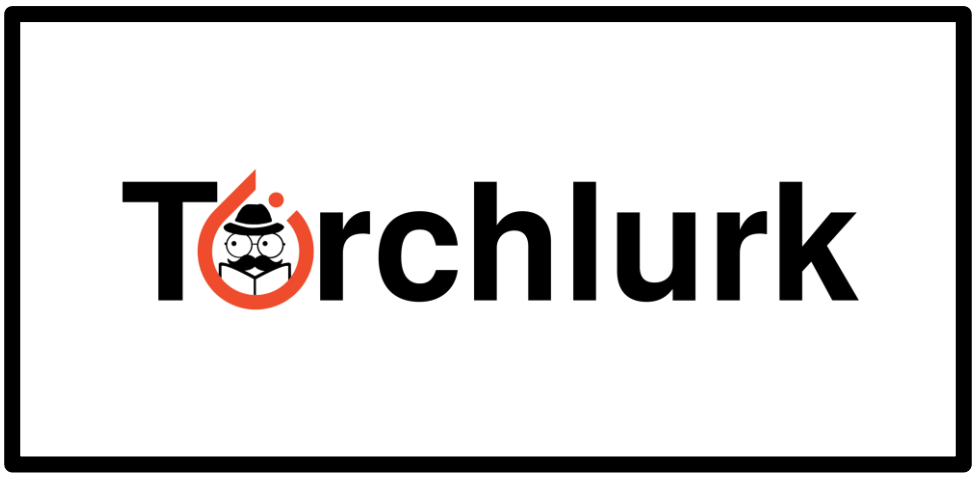


Data Visualization - EPFL **May 2020**

Yann **Mentha**

Julien **Berger**

Gianni **Giusto**

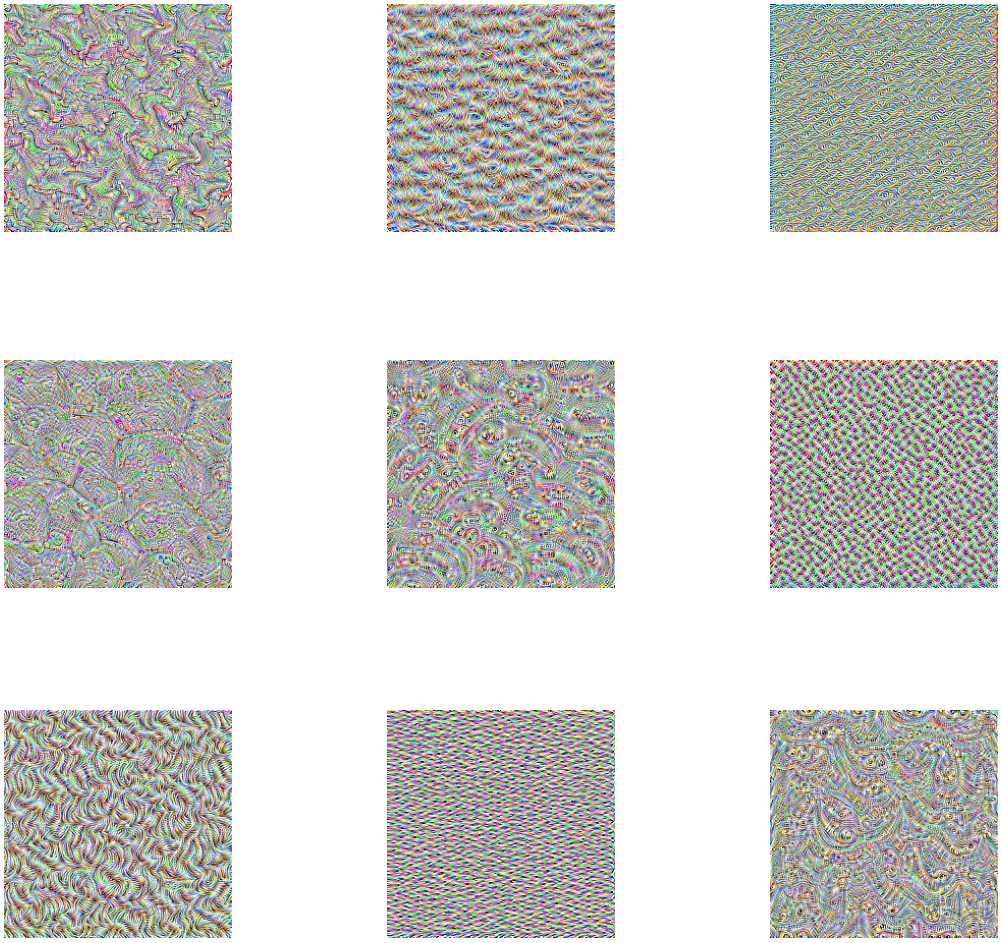
PROCESS BOOK

Data visualization (com-480)

Introduction

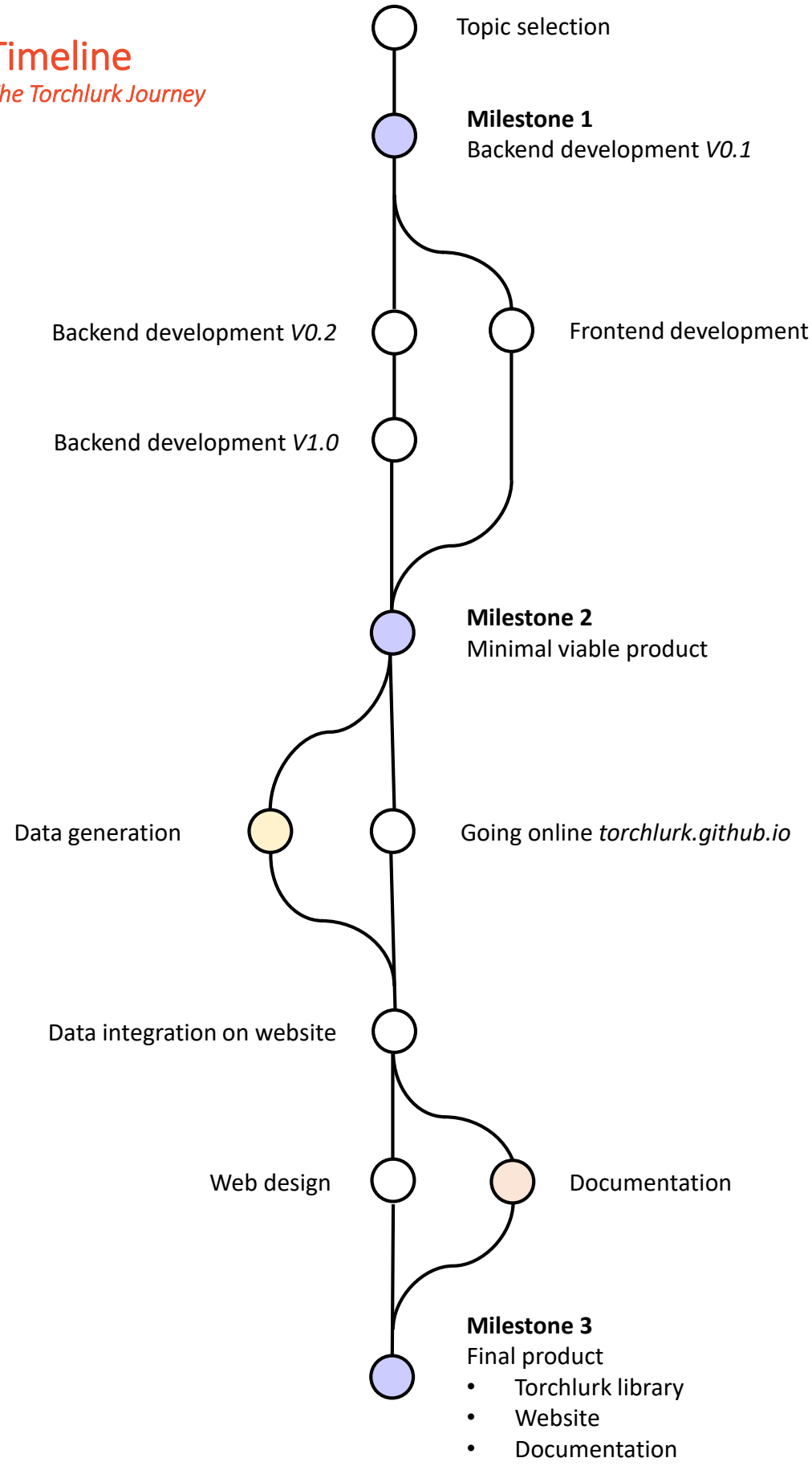
In the past recent years, Convolutional Neural Networks (CNN) have shown impressive capabilities in image recognition tasks even outperforming humans. Despite undeniable success in all field of science, one can wonder why do these processing paradigms perform so well. Indeed, complexity increases along with the number of layers and internal computation lack of transparency for the user.

Torchlurk is a visualization tool that aims to ease the interpretation of trained CNN and give insights to the user about what the network is actually learning. The name *Torch-* refers to the use of the PyTorch framework and *-lurk* to our willingness to unveil the intrinsic abstraction levels that are *hidden* in these CNNs.



Timeline

The Torchlurk Journey



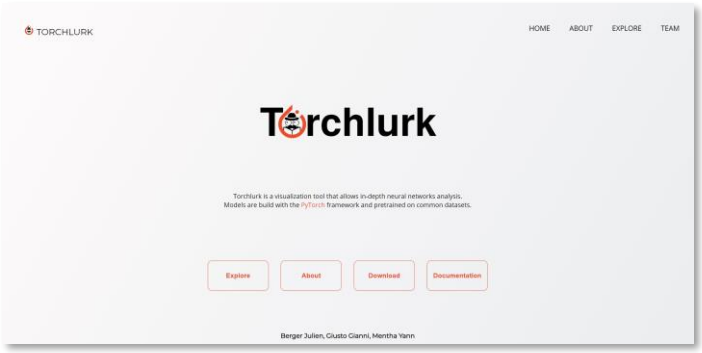
The Torchlurk Journey

From the topic selection to the final product, the graphs and the visual identity of the website have been constantly rethought in order to build an intuitive tool. We show below some initial visualizations together with their final version to attest to these changes and to illustrate the overall evolution of the project. We also display new visualizations and present the visual identity of Torchlurk.

Visual identity

We first developed a logo to give our Torchlurk tool a visual identity. It contains in particular the PyTorch logo with a man, the *Lurker*, viciously inspecting a newspaper which refers to our willingness to carefully inspect what's inside a network. The logo is incorporated as the *O* of a simple and modern font.

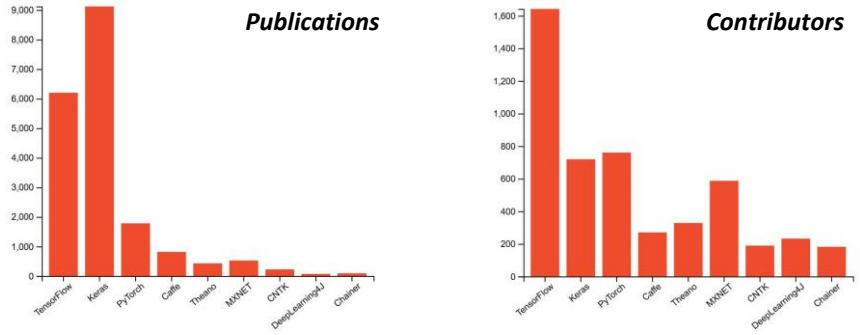
The home page has been designed to be simple and refined. Only the necessary information is shown. From this main page, we can access all the features of Torchlurk: we can either *explore* the networks, which is the main purpose of the tool or inquire about its use in the *about* section. The *download* button takes us directly to the GitHub repository whilst the *documentation* informs us on the functions implemented in Torchlurk.



The primary advantage of such a refined design for a tool like the one we developed is that it does not drown the user in a flood of unnecessary information. Instead, it offers a general overview of Torchlurk functionalities.

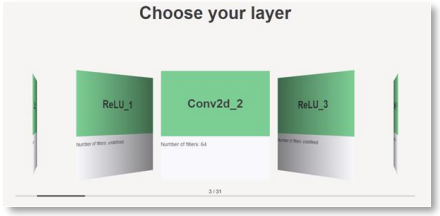
The Torchlurk Journey

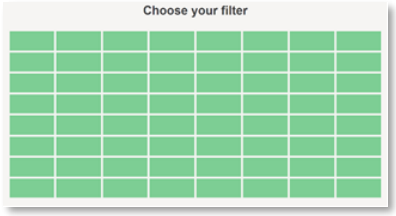
Network popularity



This visualization made with D3.js allows to dynamically select between 2 features which are indicators of deep learning framework popularity. On one side, we compare scientific publications in 2018 and on the other we compare contributors. TensorFlow remains the most popular framework overall. However, we believe that PyTorch is about to become more and more popular since it has already been closing the gap bewteen its concurrents since several years now.

Swiper





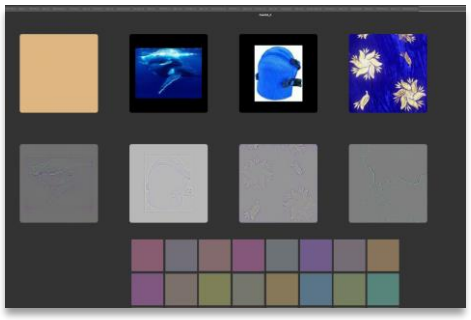
The initial swiper (*top left*) was a mean for the user to navigate through layers and select associated filters (*top right*). Nevertheless, this method does not provide an overview of the network structure. To address the problem, we decided to create an SVG from scratch, with each box corresponding to a network layer (*bottom*). The user can only select convolutional layers which are the ones we are interested in for the visualization.

The Torchlurk Journey

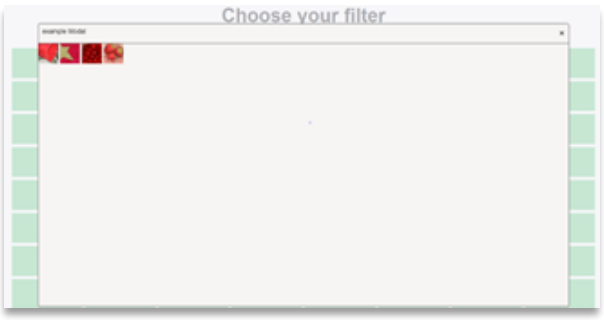
Popup window

Propably the tool that has evolved the most and which has taken us the longest to develop. Despite its mere appearance at first sight, it contains all the information specific to a given convolutional layer. Here again, it is important not to overload the popup with too much information.

Version 1

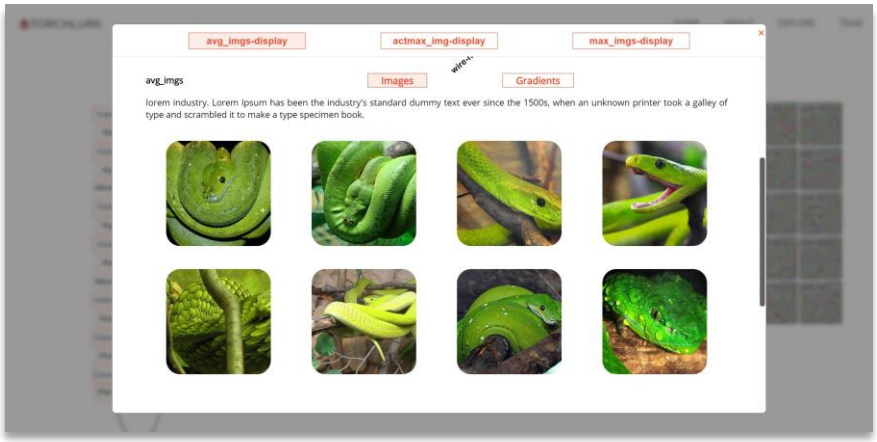


Version 2



The first version only displayed the three images that excited the most a filter and their associated gradients. This arrangement of space forced the user to scroll down to select a filter of interest and scroll back to the top to view corresponding images. This slowed down the experienced quite a bit. This led us to rethink the design and we opted for the *popup* window. The latest has the advantage of being easily accessible. The second version remained nevertheless rudimentary.

Final version

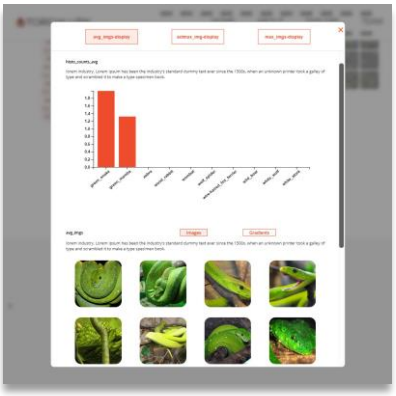


The newly designed popup windows offers a wide variety of options. We split them into different pages: *average activation*, *filter visualization*, *max activation*, each of them accessible with the top buttons. To discover all these features in more details, we invite you to have a try on the website. More detailed explanations can be found on the following page.

The Torchlurk Journey

Popup window

We provide below details about the information contained in the popup window.



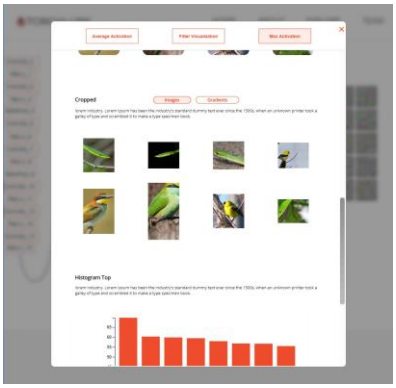
Average activations – Images and gradients

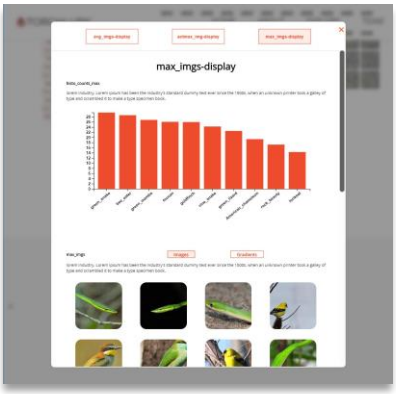
The set of 8 images are the ones from the train dataset which excite the filter the most in *average*.

In order to save space, we included two buttons that allow the user to chose between visualizing images or their corresponding gradient.

Activations – Cropped images

We also offer the possibility to visualize cropped versions of the images, *i.e.* the location in the original image to which the filter responds maximally. The histograms below represent the maximum spike corresponding to each image.





Max activations

Similarly to *average activation*, the set of 8 images are the ones from ImageNet train dataset which excite maximally the filter.

Again, with just one click on the buttons we can switch from the images to their corresponding gradients.

Challenges

Throughout the project, we kept the primary use of our product in mind: Torchlurk is a visualization tool that aims to explore convolutional neural networks. This implies that all functionalities of Torchlurk must be easily accessible and understandable by any user.

To analyze the network internal structure, lots of information can be displayed, such as images that most excite a filter, histograms class distributions etc. The key challenge was hence to avoid overloading the space with too much information. To address the problem we combined flexible and modular data visualization techniques such as *swiper* or *popup* window (*c.f.* Visualization section fot details).

From a techincal aspects, as we did not have any read-to-use data, we first had to generate the results to be displayed. This step took longer than expected as we needed a lot of computational power. To cope with this issue, we ran simulations with Google Colab even if the compuational time remains consequent. Unfortunately, we were able to generate results for only one network (VGG), but the visualization would be exactly the same for other ones.

In addition, as we deal with large files, we did not want it to slow down the navigation experience on our website once served on GitHub pages. A small timelapse is required to load the main JSON file (< 5 seconds) but once loaded, the surfing is not impeed at all.

Torchlurk in a nutshell

Torchlurk is a library that can easily be installed via pip. Simply enter the command:

```
$pip install torchlurk
```

to benefit from all the features of the tool. It comes with a full documentation listing all its attributes and functions.

What's next?

Torchlurk library will be maintained even after the project. As we were limited with the computational power, we wish to compute similar results for AlexNet and resNet. We also don't want to restrict ourselves to explore deep feedforward networks but we also want to incorporate the possibility to visualize Recurrent Neural Networks. Finally, we aim to implement new regularization methods to magnify filter visualization in the same way OpenAI Microscope did with the *Lucid* library.

Peer assessement

The communication between the 3 of us was fluent and the exchanges were constructive. All of us contributed to the final product with the objective to provide the users with a practical and intuitive interface to explore neural networks. The segregation of duties at the beginning allowed us to parallelize the job to be done and hence to work more efficiently: while one person was implementing the *backend*, the others could develop the web interface on a separated branch. Once we produced all the data required to be displayed, we were able to join forces to design and custom the website, while documenting the code. In the end, we feel like we have been carrying a project from conception to delivery and developed a practical tool that can be used by the PyTorch community.

Yann Mentha

Principal developer of the backend along with the documentation. He provided the team with all the necessary data required for the final product. He also developed the visual identity of the website and took part in every group discussion to share his opinions about the final product.

Works on the visual identity of the website. He implemented visualization tools and interactions such as the popup window with Javascript and D3.js. He took part in every group discussion to share his opinions about the final product.

Julien Berger

Gianni Giusto

Works on the website visual identity and development. He implemented data visualization graphs as well as interactions with Javascript and D3.js. He took part in every group discussion to share his opinions about the final product.