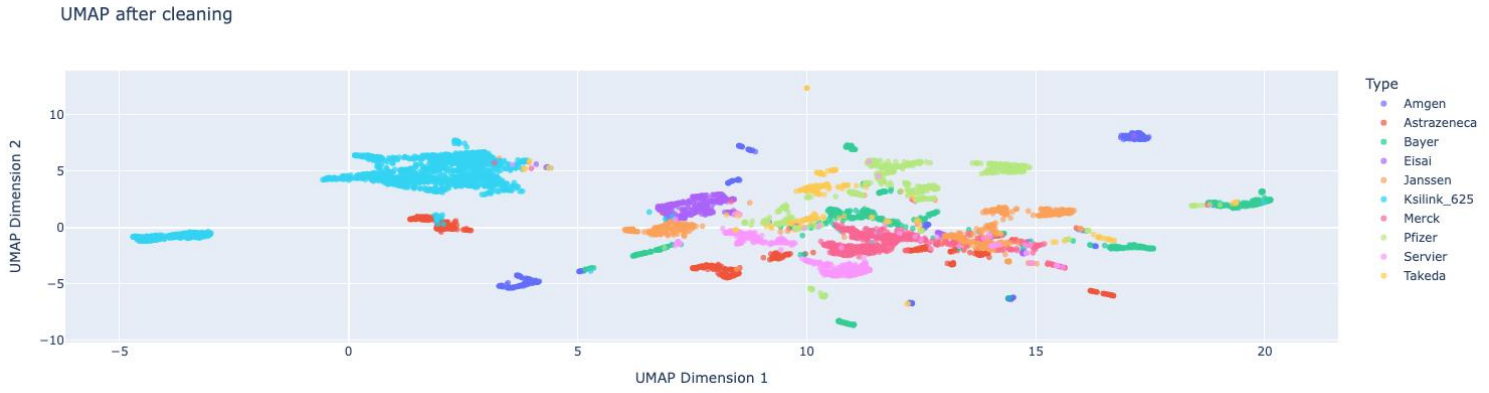
v. Appendix 5 :

Image of the first eight channels of the fourth convolutional layer as well as the four original channels. In this color gradient, the lighter it is, the more intense, and the darker, the less intense. The fourth convolutional layer takes 24 channels as input and outputs 16, of which here are the first 8.



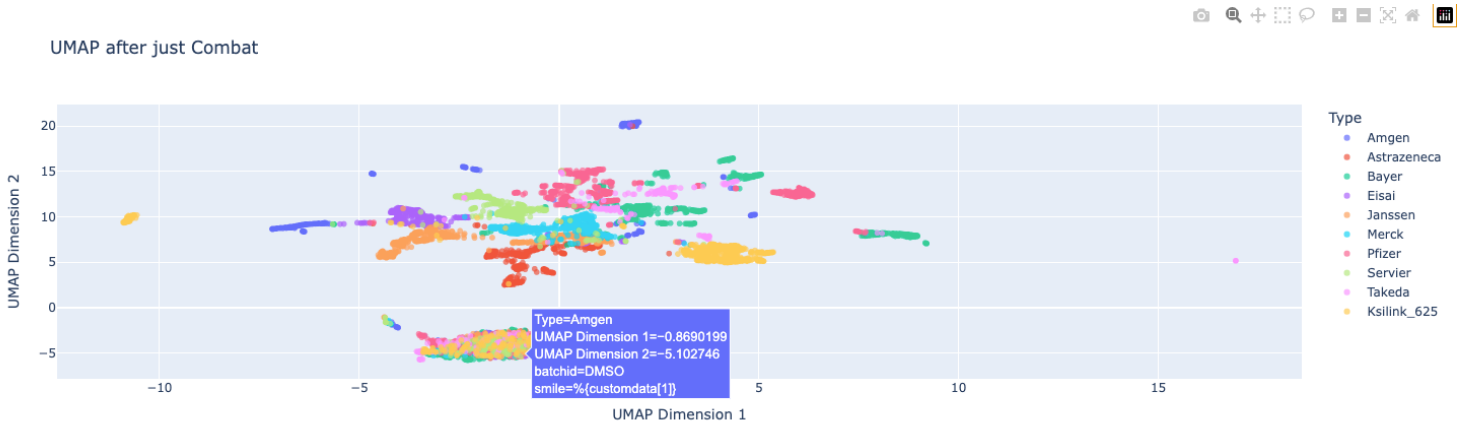
vi. Appendix 6 :

Graph representing data from different sources. Each source mostly has different compounds except for a solvent common to all sources, DMSO. The goal is for the source (microscope type, date, temperature, ...) not to influence the data. The data must therefore be recalibrated to ensure all DMSOs are grouped together, while still having distinct compound groups.
Here we can see that the compounds are grouped by source and we do not see the DMSOs grouped; this must be corrected..



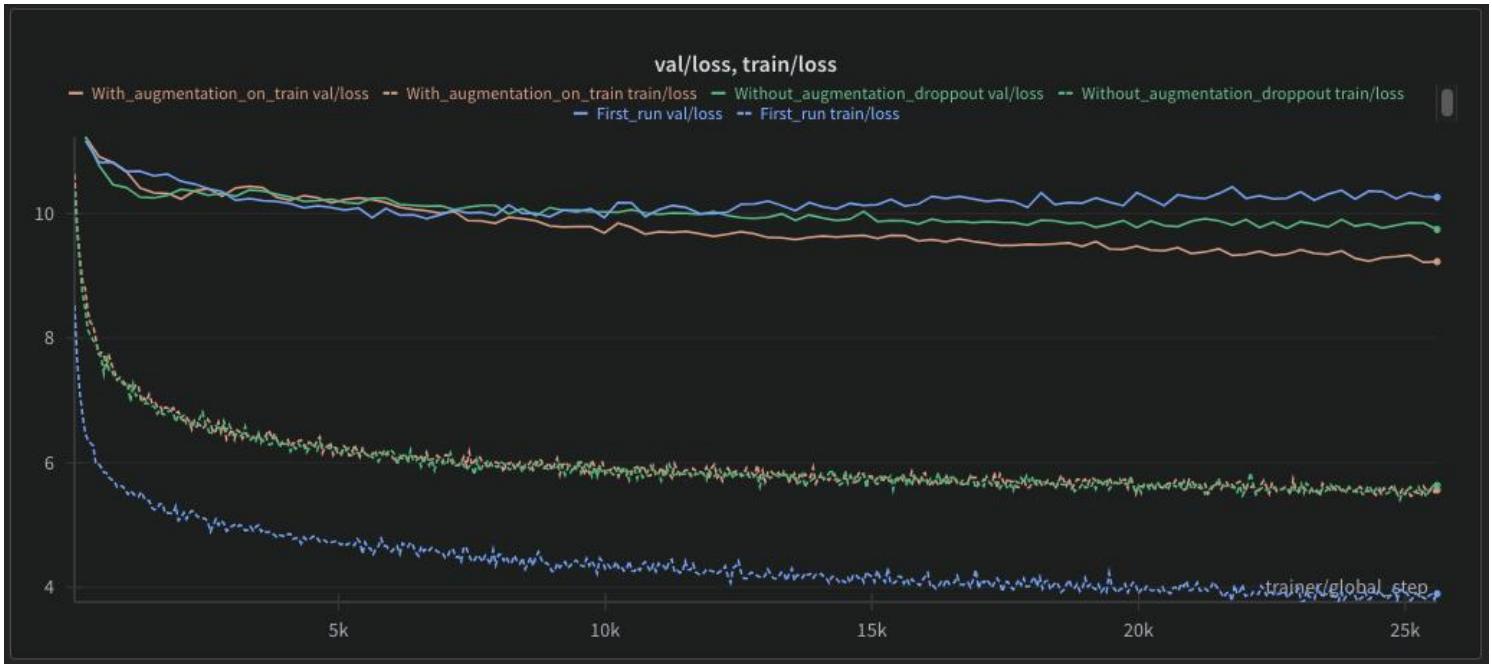
vii. Appendix 7 :

Here, we can see that the compounds are slightly grouped by source, but all the DMSOs are grouped in the same place. The correction has therefore been properly performed.



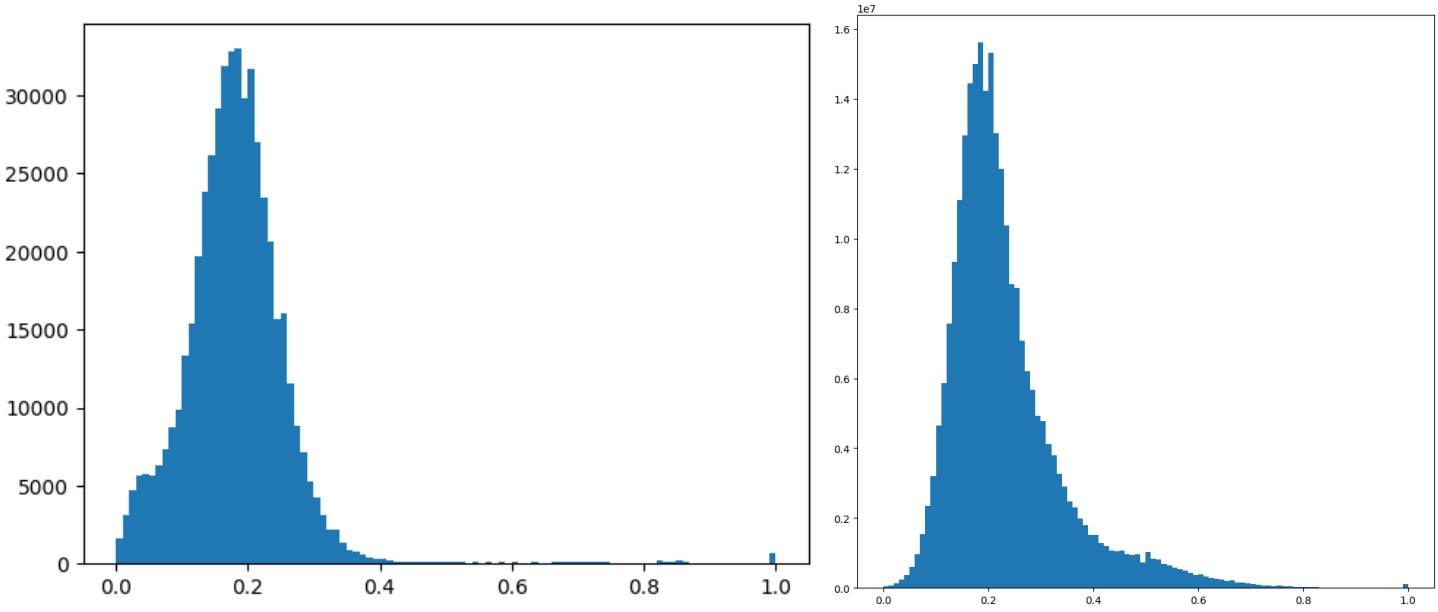
viii. Appendix 8 :

Results obtained with several dataset attempts. The solid line is the loss curve for the validation set, and the dotted line is for the training set. We can see a significant difference between training and validation, which clearly shows that the model is overfitting the training data and failing to generalize to the validation data.



ix. Appendix 9 :

On these two images, we can see a comparison of Tanimoto similarity for Ksilink's data (left) and the publication's data (right). Here, Ksilink's data was selected to be as similar as possible. Yet, despite this, Ksilink's data is not as similar as the publication's data, even though Ksilink's data contains only 700 compounds compared to 16,170 for the publication's data.



MORALES Léon

Internship Report - P2024

Development of a Large Language Model for the pharmaceutical industry

The subject of my internship focused on the development of a Large Language Model (LLM) applied to the pharmaceutical industry, by leveraging Ksilink's internal data. My internship was divided into two main projects:

Construction of a common latent space: This step aimed to integrate several types of data, including molecular descriptions (in SMILES format) and information extracted from cell images. The objective was to represent this data in a shared vector space, thus facilitating their joint analysis and exploitation with the goal of creating a chemical molecule generator.

Knowledge graph and link with the LLM: This part aimed to enrich the knowledge graph by adding new relations and information. This also included improving the interface connecting the knowledge graph to an LLM, to allow natural language interactions and the addition of features making the tool more instinctive and understandable for biologists.

KSILINK
16 rue d'Ankara
67 000 Strasbourg
www.ksilink.com