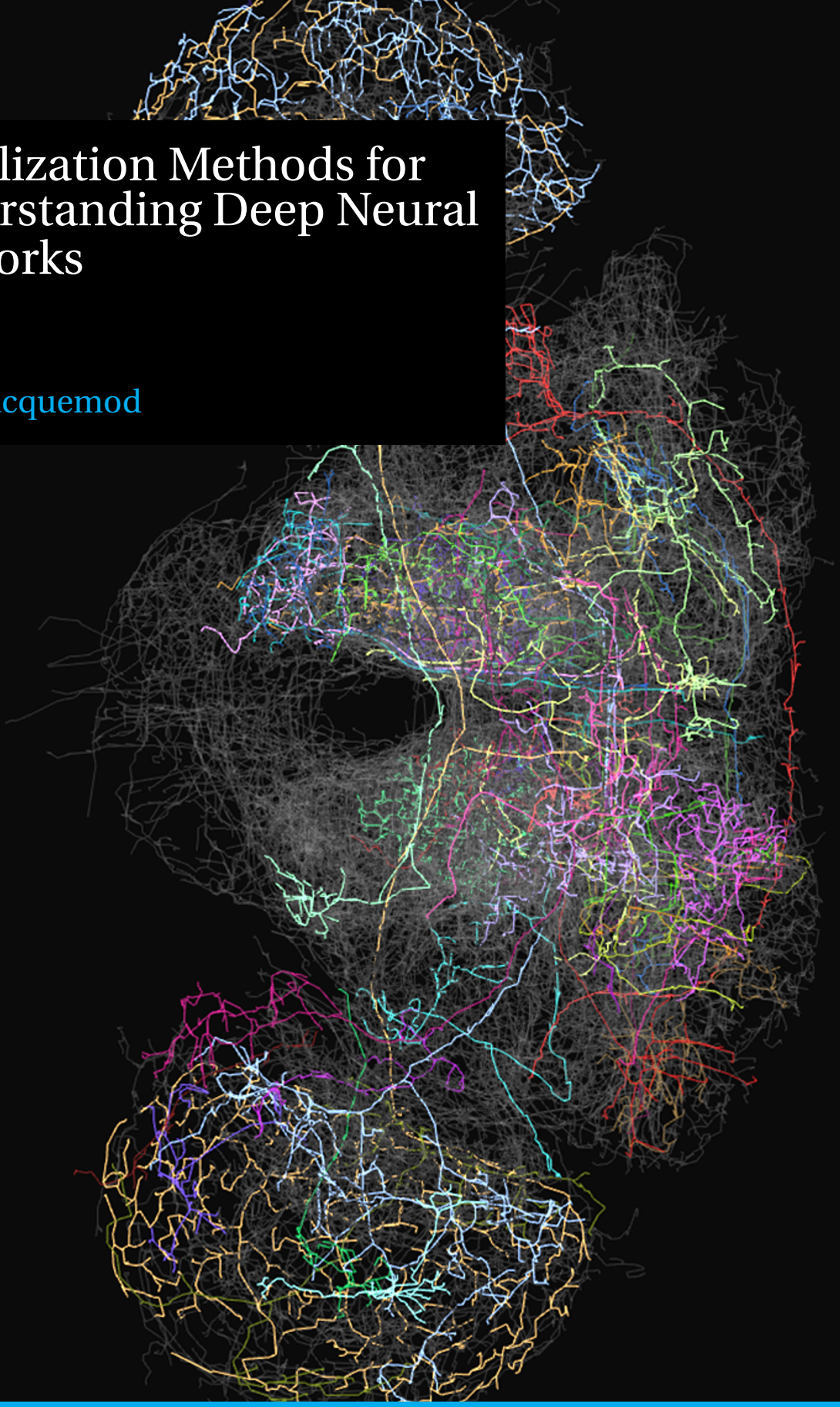
# Visualization Methods for Understanding Deep Neural Networks

Laura Jacquemod

Technische Universiteit Delft



TU Delft Delft University of Technology

PHILIPS

Challenge the future

# Visualization Methods for Understanding Deep Neural Networks

by

**Laura Jacquemod**

in partial fulfillment of the requirements for the degree of

**Master of Science**
in Computer Science, Digital Media Technology track

at the Delft University of Technology,
to be defended publicly on Tuesday August 29, 2017 at 1:30 PM.

| | | |
|---|---|---|
| Student number: | 4600231 | |
| Supervisor: | Dr. A. Vilanova | |
| Thesis committee: | Prof. Dr. E. Eisemann, | TU Delft |
| | Prof. D. Tax, | TU Delft |
| | Dr. B. Gebre, | Philips Research |



The work in this thesis was carried out at Philips Research in Eindhoven, Netherlands.

An electronic version of this thesis is available at http://repository.tudelft.nl/.

TU Delft Delft University of Technology

PHILIPS

# Contents

## List of Figures

## List of Tables

# Visualization Methods for Understanding Deep Neural Networks

Laura Jacquemod[1]
*E-mail: ljacquemod@student.tudelft.nl*

**Abstract**
Deep neural networks currently achieve state-of-the-art performance in the field of machine learning for tasks such as image classification and regression. However, because they act as "black boxes", they are hard to understand and improve. In this thesis, we present ConvViz, our visualization toolbox, allowing one to better understand, diagnose and refine convolutional neural networks. ConvViz is composed of state-of-the-art tools aiming at visualizing some parts of the network but also of tools to envision the network as a whole. For example, the role of each filter can be better understood thanks to a local visualization: input optimization using backpropagation in order to visualize the features learned. The variety of tools but more importantly the possible interactions make the toolbox useful for understanding. To evaluate this toolbox, the improvement for a specific use case was analyzed. The improvement is defined by both the increase in the accuracy measure as well as the decrease in time complexity provided by a new network defined after analyzing the visualizations of the initial network. Moreover, the generalizability of the toolbox has been examined.
The use case that has been predominantly studied in order to develop ConvViz is bounding box localization for objects on cardiovascular images. In fact, cardiovascular diseases can cause severe events like heart attacks or strokes. To avoid such events, an object can be placed in the arteries to open a blockage. Because an error could be lethal, very precise localization of this object needs to be provided to the doctors. While Philips already has a well-performing network, improvements are opportune and a need for understanding the decisions of the network is present. After analyzing the visualizations provided by ConvViz, it has been possible to increase the accuracy measure of the previous best-performing network by 5%. This increase in accuracy goes hand-in-hand with a decrease in the computational complexity by 95%, allowed by the removal of over 60% of the convolutional filters.
Finally, the IMAGENET classification task was considered and seeing the analysis made from the different tools, it has been concluded that ConvViz is well generalizable.

**Keywords**
Deep Learning — Neural Network — Visualization — Backpropagation — Dimensionality reduction — Medical imaging

[1] *Computer Graphics & Visualization Department, Delft University of Technology, The Netherlands*

## 1. Introduction

Deep Neural Networks (DNNs) have succeeded in providing breakthrough results in many pattern recognition tasks like object detection as well as image classification. One of their first ground-breaking results was achieved with ImageNet [2], a deep Convolutional Neural Network (CNN), which succeeded in reducing of over 10% the best published error rate at the time. Neural networks have also been recently used in order to show the preeminence of machine learning in terms of intelligence compared to humans, with, for example, the game of Go, usually described as the most challenging classic game for artificial intelligence [9].

The issue with using such networks comes from the fact that the improvement of DNNs is not straightforward. In fact, neural networks are black boxes due to the fact that the role of each component as well as the mechanism of the whole network, with its enormous number of parameters, are unclear. A common way of improving DNNs is by iterating many trial-and-error experiments. Actually, scientists make changes to their networks such as modifying the number of layers, the number of filters per layer or the activation function in order to see the influence on the results to reduce the error. As these decisions are made without guidance or knowledge of the internal performance, this method is extremely time-consuming. In fact, running a whole network can take several days. To analyze the internal performance of the network and guide a new design, it is, therefore, necessary to use visualization tools. In fact, those tools can help refine well-performing networks, but also diagnose why networks have failed to converge during training as well as understand the decisions made by neural networks.

Research in deep learning is currently focusing on this area. Visualization methods can be separated into two clusters. Firstly, visualizations of a particular part of the network, such as a neuron or a filter's weights, with tools such as back-propagation or input optimization, defined hereafter as local visualization, can, for instance, help understand the role of each neuron. Secondly, visualizations of the whole network can be gathered, defined as global visualization, in

order to see correlations between filters or layers for example. ConvViz, the toolbox developed during this thesis, is bringing together state-of-the-art visualizations from both types, with interactivity in order to understand networks better and to have more certainty on the conclusions made.

In order to see how straightforward each visualization is and how big the accuracy increase can be, a main case, from Philips Research, based on cardiovascular diseases, has been used. In fact, those diseases represent the leading cause of death worldwide. Diseases such as heart valve problems, strokes or heart attacks are often related to atherosclerosis, a condition that develops when a substance builds up in the walls of the arteries. This condition needs to be spotted as fast as possible and then be treated in order to avoid a decrease in blood circulation. During any operation, as well as for inspections in order to see the evolution of the blockage and reduce the revascularization rate, the object injected needs to be localized on 2D X-ray images. While the localization can be done after the X-ray scan for inspection, it needs to be performed in real-time during an operation, adding to the obvious constraint of precision. While Philips had already developed a well-performing network, an understanding of its decision was needed as well as an improvement of accuracy.

This report provides a description of the two types of visualization methods developed, as well as their evaluation according to use cases. The paper is organized as follows: Section 2 provides a background on neural networks and Section 3 describes the state-of-the-art in visualization of networks. Then, the dataset and networks used for the tests are presented in Section 4 as well as the requirements for visualization (Section 5). Sections 6 and 7 describe each type of visualization and Section 8, the interactiond between both. After analysis of the results from the cardiovascular use case use case in Sections 9 and 10 and the determination of the generalizability of the toolbox by testing different use cases and, therefore, networks (Section 11), we conclude that while local visualization is useful for improving and understanding a network, it is necessary to also consider global visualization for better understanding and bigger refinement of the network. Finally, section 12 discusses the future possibilities and challenges.

## 2. Background

The most common problems that are to be solved by Neural Networks are pattern recognition tasks such as image classification or regression. For these tasks, good feature extractors are usually implemented. Features are "good" if they are discriminating special aspects while being invariant to other irrelevant aspects of the image. While non-generic classifiers fulfill this condition, they do not generalize well. One solution is to design feature extractors by hand, a task that is time-consuming, requiring a lot of domain expertise and specific knowledge for each application. Another solution is

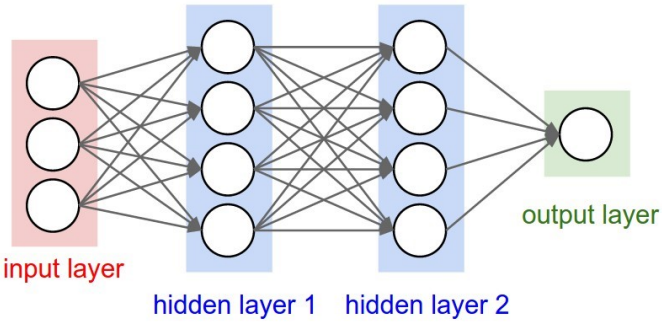



**Figure 1.** Example of a Neural Network with an input layer of 3 inputs, two hidden layers of 4 neurons each and an output layer

to provide a general-purpose procedure that learns features from the data, such as convolutional neural networks. As described by LeCun Y. et al. [10], neural networks are composed of neurons, which are warping the input space to a new output space in a non-linear way. By iterating through several non-linear mappings, the network becomes sensitive to good features and can warp the input space to an output space where the classes or localizations are linearly separable. In terms of architecture, DNNs can be modeled as graphs of neurons. A neuron is a small unit that receives several input signals, interacts multiplicatively with each of them according to weights, and outputs the sum of all of them after applying an activation function that is nonlinear.

While many kinds of neural networks exist, each with their specificities, we will introduce Multi-Layer Perceptrons as an example to describe neural networks and we will focus our analysis on Convolutional Neural Networks (CNNs), which have been behind recent successes.

### 2.1 Multi-Layer Perceptron

Not only multi-layer perceptrons but also neural networks in general are organized in layers: the input layer, one or more hidden layers made up of neurons and the output layer which provides the output of the network. The different layers are interconnected so that the outputs of neurons become the inputs of the neurons in the following layer. This structure is depicted on Figure 1, where each circle represents a neuron.

For Multi-layer Perceptron networks, each layer is fully-connected to the previous one, as it can be seen on Figure 1. More precisely, each neuron of a specific layer gets as input the outputs of all the neurons of the previous layer.

Mathematically, each neuron can be written as in $y = f(\sum_i w_i x_i + b)$ where $x_i$ are its inputs, $w_i$ the weigths, $b$ the bias, $f$ the activation function and $y$ its output. Activation functions, such as the well-known sigmoid ($f(x) = \frac{1}{1+e^{-x}}$) or ReLU ($f(x) = max(0,x)$), are specific for a layer, usually even for the whole network, and allow the warping to be non-linear.

From this structure, the learning process consists of finding the parameters, i.e., $w_i$ and b, for each filter, given a set of training data representative of the problem. In fact, the inputs at hand are divided into three groups:

- Training set: To optimize the network by learning parameters
- Validation set: To evaluate the network's performance
- Test set: To get the accuracy in the "real world"

Using the training set and its known labels, the minimum of the error function in the weight space is looked for using the method of gradient descent, starting from random weights and no bias or, in the case of fine-tuning, previously found values.

### 2.2 Convolutional Neural Networks

Convolutional Neural Networks, which consider that inputs are images, are composed of three different types of layers: convolutional, max-pooling and fully connected, the last type being the one described previously in the case of Multi-Layer Perceptrons. Figure 2 provides an example of a typical architecture of a CNN and will be used as a reference to describe the different layers.

Convolutional Layers are composed of learnable filters. Each filter, extending through the depth of its input, is slid across the input volume and a dot product is computed between the local region of the input and the filter's learned weights. A local input is represented by the red square with perspective along the depth on layer Pool1 of Figure 2. This operation is similar to performing a multichannel image convolution with the learned filters, hence the name of the layer. After applying an activation function, each filter produces a 2-D activation map, each represented by one large square, as one can see in blue on layer Conv2 of Figure 2. Finally, the activation maps are stacked to form the output of the layer, being the input volume for the next layer. On Figure 2, the first convolution takes as an input the cat image and has four filters while Conv2 has six filters. This mechanism can be also interpreted as the computation of a layer of neurons: each entry of the output volume is the output of a neuron that considers only a local region of the input and one filter.

Max-pooling layers perform a sample-based discretization process via down sampling. The dimensionality of the input is reduced by applying a max filter to subregions of this input. These layers are commonly used to reduce computation and to control overfitting [11].

All types of neural networks share a common issue: they work as black boxes because of the enormous number of parameters involved in learning. Typically, a neural network has tens of layers, each consisting of hundreds of neurons, which are connected by millions of connections. The problem of understanding neural networks comes from the lack of information on each of these components as well as from the massive number of components, making it hard to perceive the overall structure and mechanism of neural networks.

For more thorough description of neural networks, see the extensive review by LeCun Y. et al. [10].

## 3. Related Work

As introduced previously, in order to better understand, diagnose and refine neural networks, visualization can be used. The state-of the-art visualizations in deep learning can be distinguished into two major types. Firstly, local visualization, or the visualization of a particular part of the network, such as a neuron or a filter's weights. This visualization can help for diagnosis; for example, in order to spot "dead" filters, never activating. Secondly, visualization of the whole network, called hereafter global visualization, can be used to see correlations between filters or layers for instance, important for refinement of the network and understanding of how the network works on a larger scale.

### 3.1 Local Visualization

As the issue faced by neural networks is both the understanding of each local element such as filters or neurons and of the overall structure, one first idea is to visualize each part separately.

#### 3.1.1 Inversion Visualization

To understand a network, one can visualize the weights of each filter [5]. Seeing that the first layer is a weighted combination of the input space, the weights can be visualized and interpreted directly, such as edge detectors, as seen extensively on Figure 3. Furthermore, well-trained networks usually display smooth and unimodal filters. If on the contrary the patterns are noisy, it is an indicator that the

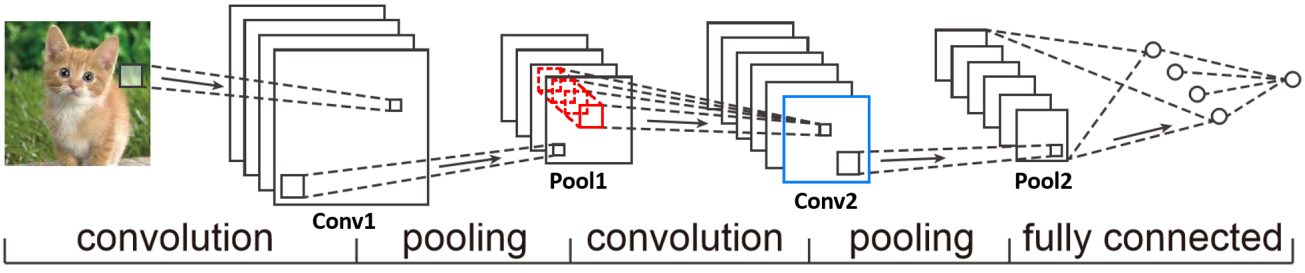



**Figure 2.** Typical architecture of a CNN, with two convolutional layers [1]

network might not have been trained enough, or that there is overfitting [11]. For deeper layers, the visualization is not straightforward to interpret because the filters are as deep as the number of filters present in the previous layer. It is still possible to have a similar visualization, by performing a linear combination of the visualizations of the previous layer' filters to which it is most strongly connected to [12]. This method is simple but it does not take into consideration the non-linearity of the activation function nor all the filters from the previous layer, potentially removing relevant information.

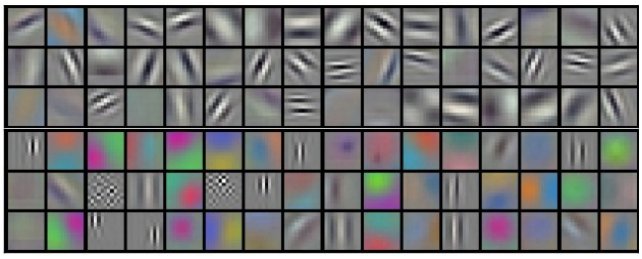


**Figure 3.** Examples of inversion visualization for the 96 first-layer filters on the IMAGENET classification task [2]

Hence, a deconvolution method was introduced by Zeiler et al. [4] to visualize filters from deeper layers. A deconvolution network, DeconvNet, inverts the data flow of the network. Typically, a backward pass is done from the layer of interest to the input space. Since the goal is to visualize one particular filter, only its activation map is not set to zero while the ones of all the other filters are. This layer is the input to the deconvolution network and the output is a probability map of the same size as the input image. To perform the reverse actions of the CNN, DeconvNet is composed of Deconvolutional, ReLU and Unpooling layers. Unpooling is carried out thanks to switches saved from a first forward pass, as shown on Figure 4 and Deconvolution is associating each input activation to multiple outputs, out of the different possibilities, as explained by Noh et al. [3].

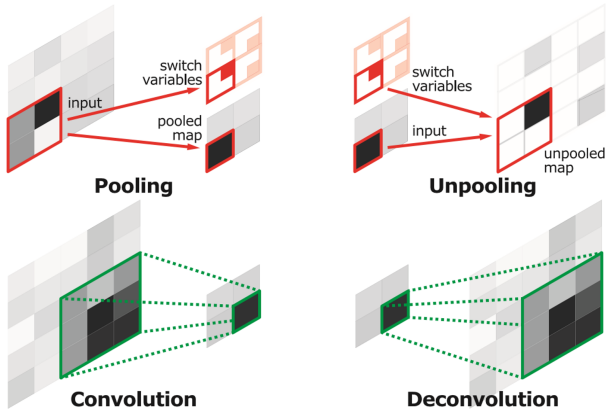



**Figure 4.** Illustration of deconvolution and unpooling operations [3]

Backpropagation, a gradient-based visualization, has been introduced as a generalization of the DeconvNet, as it can be applied to any layer, not just a convolutional one. The difference comes from the backward pass through the ReLU layers: while DeconvNet removes negative gradients, displayed as yellow locations in Figure 5, backpropagation suppresses neurons that had not fired during the forward pass, depicted in pink on Figure 5. Finally, guided-backpropagation, introduced by Springenberg et al. [13], is a combination of both methods, backpropagating only positive gradients through positive activations, working well without switches and leading to reconstructions that are more accurate. DeconvNet performs better than other methods when there are maxpooling layers, but it does not produce sharp visualizations for higher layers [4], on the contrary to backpropagation and guided-backpropagation.

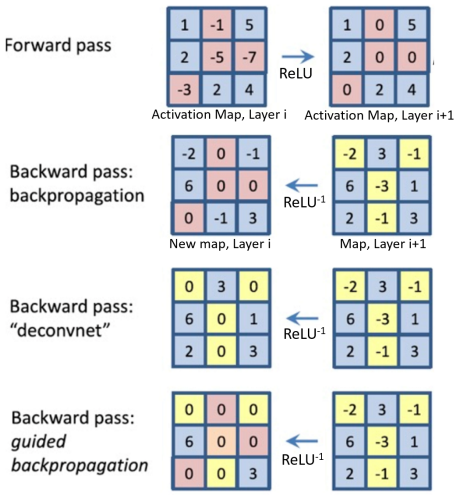



**Figure 5.** Methodology for performing the backward-pass for different visualization approaches [4]

### 3.1.2 Feature Visualization

Each filter has a certain activation map for a specific input image. The higher an activation, i.e., the value of a neuron, the more the filter is reacting to the input image and, therefore, the more the image represents that filter. A first method that was introduced by Erhan et al. [5] was to find the images in the dataset that would maximize a neuron related to the filter of interest. They later introduced a generalization [14] using the inversion method presented previously: instead of looking at images from the input space, the input image is optimized, starting from random noise. In that case, not only one backward-pass is performed as in the inversion visualization method but a succession of several forward and backward passes in order to perform the optimization. An example is provided in Figure 6, with nine random initializations per filter, on a network trained on grey-valued handwritten digits. Even though this method provides better results, as we can see clearly that some filters look like pseudo-digits, both visualizations lead to the conclusion that each filter learns a specific feature of the image and that the learned features have more complex structures

for filters from deeper layers. These features are similar to the ones manually implemented for feature-based methods in pattern recognition.

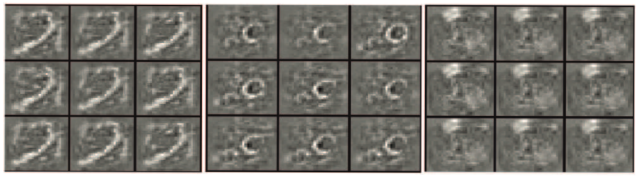


**Figure 6.** Examples of feature visualization on the $3^{rd}$ layer of a Stacked-Denoising Auto-Encoder [5]

#### 3.1.3 Focus Visualization

The deconvolution method allows one to grasp how filters are responding to an input image, therefore, how networks outputs are obtained. On the contrary, focus visualization describes what information is used by the network according to a specific input image [15]. Areas of the image that are discriminative with respect to the output selected, either a label or a filter, are highlighted, as it can be seen on Figure 7 with the top 25 filters of the last convolutional layer of a network for anatomy classification. In fact, a pixel from the input image is discriminative to a label or a filter if changing it changes the result for the selected output highly. "Dead" filters can be found as the gradient will be null and no back-propagation will be seen. As dead filters are a symptom of high learning rates [11], it is possible to know how to refine the network if they are encountered.

An improvement of this method is provided by Zintgraf et al. [16]: instead of having a binary output as to whether a pixel is discriminative for a selected output or not, it explains whether the feature is discriminative for a specific output, against it or if it has no influence on the value of that label or filter. Thanks to this signed feature's relevance, it is possible to get contextual information.

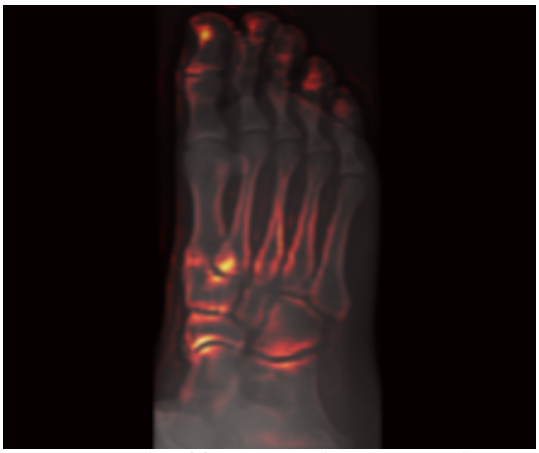


**Figure 7.** Example of focus visualization, on the original image, for the top 25 filters of a network trained for anatomy classification [6]

#### 3.1.4 Conclusion

The methods presented can be considered as local because they aim at understanding a specific part of the network such as a filter or a neuron. Because the number of parts making up a network is enormous, too many local visualizations should be analyzed, requiring global visualizations for having an indication of the interactions between the different parts, to understand and refine the network.

### 3.2 Global Visualization

A first approach that has been used extensively to improve neural networks is to plot the quantitative information such as the error rate. This method has been used by Erhan et al. [17] in order to see the influence of pre-training on the performance of the network. This allows the user to compare networks or visualize overfitting, for example. However, this kind of visualization does not provide qualitative information.

For that purpose, representation visualization can be used. Dimensionality reduction, such as t-Distributed Stochastic Neighbour Embedding (t-SNE) and MultiDimensionality Scaling (MDS), maps high dimensional datasets into lower dimensions to be able to visualize those datasets. It is a crucial tool for CNNs since there are up to thousands of features. Rauber et al. [7] visualizes the relationships between learned representations of input images by applying t-SNE on the activation map of a particular layer of the network. Clusters and outliers provide information on the set of inputs. One can also visualize the evolution "inter-layer", with each point being an input image, as shown on Figure 8, to know how many layers are sufficient.

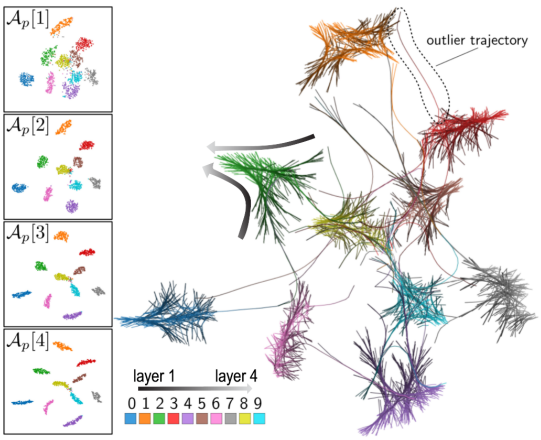



**Figure 8.** Example of representation visualization with inter-layer evolution [7]

Another dimensionality reduction-based visualization of neural networks, as in ReVACNN [18], is the 2D embedding view, for a specific layer, where many parameters can be visualized: vectorized filter coefficients, filter gradients, activation maps as well as activation gradient maps. By performing clustering, redundant filters can be outlined.

A more complete overview of the state-of-the-art global visualizations can be found in Appendix A.

### 3.3 Conclusion

While local visualization provides insight into a particular part of the network, it fails to provide information on the network as a whole, with the interactions between those parts. Global visualization succeeds in doing so, usually via dimensionality reduction or graph representation. This analysis has led to ConvViz, a toolbox putting together these state-of-the-art techniques in order to have a better understanding of the network.

## 4. Data

There are two types of data that can be visualized: the input images, already pre-processed, and the trained network. The inputs can be visualized in order to see how they are being processed by the network, which features are considered as of interest for the network and the output can be visualized in order to analyze the performance of the network. The network itself is also of interest, with its filters, layers, links, etc.

Concerning the input images, even though the amount of data needed to train a neural network is problem-dependent, the data needs to span the problem input space. This seek for data abundance complicates the visualization, since the more data, the more clutter in the visualization. It is true that not all available inputs need to be visualized, but only the ones from the validation set seeing that the aim is to know how to tune the network, the specific goal of the validation set. In this thesis, we focus on the object localization task, for which only 20% of the set of inputs is kept for validation. However, the abundance of input images still remains an issue for visualization and has to be considered when designing visualization tools.

To describe more precisely the dataset considered for the main use case of object localization on cardiovascular images, the inputs are 2D grey-scaled images, which are frames, extracted from videos. Those videos are gathered by a Philips biplane X-ray system: Allura Xper, a system that typically records fifteen frames per second, with the possibility to get from 3.75 to 30 frames per second. This system has two modes:

- Navigation mode: low dose of X-ray
- Acquisition mode: high dose of X-ray

While the input images are gathered from the acquisition mode, the localization needs to be robust and provide good results in the navigation mode as well since the doctor would like to have a localization also when navigating.

All of the frames contain an object, such as sternal wires, valves or stents. Only one object is usually present per frame and the task of the use case, for which the network has been trained on, is to localize this object. Figure 9 shows the example of a frame coming from the video that will be shown throughout the report. As in most of the cases, only one object with two markers is present. Figure 9 depicts a stent and one can see two markers with a wire that is used by scientists for manual localization. It is of importance to note the utility of markers and the wire in order to see if the network is focusing on the same regions of the image as a doctor would do.

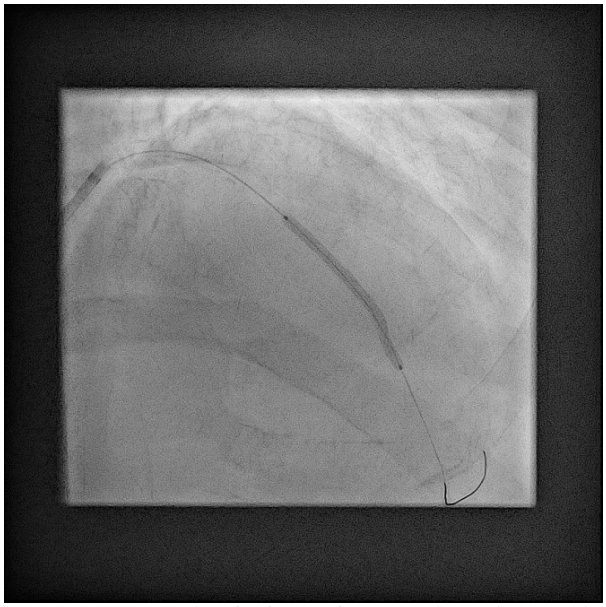


**Figure 9.** Example of a frame from a selected example video used throughout the report

**Table 1.** Network configuration: the convolutional layers' parameters are denoted as 'conv<receptive filter size>-<number of filters>'.

| Network Configuration |
|---|
| input (512 × 512 greyscale image) |
| conv3-32<br>conv3-32<br>conv3-32 |
| maxpool2 |
| conv3-32<br>conv3-32<br>conv3-32 |
| maxpool2 |
| conv3-32<br>conv3-32<br>conv3-32 |
| maxpool2 |
| FC-32,768 |
| FC-250 |
| soft-max |

Concerning the network, it needs to be already trained, with its weights and bias accessible for each layer. Even if it is demanding, the different tools introduced in this report are only of interest for trained networks. The main

network of interest used throughout this thesis is a VGG-style network, a network architecture first introduced by the Visual Geometry Group for the IMAGENET challenge [19]. The whole architecture of the network is described in detail in Table 1. It needs to be noted that there are nine convolutional layers, an important information seeing that only convolutional layers are being visualized in ConvViz.

Since the aim of this thesis is to develop a visualization toolbox in order to understand better deep neural networks and, therefore, be able to improve them, the data, both the network and the input images, should not influence the effectiveness of the visualization. Hence, in order to evaluate the generalizability of the different tools provided by ConvViz, different networks have been visualized for the object localization task with the same training set but also another use case was considered with different input images as well as networks, as described hereunder.

The second use case is, instead of localizing objects on cardiac images, classifying natural images into different sets. More precisely, this use case comes from the well-known Large Scale Visual Recognition Challenge (ILSVRC) and the network visualized is the one that won the competition in 2014 in terms of localization and arrived second for classification: the VGG-16 network [19]. For generalization purposes, as localization has been dealt with thanks to the previous use case, the classification part of the network was examined. In that case, the inputs are 2D-coloured images. Moreover, the issue of abundance of data is even more concrete since the dataset is made up of over 1.3 million images and 1,000 predefined categories such as "leopard, Panthera pardus" or "Eskimo dog, husky".

It needs to be noted that all of the networks analyzed are convolutional networks seeing that, as explained in Section 1, they have been behind the recent successes and, therefore, the different tools from ConvViz are focused on visualizing convolutional layers.

# 5. ConvViz

ConvViz was developed because of the need to understand the inner mechanisms of neural networks. It is a combination of existing techniques and previously available information, for better understanding of networks. In fact, using several techniques allows extracting more information on the network as well as double-checking to reduce possible errors in making hypotheses concerning the decision making of the network at hand. Moreover, visualizations can help see if the important parts of an image according to the network are identical to what experts consider as important. If it is the case, it can help users build trust with their clients since it is possible to describe the process of decision of the network or, if, on the contrary, the image parts are different, one can investigate and find new areas that could be informative for decision-making.

## 5.1 Requirement analysis

Thanks to discussions with CNNs developers, the following requirements have been identified.

**1 - Exploring the learned features of filters.** To begin an analysis, one needs to examine the quality of learned features layer by layer to understand what features a network is looking at in order to make its decisions and where potential problems might come from. However, because a deep CNN can have up to hundreds of layers and thousands of neurons in a layer, it is not possible to explore all of the filters. An ordering of filters needs to be created to provide an overview of the learned features of the important filters only.

**2 - Revealing the most important filters and their top neurons.** As hinted previously, an ordering of filters needs to be introduced and, for those important filters, with specific inputs, the maximum neurons need to be visualized. The localization of the top neurons allows understanding of which spatial locations of the input are most important for the network, allowing understanding of the functioning.

**3 - Providing an overview of dead filters with a cause of death.** It is a common problem for neural networks to have dead filters. Seeing that they add computation while not being useful for the final decision, it is important to spot them in order to either remove them or train them again. Moreover, understanding the reason to their death is useful, e.g., if all of their weights are negative, for understanding how they need to be re-trained.

**4 - Analyzing the correlations between filters.** Another issue similar to the one of dead filters is redundant filters. It is required to be able to see the correlations between filters in order to spot redundant ones but also to check if all the possible features are being looked into.

**5 - Linking global aspects of the network with local ones.** Global aspects, such as correlations between filters, need to be linked with local aspects, such as the pattern learned by filters. This link is required to get an understanding of each aspect of the network as well as their interactions.

**6 - Exploring the set of inputs.** In order to construct a well-performing network, it is essential to understand the dataset considered. It is needed to be able to browse through the input space and to check how the network is understanding this space layer by layer.

## 5.2 System overview

We have therefore developed ConvViz according to the previous list of requirements. From a trained convolutional network and the corresponding validation dataset as the inputs, visualizations are extracted and the toolbox is composed of two major parts:

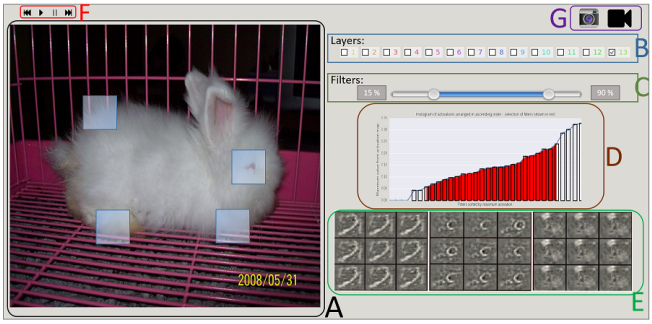



**Figure 10.** Design of ConvViz for local visualization

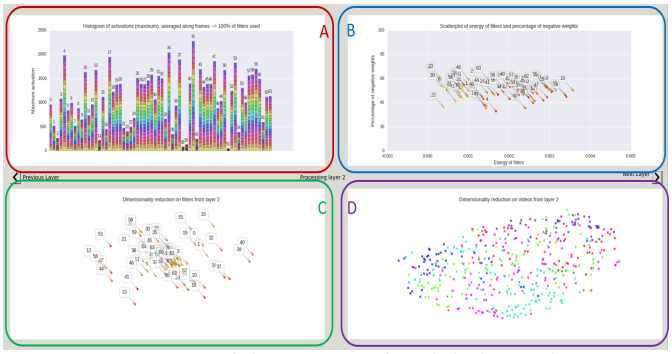



**Figure 11.** Design of the ConvViz for global visualizations

- Local visualization, its panel is depicted in Figure 10, with the following tools:
  - Localization of maximum neurons for selected filters, on a specific input, following the focus visualization method [6], area A (Requirement **2**) with possibility to browse through inputs and, for videos, to browse through frames, area F (Requirement **6**)
  - Features learned by each filter, area E (Requirement **1**), implementing the inversion method [14]
  - Plot of the importance of each filter, according to a measure introduced to order filters, area D (Requirement **2**)
  - Interactivity to select the layers of interest, area B and the filters ordered according to the measure introduced, area C

- Global visualization, depicted in Figure 11, made up of four plots:
  - Cumulated histogram of activations for each filter, to detect dead filters (Requirement **3**), area A
  - Plot to understand dead filters (Requirement **3**), area B
  - Correlations between filters (Requirement **4**), area C, adaptation of ReVACNN [18]
  - Correlations between inputs, seen by the network, layer-by-layer (Requirement **6**), area D, adaptation of the method by Rauber et al. [7]

ConvViz also needs to have a link between global and local visualizations. It needs to be possible to select filters from the global visualization in order to see them more specifically with the local visualizations in order to confirm possible hypotheses made from the first observations. This interactivity will, therefore, satisfy requirement **5**. Finally, in order to show the visualizations to customers in an easy manner to build trust, it is required to be able to save the visualizations, both for videos and images, as shown with area G.

## 6. Local Visualization

### 6.1 Convolution

For the first convolutional layer, a simple way of understanding what the filters are doing is to look at their weights and the convolution on a selected input image since it shows what is kept from the image and, therefore, the features learned (Requirement **1**). In fact, it is possible to see the weights of the filter, in order to see what shape it detects - typically corners for the filter in Figure 12 -, the result of the convolution and the output of the layer, after adding the bias and passing through the ReLU layer, with a "coolwarm" color map to see the result in an easier way. As explained in Section 3.1.1, for deeper layers, the input image to the filter has a depth equal to the number of filters from the previous layer. For the use case considered, there are 32 filters for all the layers in the network at hand, which would require analyzing 32 convolved images for each of the filters.

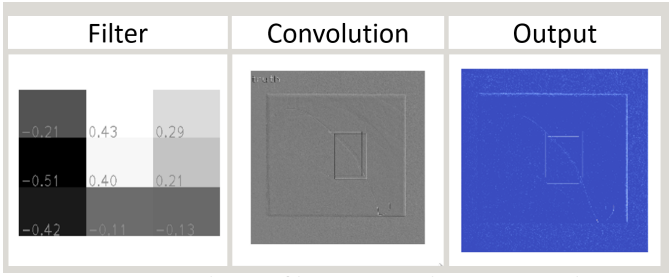



**Figure 12.** For a chosen filter: its weights, its convolution with the input image and the output of the layer after adding the bias and passing through the ReLU

### 6.2 Feature visualization

An improvement of the convolution, because it extends to all layers and it does not depend on the input image, is, as described in Section 3.1.2, for a specific filter, to optimize the input image to look at its learned feature. With the weights of the trained network given as input fixed, the gradient is computed with respect to the input pixels. Starting from a specific image and the calculated gradients, backpropagation of the gradients is, therefore, performed in order to maximize the mean activation map of the given filter.

To accomplish this optimization, the code developed for Keras Visualization Toolkit [20] was taken as a reference, with slight changes made to it. Typically, while constraint functions can be used in order to have smoother results during deconvolution, none of them was applied to our visualization because they were adding circular shapes to simple and rough shapes in the first layers, for the task studied.

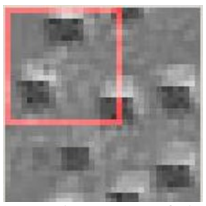


**Figure 13.** Feature visualization of the top filter from the sixth layer with the object localization network

It is common to have a random image of the same size as the input size as the starting point of the optimization. Seeing that the effective filter size is smaller than the size of the input, choosing that size forces the optimization on several pixels, adding redundancy and overlapping in the optimized image. To remove that issue, we have chosen to have the initial image of the size of the effective filter size or of the size of a minimum threshold if the effective filter size is smaller than this threshold. Its value was found, empirically, of 35 pixels, because, according to tests, if the size is under this threshold, the optimized image will be too small and the shape learnt not easily recognizable. In case the effective filter size is smaller than the threshold, in order to have a clear view on its size, a red box shows that size, as it can be seen on Figure 13. In that case, one can see that the filter can recognize up blobs and it is possible to estimate the size of the blobs recognized. Starting from a random image succeeds well for first layers but the optimization often does not update for deeper layers. In fact, the deeper inside the network, the more specific the filter. Hence, it becomes more likely that the shape of the filter will not be represented in a random image and that the whole activation map will be equal to zero, leading to a null gradient and not allowing backpropagation to be performed.

Typically, when using the deep and wide VGG16 network on the image classification task, for the last convolutional layer, with 50 filters chosen randomly out of the 512 available filters, on average only 11 of them do not get stuck directly after the first forward pass, with 15 of them shown on Figure 14. Black patches represent input images that did not optimize. A first greedy and slow method would be to pass every possible patch from all the different images from the validation set as the initial input of the optimization and try to optimize it to finally extract the one with the lowest loss value. While this method would end up with the best possible visualization from the dataset, it also provides an optimization that is dependent on the chosen input image. In order to remove that dependency, the following approach has been taken:

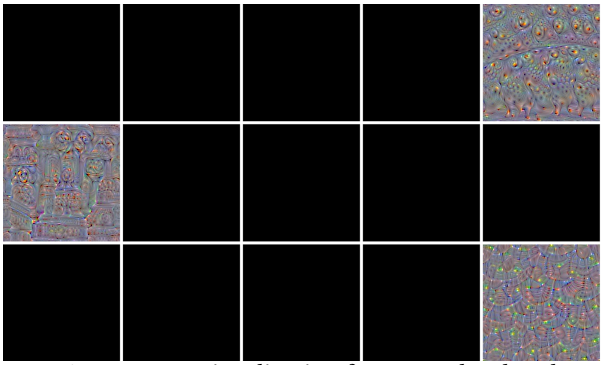


**Figure 14.** Feature visualization for 15 randomly selected filters, from the last convolutional layer for the image classification task, starting from a random image. Black filters are filters that did not resolve the noisy inputs

1. Choose randomly 4 images from the validation set, keep only the center parts of the size of the effective filter size and compute their mean by adding them up
2. Perform input optimization on this mean image
3. If the optimization:
    - Has no gradient: If it is less than the $100^{th}$ attempt, retry with 4 other images, otherwise stop the optimization
    - Has been performed: Subtract the input image to the optimization and perform input optimization on that new image

In this way, the optimization does not stay stuck due to an absence of gradient and the dependency on the input image is reduced both by blurring the input image by calculating the mean over different images and by subtracting the input image. These results are present in Figure 15, where the selected filters are the same ones as in Figure 14: one can see that all of the filters are being optimized and do not get stuck without a gradient. Checking that the optimization is not done over a hundred times is needed because optimization might not be possible for some filters. For example, if the weights of a filter after the first hidden layer are negative and its bias is also negative, which is rare, the activation map would be zero for all entries and no backpropagation would be possible, whatever the input. No optimization would be provided to the user, which is not a problem because it is a rare case and it shows a dead filter, which is of importance to the user. More information on this research, how the approach described previously was designed as well as the images of the feature visualized, using both use cases, can be found in Appendix B.

From these visualizations, it is possible to see what the learned features are per filter. It helps understanding what features the network is considering in order to take its decision but also to see if the shapes are smooth seeing that

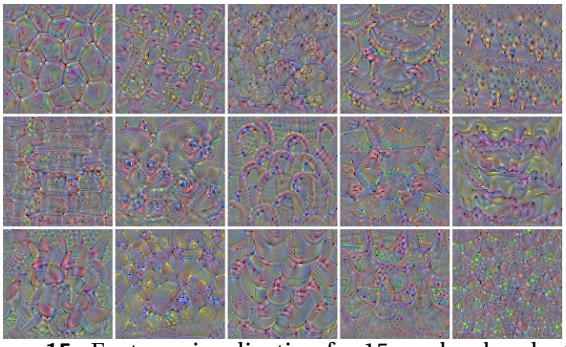


**Figure 15.** Feature visualization for 15 randomly selected filters, from the last convolutional layer of the network trained for image classification, starting from the mean of four randomly selected images

noisy patterns are indicators of overfitting or, on the contrary, that the network was not trained enough, as explained in Section 3.1.1. Finally, a hypothesis on dead filters can also be done if no optimization is performed.

## 6.3 DeconvNet

### 6.3.1 Method

To understand how a network takes its decision, one can also look at which parts of a specifically chosen input image are considered as important, or influential for the decision, for each layer of the network, as described in Section 3.1.3. For a specific layer chosen by the user, as shown on Figure 16a, an ordering of its filters is created and the top ones according to this measure are found, Figure 16b, then the maxima of the layer's activation map are kept and the rest of the activation map of that layer, including the activations from other filters, are put to zero, as depicted in Figure 16c. From this updated activation map, a backward pass is performed using the DeconvNet, as described by Springenberg et al. [4], in order to get a focus visualization, i.e., a visualization of which parts of the input image the network is focusing on or considering as important. The different steps of the backward pass through to the depth of the network, using deconvolution and unpooling layers as explained in Section 3.1.3, in order to get to the input layer, are shown on Figure 16d.

This backward-pass method in order to get a focus visualization has many parameters; their description as well as the choices made for the development are described next.

- An ordering of filters is needed in order to define a "top filter". We define top filters as the ones with their learned features clearly present in the input. Mathematically, it means that the maximum value of the activation map of this filter is the highest compared to the ones of other filters. It is also possible to consider the mean instead of the maximum seeing that it would mean that on average the shape recognized by that filter is highly represented. However, this is not coherent with the previously made definition of a top filter seeing that the aim is to have a clear shape and not several look-alike shapes. Therefore, an ordering of filters is made by considering the maximum of their activation map.

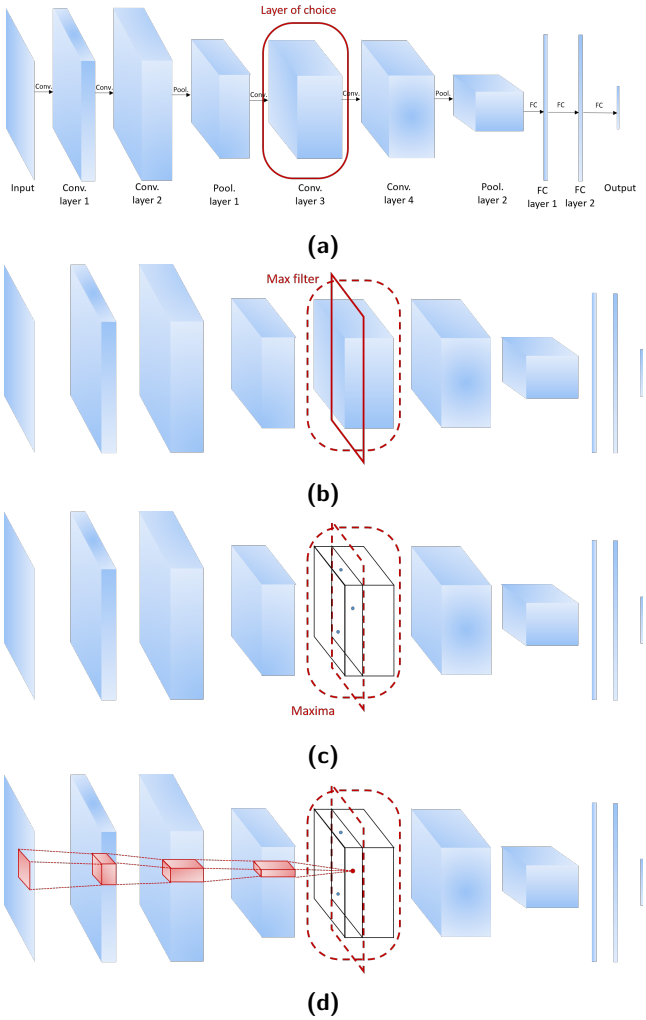



**Figure 16.** Description of the consecutive steps performed for focus visualization. Conv. stands for Convolutional, Pool. for Maxpooling and FC for Fully Connected. (a) User selection of a specific layer with a specific input already forward passed (b) For the selected layer, calculation of the top filter (c) Extraction of maximum values from the activation map and setting the rest of the map to zero (d) Backward-pass of the selected maxima neurons to the input space

- The number of top filters to visualize is another parameter. The number of filters of interest is difficult to predict: it depends on the network as well as on the chosen input image. Moreover, while most of the information comes from the top filters, the difference between two slightly different decisions in the final output might come from the information of the "last" filters, i.e., filters with the lowest maximum activation on their activation map for that specific input image; making it necessary to know also what those filters are focusing on. Therefore, it is the up to the user to choose

the amount of filters to visualize. An ordered histogram of maximum activation for all the filters is displayed, depicting the ordering of filters as described beforehand. According to this histogram, the user can choose the minimum filter and the maximum filter to visualize in order to envision all of the filters between those bounds.

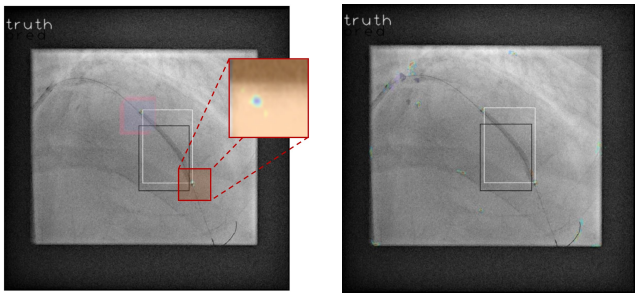
- Concerning the choice of the values of the activation map to keep for the backward pass, one first option would be to extract only the neurons reaching the maximum activation value for the chosen input and filter. Nevertheless, some neurons might have an activation close to that maximum and be of interest. Empirically we decided to keep the neurons with an activation value over $0.98 * max$ for the backward pass, with $max$ the maximum activation value of the filter. This choice of threshold was motivated by the fact that no additional neurons were found with a smaller threshold and when taking only the neurons with values equal to the maximum, fewer neurons were extracted.
- Finally, different methods for performing a backward pass exist such as backpropagation, DeconvNet, guided backpropagation, CAM (Class Activation Mapping) or grad-CAM. Seeing that the toolbox needs to be generalizable, it needs to work for different tasks. CAM approaches are usually performed on the whole network and are class discriminative. As the first case of interest is not classification but regression, they are not the right methods for ConvViz. Concerning the other methods, according to their description in Section 3.1.1 and the fact that maxpooling layers are generally part of networks, the DeconvNet approach was chosen.

### 6.3.2 Visualization

Since for each of the neurons kept on the activation map, the backward pass results in an activation map inside a square of the size of the effective filter size, it is of importance to show that square filled. Not only does it make the localization more visible, but it also allows identification of filters thanks to colouring, as seen on Figure 17a with each filter represented in one colour. In fact, it is important to know that some filters are looking at the region of interest and others at unimportant parts of the image, but it is even more important to know their identity to improve the network. It is also possible to remove square filling, to avoid clutter, as depicted in Figure 17b. It was found that the effective filter size found experimentally was the same as the one calculated with formula 1 for convolutional layers. More explanation on how those formulae were found can be found in Appendix C.

$$\begin{cases} size_{next} = & size_{prev} + (F-1) * S^p, \\ & \text{if prev is a convolutional layer} \\ & size_{prev} + (F * S - 1) * S^p, \\ & \text{if prev is a maxpooling layer} \end{cases} \quad (1)$$

with p the number of maxpooling layers before the targeted layer, F the spatial extent of convolutional layers and S the stride of maxpooling layers.




(a) (b)

**Figure 17.** Backward pass (a) with square filling and zoom of one activation as well as (b) without square filling

Only the backward passes for each filter are saved and the squares are added according to the non-zero values from the backward pass. Typically, for each filter, the square filling is added with the top left corner being the first non-zero value from the matrix of the backward pass and a copy of the backward pass is made with all the activations inside that bounding box put to zero. Then the same procedure is applied to find non-zero values on the copied and updated matrix until that matrix is full of zeroes. In fact, different squares might be present for one filter seeing that the neurons with an activation over a specific threshold are kept and not only one maximum. Lastly, the opacity of those squares is set to 0.3 in order to be able to see them as well as the background made up of the input image, as seen on Figure 17a.

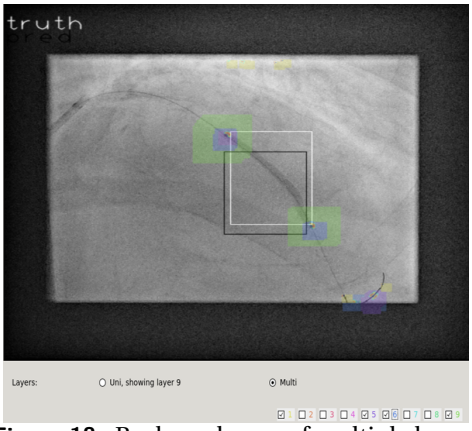



**Figure 18.** Backward-pass of multiple layers

For a video, while each frame is treated separately, it is possible to see the evolution of selected filters across frames and if the regions of interest change from frame to frame or follow the same spot. This provides additional information on the role a specific filter.

Finally, it is also possible, with ConvViz, to see multiple layers backward-passed at the same time, as shown on Figure 18. This visualization is important in order to see the evolution of the regions of interest across layers. The depth

in the network is shown thanks to the size of the filter - the deeper the layer, the larger the effective filter size - and the differentiation between layers is made with a categorical color palette.

## 7. Global Visualization

While local visualizations provide information on one aspect of the network such as a specific filter, it does not help the user understand how the network works globally because of the abundance of filters and layers and the interactions between the different parts. Therefore, global visualizations are required to get that aspect.

### 7.1 Histogram of Activations and Plot of Weights

The first plot that is available with ConvViz can be seen as a generalization of the histogram available in the local visualization. In fact, instead of visualizing the histogram of the maximum activation for each filter for one specific input, it is a cumulated histogram of those maxima over several inputs, as on Figure 19. Hence, dead filters can be spotted as well as the ordering of filters as defined in 6.3.1.

The local histogram is bias to the input: some filters might not be activated for one specific input but still be useful for other images. In order to avoid extracting the maximum activation of each filter for all the inputs, an analysis of the dependency of the results regarding the number of inputs taken into consideration was done. Hence, the histograms were calculated over the following sets of data:

- All the available inputs (training, test and validation sets) to have a global view of the filters
- All the inputs augmented in the same way as when the network was trained (rotations and flips) to remove bias as much as possible
- Validation set, made up of 463 images for the object localization task
- 10, 20, 50 and 100 inputs selected randomly from the validation set

It was found, for our object localization case, that when 10 inputs were selected, the histogram was slightly biased to the selected inputs: some filters were not activated because their learned features were not present in those inputs. However, as soon as there were at least 20 inputs, the cumulated histogram had the same shape as when all the inputs were used and augmented: same filters never activated, same level of activation per filter. It has therefore been chosen, for ConvViz, to select randomly 20 inputs from the validation set and perform a cumulated histogram of the maximum activation per filter, showing each input in a different colour. In this way, this choice can be generalized to a new case if it is possible to spot the uniformity across inputs: each filter more or less equally activated for each input means that there are enough inputs. Figure 19 shows the results for the sixth layer of the network for the object localization task.

This first plot, i.e., the cumulated histogram of activations, allows one to spot which filters are dead and should

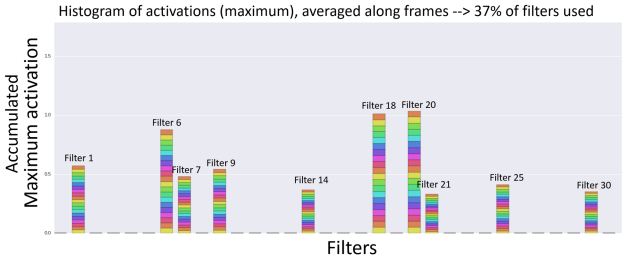



**Figure 19.** Histogram of the maximum activation for each filter of the sixth layer, cumulated over 20 inputs

be removed or re-initialized for training. However, one does not understand what causes a filter to be dead. This explanation is provided partially by the second plot provided by ConvViz: the scatterplot of the percentage of negative weights against the energy of the filter, as in Figure 20.

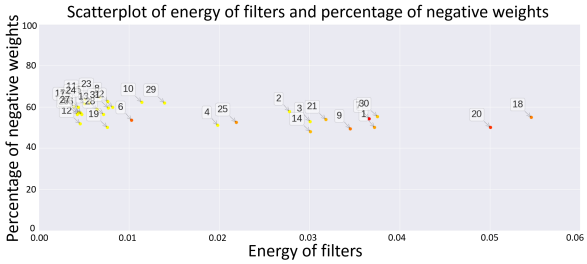



**Figure 20.** Scatterplot of the percentage of negative weights for each filter against its energy for the sixth layer

If a point is at the top of the plot, it means that 100% of its weights are negative and it can, therefore, be considered as dead seeing that, at least for inner layers, the output of the convolution will be negative, and removed by the ReLU layer. To check this fact, it is necessary to check the value of the bias for this filter, which is available when clicking on the filter. Otherwise, if a point is too much on the left of the plot, its energy is very low meaning that this filter will not be very useful for the final decision. It can, therefore, be considered as dead according to its position along the x-axis. After concluding that a filter is dead, it is possible to increase the bias, remove the filter or re-train it. Moreover, every filter has a specific colour representing how activated it is in a general manner: the more red a point representing a filter is, the more activated it is and the more yellow it is, the closer to dead it is. The colours are calculated according to the previous cumulated histogram. This helps the user relate this scatterplot to the previous histogram.

### 7.2 Dimensionality Reduction on Filters' Weights

While the previous plots were allowing one to better understand how activated each filter is and why it might be dead, this plot shows the correlations between filters within a layer. Similarities between filters in a given layer are represented by similarities in their weights seeing that only

convolutional layers are considered. Dimensionality reduction is performed, with each input being a vector with the weights of each filter, in order to see which filters are similar in an easier manner. Seeing that an unsupervised method is required, PCA is chosen to perform the dimensionality reduction, with the Euclidean distance between those vectors of weights. In the 2D space, each point is a filter that can be identified thanks to its annotation but also its color, in the same manner as in the previous scatterplot (Figure 20), to show how activated it is. This helps finding similar filters and seeing if some used filters are redundant. If redundant filters are detected, the user will be able to remove them or re-train them so that the network will look at other features and perform better.

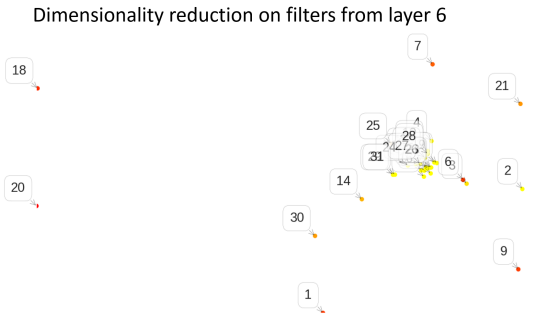
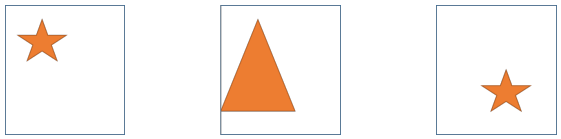



**Figure 21.** PCA on the weights of each filter of the sixth layer

An example of such a dimensionality reduction is provided in Figure 21 with the sixth layer of the object localization network. In this example, all the filters that can be considered as dead are grouped together at the center of the plot and, from the rest of the filters, no redundant filters can be located from the visualization of this layer.

### 7.3 Dimensionality reduction on images

The last plot in the global visualization provides information on the dataset at hand. It is of interest to see which type of data is available and more importantly, how the network is using this data. Some images might be similar according to the first layers but the network will learn higher-level features and see them as dissimilar in the following layers. The activation maps of all the filters from a given layer are compared to see similarities between images, as seen in Figure 23. It is of importance to reorder the activation maps by ascending order for each filter because otherwise similarities would be due to location whereas it should be due to the content of the input: similar patterns rather than identical location should cluster together. This is represented in Figure 22: the first image, 22a, represents the input image for which the closest neighbor needs to be found from the two other inputs, 22b and 22c. If no ordering of the activation maps is performed, seeing that the Euclidean distance is calculated in order to carry out the dimensionality reduction, the closest image will be 22b; whereas if a user were to group those images, it is more likely that it will cluster together the two images with the stars. The ordering of each activation map is, therefore, required. The input of the dimensionality reduction is made up of vectors, each being specific for an input, made up of the concatenation of all the ordered activation maps for each filter.



**(a)** Input image **(b)** Closest image to (a) without ordering of activation maps **(c)** Closest image to (a) with ordering of activation maps

**Figure 22.** Comparison of the closest image, in terms of Euclidean distance, with and without ordering of the activation maps

For example, with the network considered, seeing that there are 32 filters per layer and that the activation maps are of size 512*512 before any maxpooling is done, i.e., for each of the three first convolutional layers, each vector is of length 8,388,608. Therefore, the method used for dimensionality reduction is the t-SNE. This choice was made because it is a nonlinear dimensionality reduction technique well suited for high-dimensional data which preserves clusters and neighborhoods. Each point represents an input image and it is possible to see some clusters such as the green cluster on the left part of Figure 23. In fact, to discern patterns in an easier way, a colorization of the t-SNE is performed. Each color is selected according to the largest dimension of each vector before dimensionality reduction. The motivation behind this choice is the fact that high dimensional data is sparse and, therefore, close points should have a similar ordering of their dimensions. While the color itself is irrelevant, it allows one to see clusters more easily, adding information on how inputs clustered in high dimensions. Moreover, it is possible to select a specific cluster manually, either by selecting a specific color representing the inputs with the same largest dimension or by creating a shape around a desired cluster. In this way, by studying some images from one cluster, a pattern should be found and it should, therefore, be possible to understand why some images cluster together. By browsing through the dimensionality reduction visualizations made for each convolutional layer, one can see how the clusters evolve throughout the network and, consequently, how the network is understanding the data and which features it is considering as important.

An important parameter for the t-SNE is the perplexity. It represents the balance between local and global aspects of data and it can change dramatically the output. While the perplexity should be chosen between 5 and 50, according to t-SNE's developers [21], important changes can still be seen in that range. Actually, the right clustering is only visible when the correct value of perplexity is chosen. To perform this visualization in ConvViz, the value of 50 was chosen em-

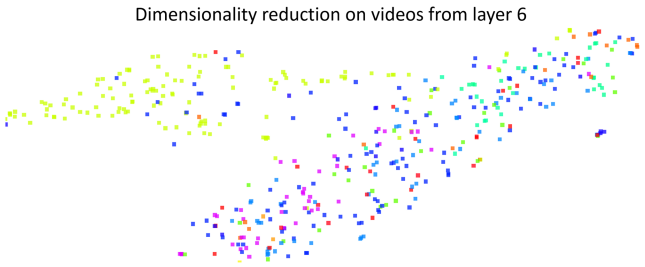



**Figure 23.** Dimensionality reduction, using t-SNE, on the activation maps of the $6^{th}$ layer for each input, in order to see clustering in the inputs

pirically for the perplexity because too many outliers were found for a higher value and one single cluster was always present, not allowing any analysis, for a smaller value. However, the perplexity should depend on the layer considered and on the network at hand seeing that the length of the vectors can highly change from one network to another.

## 8. Link between Global and Local Visualizations

A global view is necessary in order to understand what is going on with the network in a general way, independently of the input, in between filters from the same layer as well as in between layers. However, local visualization is also needed to visualize the influence of filters on a specific input, for example, or to understand better what a certain filter is doing. The complementarity of both visualizations has led to the possibility to select filters on global visualizations to have a better understanding of those filters thanks to local visualization. In each of the three global visualizations dealing with filters, it is possible to select filters interactively and then see only the selected filters on the local visualization. The selection is done manually, through a rectangle, circle, free shape or through cursor clicking so that it is possible not only to compare the filters that are close by in those plots but also ones that look different from each other.

ConvViz allows one to confirm relations between filters, redundancy as well as understand the different aspects each filter is looking at or the reason why a filter is dead. Typically, one example is shown in Figure 24 with the local visualizations of two filters that looked close once PCA was performed on their weights. It is possible to see that the filters are very similar and could be qualified as redundant: their feature visualization is similar as both seem to recognize blobs, they focus on the same areas of the input image and their maximum activations are close.

The overall visualization toolbox of ConvViz, made up of local and global visualizations as well as interactivity to go from one to the other, is developed with PyQT, using the libraries of OpenCV for image visualization and Matplotlib for all the different plots.

## 9. Results: Pre-processing

Ideally, ConvViz would be real-time, extracting all the information and displaying it directly. However, there is an abundance of data to visualize and it is time-consuming to extract all the required information. The workflow has, therefore, been separated into two steps: first the extraction of information and, then, the visualization itself.

The information saved during the first step is summarized in Table 2.

The first row of Table 2 represents information on the network itself, with the effective filter size calculated according to the formulae from Appendix C. The information from the second row is extracted after performing a forward pass and the data from the third row comes from backward

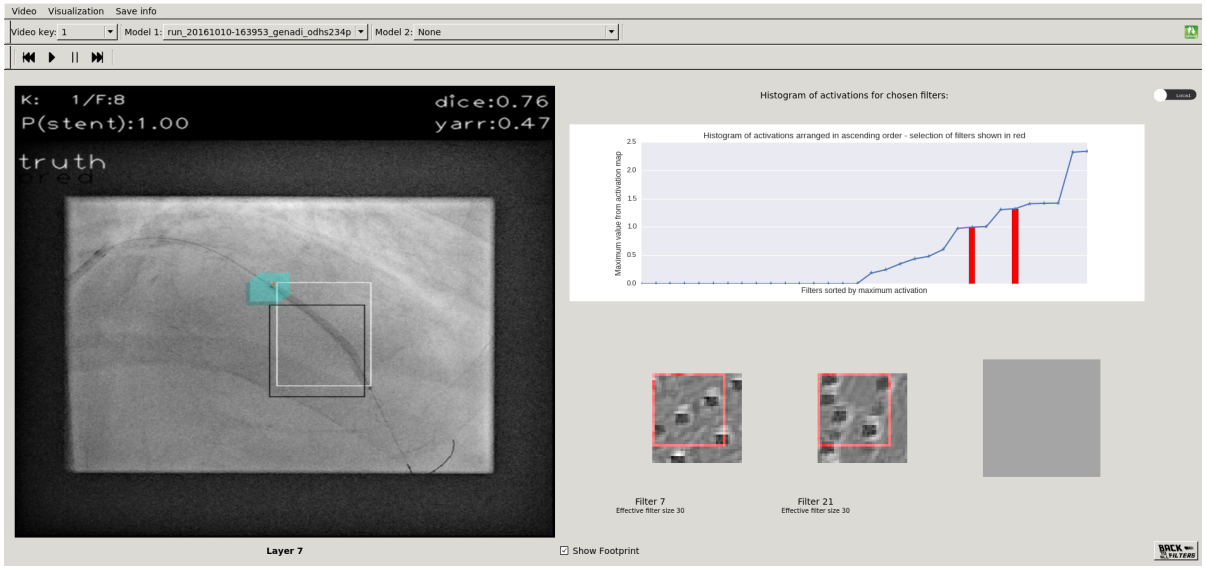



**Figure 24.** Local visualization of two filters chosen through global visualization

**Table 2.** Summary of Information Extracted

| | |
|---|---|
| Effective filter size | Weights and bias |
| Localization prediction | Activation maps |
| Input optimization | Backward pass |
| t-SNE on activation maps for each input | |

passes or from backpropagation to optimize the input. Both of these pieces of information are extracted for all filters of all convolutional layers. Moreover, the backward pass using the DeconvNet is calculated for all inputs. As the user would not analyze the local visualization for all of the inputs from the validation set, it is of importance to first select inputs to visualize. This selection can be done randomly, hoping that the input space will be partially spanned by these inputs, by manual selection in case the user knows the dataset at hand well, or by looking at the dimensionality reduction of the activation maps from the first layer for all the different inputs, as in Section 7.3. By targeting the first layer, only low-level features will be considered but the clustering will be less dependent on the network. This will allow the user to choose an input image per cluster in order to, afterwards, analyze the local visualizations on all types of inputs. Finally, the last line of Table 2 represents the positions and colors of the points in the 2D space after dimensionality reduction to see the clustering between inputs. All of these pieces of information are saved to save time but also to avoid the need of using libraries related to Deep Learning such as Keras or TensorFlow during visualization.

While the first type of data, i.e., the first row of Table 2, is only extracted once, the prediction and activation maps are needed for all the selected inputs. Seeing that the extraction does not need to be done in real-time, this is not an issue even though it adds up to a consequent amount of time. For example, for the object localization task, the backward passes, to get the areas that are mostly activating, can take up to 10 hours for all the filters from all convolutional layers for an entire video of, on average, 35 frames. This explains the reason why it is significant to choose specific inputs from the validation set in order to visualize them and not perform the extraction on all the inputs from the validation set. The dimensionality reduction to see clusters on the inputs was performed on GPUs with the implementation of the multicore t-SNE [22]. The calculation still takes up to half an hour for the last convolutional layers because of the concatenation of all the activation maps. The summary of the time required for each data extraction and the number of extractions is summarized in Table 3. As some operations, such as feature visualization via input optimization, take more or less time according to the layer considered, the average time is provided. While the extraction of all pieces of information takes a lot of time, one needs to keep in mind that all of these pieces of information can be extracted in parallel and that this is the first pre-processing step.

**Table 3.** Time required for data extraction for the visualization, per network, using CPU

| Data | Time per iteration | Number of Iterations (e.g. for object localization) |
|---|---|---|
| Effective filter size | 10ms | Once (1) |
| Weights and bias | 12ms | Once (1) |
| Localization prediction | 0.5s per frame | Size validation set * Number of frames per input (18,520) |
| Activation maps | 0.3s per frame | Size validation set * Number of frames per input (18,520) |
| Input optimization | 1min per filter | Number of filters (288) |
| Backward pass | 15min per frame | Size validation set * Number of frames per input (18,520) |
| t-SNE on inputs | 25min | Number of layers (9) |

## 10. Results: Visualization

Once all of the data required is extracted, the second phase of the workflow can start: the user can launch ConvViz. In order to see the different visualizations, the user only needs to choose the set of inputs desired (either a folder or a HDF5 file) and the network considered, by selecting the folder where the pre-processing information was saved.

Both types of visualizations described in Sections 6 and 7 are gathered to form ConvViz. Figure 25 provides a screenshot of the local visualizations possible with ConvViz, accomplishing what was desired from the design on Figure 10, matching the circled areas with the same letter. More precisely, one single layer (B1) or multiple (B2) layers can be selected and the square filling for the backward pass can be performed, or not (G). As the identification of each filter is important, it is possible to click on a filter from the feature visualization in order to see what is its region of interest on the input image. The other filters will not be shown and only that filter will be backward passed to the input space. One can, therefore, better understand what is going on with a specific filter and which shape it has learnt.

According to this local visualization, for the specific input video considered throughout this report, it is possible to see that the top filters localize both of the markers early in the network but that the whole object cannot be captured by a single filter. Actually the two top filters are focusing on at least one of the markers, for most of the frames already in the fifth layer, as Figure 17a shows. The wire is also taken into consideration by the top filters early in the network, as in Figure 25A with the filters dispayed in blue and yellow. This information is essential because it shows that the network is considering the same regions of interest as scientists do when they need to localize markers. This helps understand why the network is that accurate. From the fifth to

the last convolutional layer, the top filters backward-passed are usually located at the regions of interest according to doctors (markers and wire). Ending the network at the fifth layer would not be sufficient because, for some inputs, the top filters localize only one of the markers, but, from the sixth layer, accurate localization is seen. This provides a first hypothesis that the network could be shortened in order to solve that task. This conclusion is reinforced by feature visualization: the filters from the sixth layer onwards learn features that look like blobs, as Figure 13 depicts. Seeing that markers can be seen as blobs, it confirms the fact that the filters are trying to detect markers and the network could be reduced in terms of number of layers. Actually, accurate localization is available from the sixth layer, making the following layers irrelevant. On the contrary, Figure 18 shows that the whole object cannot be localized by a single filter. The deeper inside the network, the larger the effective filter size, and the filters from the last layer, displayed in green on Figure 18, cannot capture the whole object. Another contradictory hypothesis could be made: the number of layers should be increased. This would allow the network to understand the structure of the object and not just separately recognize each marker. The size of the object to localize obviously depends from one input to another and since the size of the network is not dependent on the input, the average size should be considered in order to see how many layers should be added.

After first seeing the histogram of activations from local visualization, as seen in Figure 26, it seems that, in the first layers, either many filters are very specific and not being activated for this input, or that they are not being used: they can be considered as dead filters. The later possibility is confirmed by the visualization of the cumulated histogram,

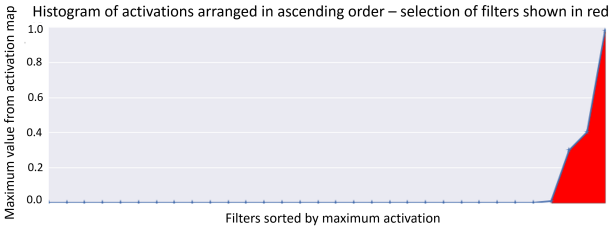



**Figure 26.** Histogram of activations of the third convolutional layer for the input of Fig. 9

as seen in Figure 19: most of the filters can be considered as "dead" because their energy is really low, as it can be concluded from Figure 20. ConvViz fulfills Requirement **3** in this case, but it needs to be confirmed that low energy filters are actually dead and not just contributing less compared to the others. The summary of the number of used filters per layer, according to the analysis of histograms, is provided in Table 4. Therefore, it looks like there are on average over 60% of the filters that are not being used, whatever the input. This leads to the conclusion that a network with less filters could be designed, allowing faster computations since in that configuration the number of computations performed would be decreased by 95%, allowing this network to run in real-time.

In order to make sure that those filters are not being used and can be discarded, a new network was designed, setting the weights of the unused filters to zero. Not only did the accuracy measure not decrease after this removal, but it actually increased. Even though the increase was very small (only 0.02%), it still emphasizes the fact that most filters are unused. A potential reason to this increase is that, for each

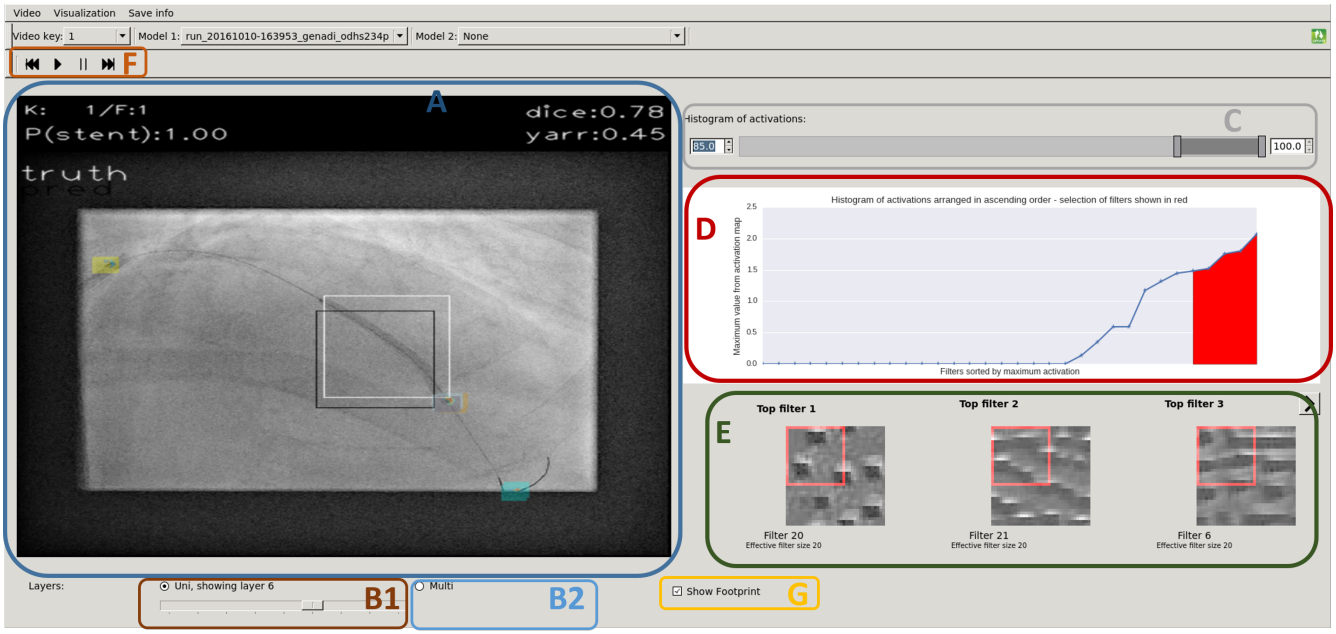



**Figure 25.** Screenshot of all the local visualizations from ConvViz

convolutional layer, the convolution is performed such that the output has the same size as the input. The necessary padding before performing the convolution adds noise on the borders and is the reason of this accuracy decrease when dead filters are considered.

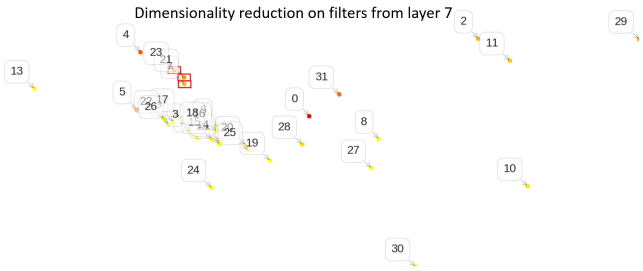



**Figure 27.** Dimensionality reduction on all filters from the eighth layer, with a cluster of filters circled

By analyzing the dimensionality reduction on the filters' weights, as depicted in Figure 21 for the sixth layer, few useless filters, i.e., red ones, are redundant because no clustering can be seen. However, there are some redundant filters on the last layers, as the circled filters in Figure 27 shows. The hypothesis of redundancy from the dimensionality reduction on filters' weights can be confirmed by analyzing specifically those filters. This has been done for two filters from the seventh layer, as shown on Figure 24: the hypothesis becomes stronger because the filters have learnt the same shape and they focus on the same areas, for a given input. Hence, ConvViz allows visualization of correlations between filters (Requirement **4**) and linking between low-level features and high-level features (Requirement **5**).

**Table 4.** Number of filters used in the network

| Layer | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| Filters used | 4 | 5 | 3 | 7 | 9 | 12 | 16 | 24 | 29 |
| Percentage | 13% | 16% | 9% | 22% | 28% | 38% | 50% | 75% | 91% |

Finally, the exploration of the data (Requirement **6**) is also offered by the dimensionality reduction of activation maps of all inputs for a given layer, as in Figure 23. These plots are, however, difficult to analyze with this use case because all the inputs look more or less similar. Hence, no additional information is provided by the selection of some points from the scatterplot.

Therefore, for the use case at hand, different hypotheses have been gathered according to the visualizations:

- The number of convolutional layers should be increased, such that one filter in the last convolutional layer should be able to look at the whole object to localize and not just at one of its markers.
- The number of convolutional layers should be decreased since accurate localization can already be gathered from layer 6. This will enable deployment of the network for real-time applications.
- The number of filters per layer should be decreased for better accuracy and, again, for real-time applications.

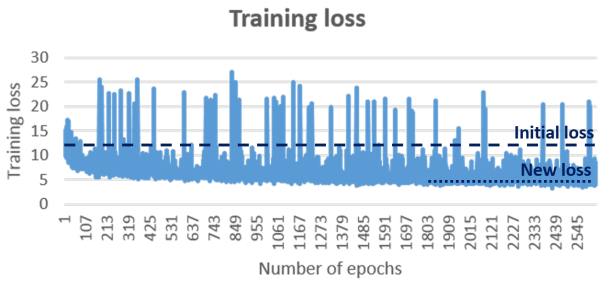



**Figure 28.** Plot of the evolution of the training loss during training of the new shallower network

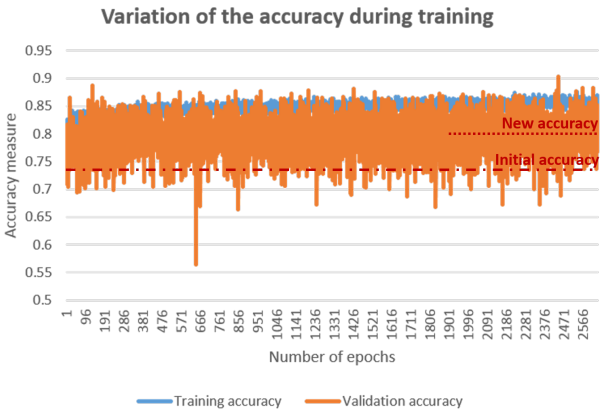



**Figure 29.** Plot of the evolution of training and validation accuracy measures during training of the new shallower network

The last conclusion has been confirmed by training a network that was satisfying this new constraint. In this way, we initialized the weights of the kept filters to the weights of the trained network studied for the visualization while removing the undesired filters. The learning rate was set to $10^{-4}$ and we used the RMSprop optimizer, which divides the learning rate for a weight by a running average of the magnitudes of recent gradients for that weight. The results from that experiment are depicted in Figure 28 for the loss and 29 for the accuracy measure. The decrease of the loss value as well as the increase in the accuracy value during training are common evolutions. However, seeing that we were considering a network that already performed well (an accuracy measure of 0.748), those changes were harder to get. It can, nevertheless, be seen that the training loss is decreasing compared to the one of the previous network, even though it starts from a very low value of 15 (networks initialized with random weights have initial loss values of 2,000 - 3,000) and the accuracy measure rises to 0.801 for the validation set. This means that by following one of the

conclusions made from the visualization, it is possible to decrease the computational complexity by 95% while increasing the accuracy by over 5%.

## 11. Generalization: Application to the IMAGENET data

As the development of ConvViz has been done after studying the network for the object localization task, its generalizability to other networks and use cases needed to be studied. For object visualization, the inputs were grey-valued videos, the task was localization and the network considered was not too deep. In order to analyze a different situation, the well-known problem of IMAGENET Challenge was studied. For a change, the task of classification was considered with the VGG16 network [2] as it had achieved the best performance at the time and has, since then, become a reference for DNNs. Hence, the inputs (coloured images), the output (a class), the task (classification) as well as the aim of the visualization (understand what the network is doing, instead of refining it) were different from the previous study.

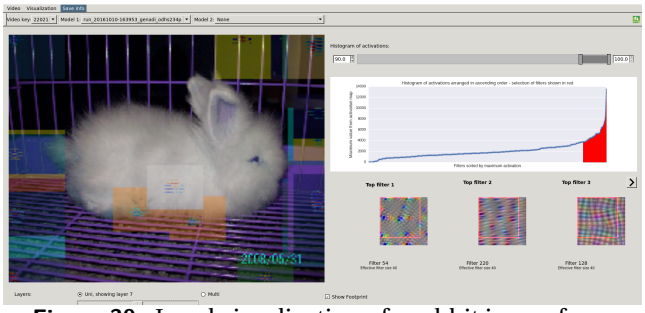


**Figure 30.** Local visualization of a rabbit image from IMAGENET with a VGG16 network

Figure 30 represents the local visualization of the $7^{th}$ convolutional layer, i.e., the end of the third block from the network, using a rabbit image as the input. Typically, thanks to ConvViz, one can confirm previously made hypotheses such as the fact that the filters from the first layer detect lines and from the second layer distinguish colours in the case of coloured images. When looking at Figure 32a, it looks like most of the filters from layer 2 are identical. When selecting them in order to have the local visualization of this cluster of filters, the input optimization shows that they are actually different since they focus on distinct colours, as seen in 32b with six filters coming from that central cluster. It is possible to analyze the rest of the filters from the dimensionality reduction and learn that they target oblique lines, and that the ones clustered together recognize the lines with the same angles. Hence, redundancy can, also in this case, be spotted to know how wide the network should be and which filters to keep if one aims at performing fine-tuning.

Finally, it is also possible to make new hypotheses from these visualizations: for instance, many filters seem redundant in the first layers, as in Figure 32a, whereas it does not

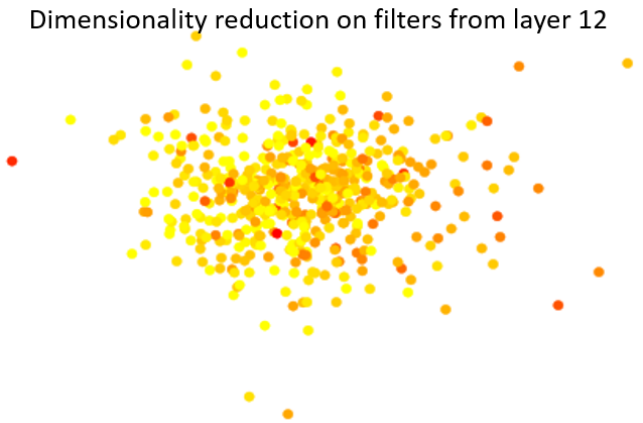



**Figure 31.** Dimensionality reduction on the 512 filters from the twelfth convolutional layer

seem to be the case for later ones which are more equally spread out, Figure 31. This means that even though developers usually increase the number of filters per layer with the depth of the network, it should be even more the case than how the networks are currently designed.

Moreover, thanks to ConvViz, one can better understand the dataset at hand and how the network is seeing it in order to take its decisions, via the dimensionality reduction performed on the activation maps of all the different images from the validation set, as explained in Section 7.3. In the case considered, we get data that is spread around for the first layers, not depicting clear clusters, as it can be seen on Figures 34a and 34c. This can be interpreted as the network not having clustered the different images yet, and, therefore, showing the importance of the following layers. However, it could also come from the fact that the data is so high dimensional and sparse that no clusters can be seen. With a willingness to study those visualizations and to understand what the network is focusing on at those layers, we selected points that seemed to represent clusters according to their localization after dimensionality reduction as well as according to their colour, i.e., according to their largest dimension before applying the t-SNE. The selection is represented by a red rectangle on Figure 34. Figures 34b and 34d show some inputs that are part of those clusters. In fact, for each cluster, six inputs are selected randomly and displayed. It is, therefore, possible to see that the first layer seems to be looking at very simple features seeing that all the images, on Figure 34b, have for dominant colors brown and white, with a specific full-of-light-white representing snow on 2 out of the 6 images. In the fifth layer, as seen on Figure 34d, the inputs which are clustered together are more similar as they represent one or two animals, most of them of dark coloured with a plain background. This clustering does not suffice in taking a decision as specific as IMAGENET needs to be, seeing that the species of an animal needs to be output for example, but some clustering is already visible at this point.

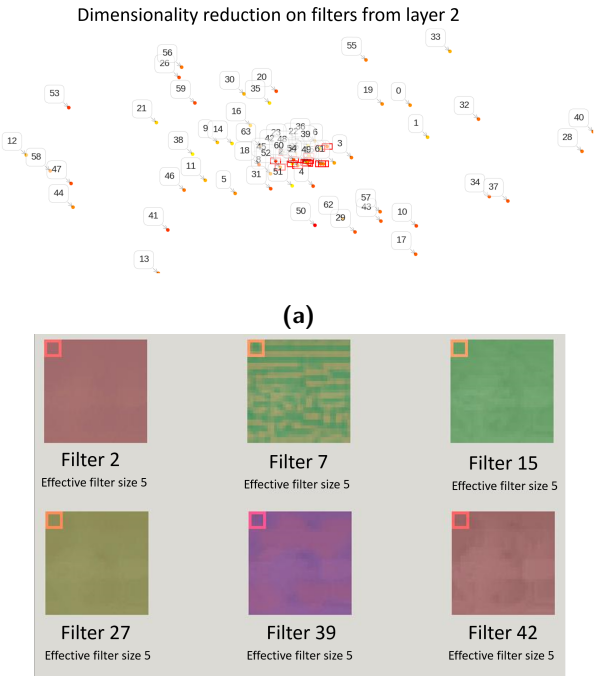



**Figure 32.** (a) Dimensionality reduction on the filters from the $2^{nd}$ convolutional layer, with selected filters surrounded by a red box. (b) Input optimization of the filters selected in (a).

Even though most of the visualization tools in ConvViz are well adapted to this use case, there is an issue with the global visualizations when the network is too wide, typically when there are over 200 filters per layer. This is shown in Figure 33 with the histogram of activations for the fifth convolutional layer. In that case, there are 256 filters, making the histogram illegible even though it is not even the widest part of the network.

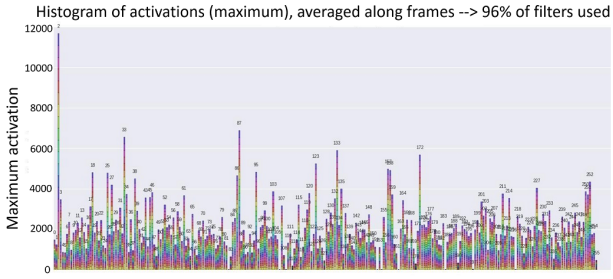



**Figure 33.** Histogram of activations for a very wide network

To conclude, ConvViz generalizes well to other networks and use cases, being useful for understanding and refining networks but still has some limits such as scalability.

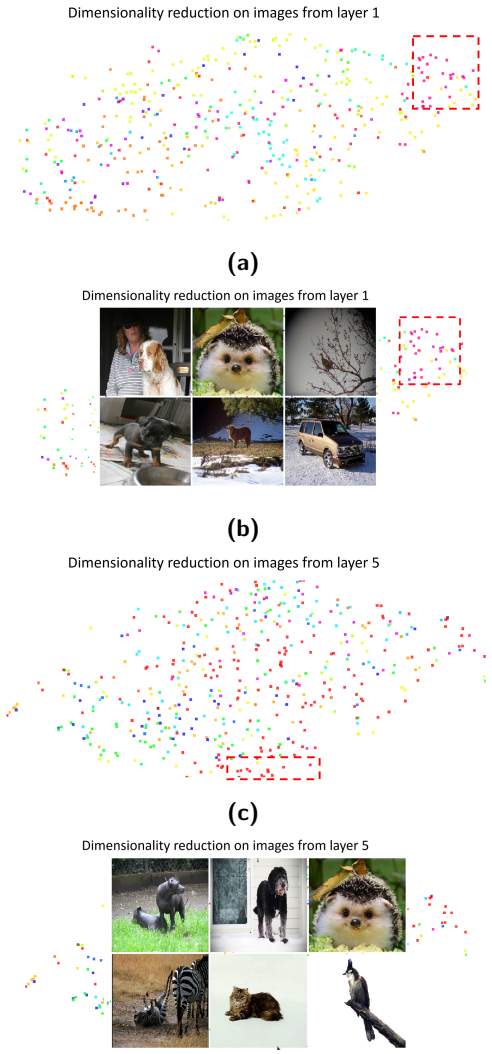



**Figure 34.** Dimensionality reduction on the activation maps of all inputs for the $1^{st}$ (a) and the $5^{th}$ (c) convolutional layers. Visualization of some inputs selected (circled by red boxes) (b) for the $1^{st}$ layer and (d) for the $5^{th}$ layer.

## 12. Conclusion and Future Work

In this paper, we described ConvViz, a framework for visualizing convolutional neural networks in order to better understand and refine them. In fact, in order to avoid iterating the time-consuming trial-and-error method used to improve a given network as well as lighten that black box that is a deep convolutional neural network, visualization can be used. We envisioned a framework, composed of two steps: pre-processing and extraction of data, followed by the visualization itself, composed of both global and local state-of-the-art visualizations. We implemented those visualizations and analyzed the results on two different use cases in order to evaluate the usefulness of this toolbox and its generalizability.

Thanks to ConvViz, one can make hypotheses concerning the number of layers a network should have. This can be concluded according to the objects the top filters are focusing on, by seeing when the results are already good enough. Moreover, the ability to see redundant and dead filters, or on the contrary, the differences between filters, makes it possible to understand their role and if their number is correct, should be increased or decreased. Therefore, in order to get a well-performing and robust network, it could be possible to train a very deep network and then prune it thanks to the visualizations. It is also possible to better understand the network itself by visualizing selected filters and their regions of interest. One can also compare two different networks using ConvViz, looking at their regions of interest and comparing their outputs. Moreover, thanks to ConvViz, one can better understand the validation set at hand and how the network is seeing the images for all convolutional layers.

The case studies have given promising results, with, for example, the possibility to decrease the computational complexity by 95% while increasing the accuracy by 5% for the object localization task with the network at hand. Those improvements are all the more important that one of the main issues faced by neural networks is their time complexity and, therefore, difficulty to deploy for real-time problems. Moreover, the generalization of ConvViz has been tested and accepted by visualizing a differently designed network, with a distinct task and inputs. However, the evaluation of the visualizations' effectiveness needs to be extended. Finally, future work and research still needs to be done for ConvViz, as it could be improved as follows:

**Maximum filter and neurons** As explained in Section 6.3, the top filters for each layer are chosen according to their activation map: for each input, the top filters are the ones with the highest maxima on their activation map. This method could be refined: one could consider a filter to be a top one if it is of importance for the final decision. This latest approach has been followed by R. Selvaraju et al. for Class Activation Mapping, leading to grad-CAM [23]. This method provides the heat map of how important each part of the input is for the network's decision, using gradient-weighted activation maps. It would be of interest to study this approach for ConvViz in order to see if the top filters would change as those new ones would be more coherent with the definition of top filters provided in 6.3. Moreover, an improvement would be to let the user choose the threshold of the activation value for the selection of neurons to visualize. At the moment, the code used to perform the backward-pass is time-consuming, so it needs to be made faster in order to be able to have different values of threshold.

**Attention maps** Even if the backward pass of the maxima with DeconvNet does tell the user where the filters are focusing, it does not give an overall view on the input pixels' importance as attention maps do [16]. In this way, it would be interesting to visualize the attention map of each filter selected by the user.

**Variation of perplexity** The quality of dimensionality reduction on high-dimensional datasets is hard to assess as information is lost; typically t-SNE preserves clusters and neighborhoods but does not preserve distances. Moreover, the perplexity influences highly the results, as explained in Section 7.3. For this reason, an improvement of this toolbox would be to allow the user to change the value of the perplexity and to see the alterations on the 2D visualization dynamically.

**Time complexity** During the first step of the framework, important information for the visualizations is extracted. A lot of work still needs to be put in this step seeing how time-consuming it is, as it is summarized in Table 3.

**Visualization while training** One of the aims of ConvViz is to avoid iterating trial-and-error in order to save time. Time saving should be even more at the heart of ConvViz by allowing visualization during training, as TensorBoard does [24]. Therefore, it would be possible to make changes to the network architecture even before the training has finished.

**More generalization** Finally, these visualizations are only available for convolutional layers and are intra-layer. Even though the extension to fully-connected layers is straightforward for several tools, networks that are not convolutional cannot be visualized with ConvViz. This is not a core concern, as most of the networks used nowadays are convolutional. Moreover, only networks considering images or videos as inputs can be visualized, which is not a restriction to Philips since the tasks at hand are dealing with medical imaging. However, one important constraint is the scalability of this toolbox. While it performs well for normal to deep networks, it is not the case for very deep or wide networks, especially concerning the global visualizations. A solution would be to perform clustering on the layers for deep networks (over 20 layers) or on the filters for wide ones (over 200 filters per layer) and then select one per cluster, as done by CNNVis [8]. Finally, the fact that all the visualizations are intra-layer is a restriction to the possibility to have a global view on the network. Therefore, visualizations across layers should be added, such as comparing the filters from different layers.

In conclusion, the present master thesis allows better understanding and refining of deep convolutional neural networks by putting together state-of-the-art visualization methods and allow interactions between them. The toolbox ConvViz has given encouraging results on the case of object localization and has generalized well to another use case and other networks. Philips Healthcare is interested in extending the research in this direction for different use cases.

## Acknowledgments

I would first like to deeply thank Binyam who was always open for questions whenever I ran into a trouble and who gave me valuable comments. I know Marta will grow up with great advice. I also would like to express my gratitude to my supervisor Anna as well as Mark and Javier for the time they dedicated to me. Finally, I would like to thank my loved ones who remotely supported me.

## References

[1] Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5188–5196, 2015.

[2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[3] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1520–1528, 2015.

[4] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision*, pages 818–833. Springer, 2014.

[5] Dumitru Erhan, Aaron Courville, and Yoshua Bengio. Understanding representations learned in deep architectures. *University of Montreal / DIRO*, Technical Report 1355, 2010.

[6] Devinder Kumar and Vlado Menkovski. Understanding anatomy classification through visualization. *arXiv preprint arXiv:1611.06284*, 2016.

[7] Paulo E Rauber, Samuel G Fadel, Alexandre X Falcao, and Alexandru C Telea. Visualizing the hidden activity of artificial neural networks. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):101–110, 2017.

[8] Mengchen Liu, Jiaxin Shi, Zhen Li, Chongxuan Li, Jun Zhu, and Shixia Liu. Towards better analysis of deep convolutional neural networks. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):91–100, 2017.

[9] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[10] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[11] Andrej Karpathy. Lecture notes in cs231n convolutional neural networks for visual recognition. Stanford University / Computer Science. http://cs231n.github.io/understanding-cnn/, 2017.

[12] Honglak Lee, Chaitanya Ekanadham, and Andrew Y Ng. Sparse deep belief net model for visual area v2. In *Advances in neural information processing systems*, pages 873–880, 2008.

[13] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

[14] Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent. Visualizing higher-layer features of a deep network. *University of Montreal*, Technical Report 1341:3, 2009.

[15] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *Proc. International Conference on Learning Representations Workshop*, arXiv preprint arXiv:1603.02518, 2013.

[16] Luisa M Zintgraf, Taco S Cohen, and Max Welling. A new method to visualize deep neural networks. *arXiv preprint arXiv:1603.02518*, 2016.

[17] Dumitru Erhan, Pierre-Antoine Manzagol, Yoshua Bengio, Samy Bengio, and Pascal Vincent. The difficulty of training deep architectures and the effect of unsupervised pre-training. In *AISTATS*, volume 5, pages 153–160, 2009.

[18] Sunghyo Chung, Cheonbok Park, Sangho Suh, Kyeongpil Kang, Jaegul Choo, and Bum Chul Kwon. Revacnn: Steering convolutional neural network via real-time visual analytics. *Workshop Conference on Neural Information Processing Systems*, pages 30–36, 2016.

[19] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, arXiv preprint arXiv:1409.1556, 2014.

[20] Raghavendra Kotikalapudi and contributors. keras-vis. https://github.com/raghakot/keras-vis, 2017.

[21] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[22] Dmitry Ulyanov. Muticore-tsne. https://github.com/DmitryUlyanov/Muticore-TSNE, 2016.

[23] Ramprasaath R. Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, and Dhruv Batra. Grad-cam: Why did you say that? visual explanations from deep networks via gradient-based localization. *CoRR*, arXiv preprint arXiv:1412.6806, 2016.

[24] Joseph J. Allaire, Dirk Eddelbuettel, Nick Golding, and Yuan Tang. *Tensorflow: R Interface to TensorFlow*, 2016. Software available from tensorflow.org.

[25] Daniel Bruckner. Ml-o-scope: a diagnostic visualization system for deep machine learning pipelines. *DTIC Document*, Technical report, 2014.

# Appendices

## A Related Work - Global Visualization

As introduced in Section 3 with a quick overview of methods for visualization, research in Deep Learning is focusing on the visualization of neural networks. A more complete review of the state-of-the-art methods for global visualization is provided in this section.

Because the number of parts making up a network is enormous, too many local visualizations should be analyzed, requiring global visualizations for having an indication of the interactions between the different parts, to understand and refine the network. A first approach that has been used extensively to improve neural networks is to plot the quantitative information such as the error rate, the loss or the evaluation of the accuracy during training. This has been used by Erhan et al. [17] in order to see the influence of pre-training on the performance of the network, by ML-o-scope [25] with an improved confusion matrix or by TensorBoard [24]. In fact, the latest allows the visualization of the distribution of activations or of weights coming off a particular layer, as shown on Figure 35. From these visualizations, it is possible to visualize redundancy in the layers. For example, if the activation function is the ReLU and almost all the weights of a layer are positive, as in layer 4 of Figure 35, redundancy can be spotted. In fact, the output of the convolution will be positive and the layer will be linear seeing that the activation function will not change the output of the convolution. Hence, this provides information on how to modify the network: with the previous example, the redundant layer can be removed. These methods, considering quantitative information, allow the user to compare networks or visualize overfitting for example. However, they do not provide qualitative information.

### Representation Visualization

Dimensionality reduction, such as t-Distributed Stochastic Neighbour Embedding (t-SNE) and MultiDimensionality Scaling (MDS), maps high dimensional datasets into lower dimensions (usually two or three) to be able to visualize those datasets. It is crucial for CNNs to perform dimensionality reduction since there are up to thousands of features in each network.

Rauber et al. [7] visualizes the relationships between learned representations of input images by applying t-SNE on the last hidden layer of the network. From this visualization, it is possible to get qualitative feedback on clusters. The aspect of clusters provides hints concerning which steps should be taken in order to improve the network. Moreover, outliers can be spotted from this visualization, allowing the user to remove them from the dataset or to pre-process them.

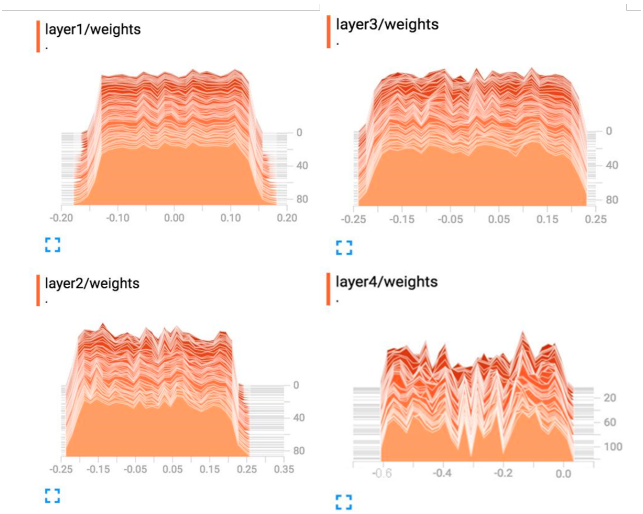



**Figure 35.** Example of distribution of weights, visualized with TensorBoard

It is also possible to have a dynamic visualization [7]: for a specific layer and inputs, the evolution "inter-epoch" with t-SNE and 2D trail bundling can be visualized, representing the state of the layer after each epoch, i.e., after each forward and backward pass for all training inputs, during training. This shows the evolution during training and when convergence has been reached. One can also visualize the evolution "inter-layer", with each point being an input image, as shown on Figure 8. One can provide a hypothesis as to how many layers are sufficient and when there is overtraining caused by a network that is too deep: when clusters start to get well-separated on the different captions of Figure 8. However, subjectivity comes into play: one can see the gathering of all points from one class to the same area, as seen with class 2 (the green one) on Figure 8, as overtraining or, on the contrary, as a good classification and generalization. Interpretation is a crucial problem when visualization is concerned.

Another dimensionality reduction-based visualization of neural networks has been recently used with ReVACNN, Real-time Visual Analytic system for CNNs [18]. It offers a 2D embedding view, for a specific layer, where many parameters can be visualized: vectorized filter coefficients, filter gradients, activation maps as well as activation gradient maps. By analyzing clustering, redundant filters can be outlined and, therefore, low accuracy.

### A.2.2 Network as a Graph

CNNs have often been pictured as directed acyclic graphs, because of their structure with consecutive layers made up of filters. This relation between CNNs and graphs has been analyzed by Chung et al. [18] and has led to a network view in the ReVACNN toolbox. With each node representing the activation map of one filter and each edge a connection between filters of consecutive layers, the whole network can be visualized. To avoid visual clutter, only some links

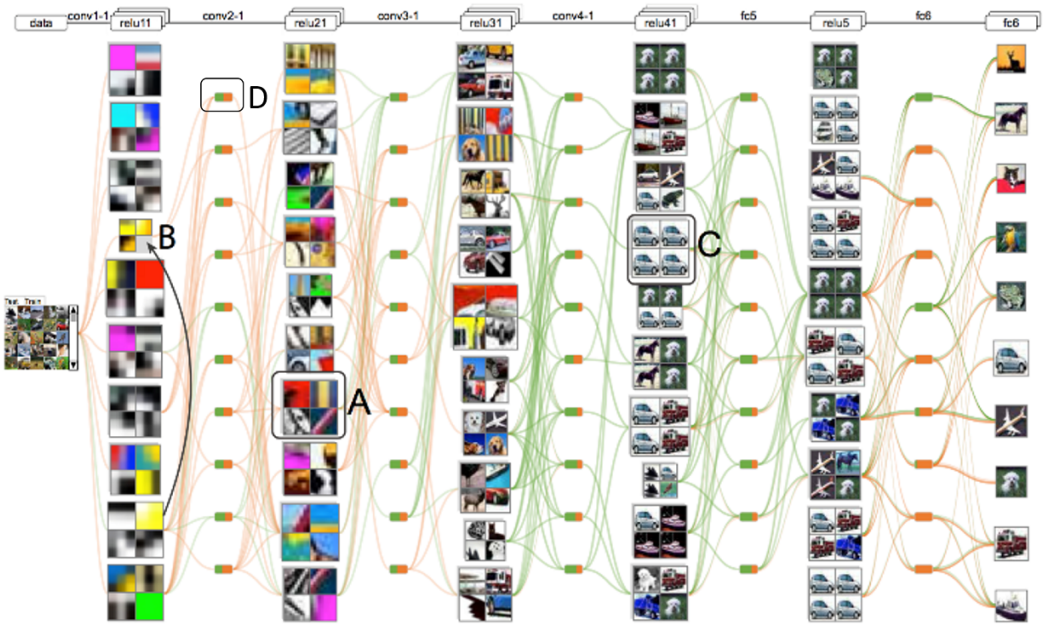



**Figure 36.** Example of graph visualization with CNNVis [8]

between layers are shown. The tool is interactive, allowing the user to add or remove filters of hidden layers during training and see the direct effect on the network. In the same way, it is possible to freeze some nodes and layers when convergence seems to be reached in order to reduce the training time.

Liu et al. [8] also studied this relationship and came up with CNNVis which offers a visualization of the overall training process through a graph, as seen on Figure 36. While it requires the network to be trained, it is more scalable than the previous tool as only some layers and neurons are showed. For layers, clustering, using k-means, is first performed and then only one layer per cluster is kept. In the same way, clusters of filters are found and, for each of them, Inversion Visualization, as in 36A, described in Section 3.1.1 and Feature Visualization, as in 36C, described in Section 3.1.2, are depicted. A biclustering-based edge bundling is performed, i.e., simultaneous clustering of rows and columns of edges, displaying the proportion of negative and positive edges, showed in 36D. Finally, interaction is possible by modifying filters clustering, as in 36B. Hence, CNNVis allows one to come up with the conclusions found using local visualizations as well as new conclusions. For example, as the distribution of weights is shown through biclustering of edge bundling, many positive edges will represent redundancy, following the same way of thinking as with TensorBoard previously. Also, if redundant filters are visualized, one will understand that there must have been overfitting.

## B Feature Visualization

As explained in Section 6.2, feature visualization is one of the visualizations that help the user understand which shape a filter is being discriminative to. As explained in that section, the size of the input to optimize is the effective filter size in case it is bigger than a fixed threshold; otherwise it is a square of the size of that threshold. For ConvViz, a threshold of 35 has been used. What is commonly being done is to start from a random image and then optimize it. While this method works well for first layers, as it can be seen on Figure 37, it does not work for deeper layers since the filters are becoming more specific and it is unlikely to find their learned shapes in random images. Hence, the values of the activations are either zeroes or close to zero, as it can be seen on Figure 14 or on Figure 38 for the use case of the object localization. Therefore, no optimization can be performed and the input image is not modified.

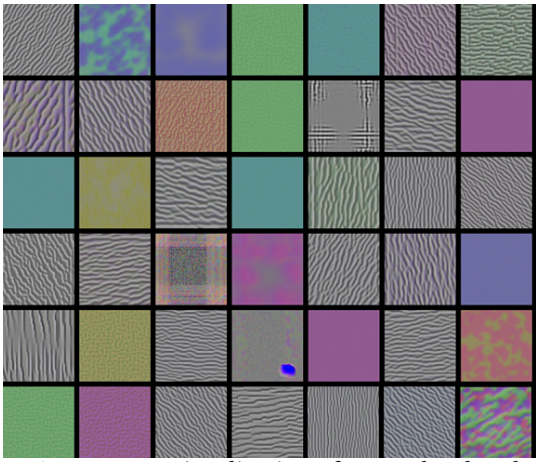


**Figure 37.** Feature visualization of 42 randomly selected filters from the second layer of the network trained with IMAGENET, starting from random image

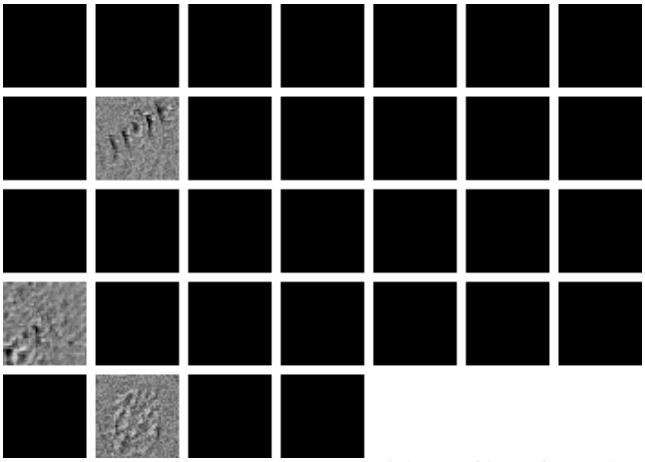


**Figure 38.** Feature visualization of the 32 filters from the last layer of the network trained with object localization, starting from random image

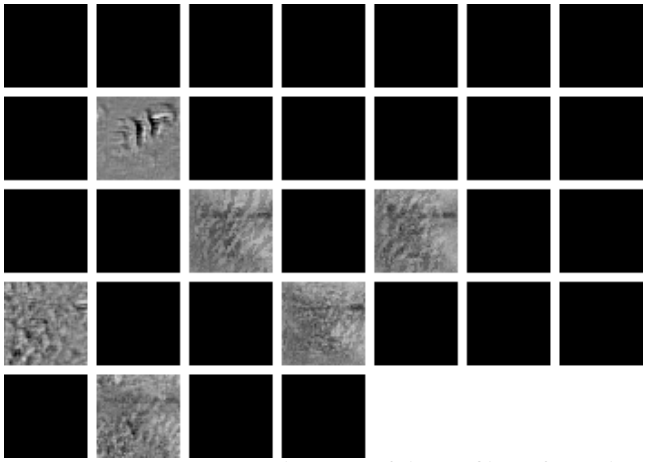


**Figure 39.** Feature visualization of the 32 filters from the last layer of the network trained with object localization, starting from an input of the validation set

A first hypothesis was the fact that the input image needs to be of the same type as the other available images. Typically, the same type of histogram or value of entropy should be chosen for the input image. In order to verify that hypothesis, the filters were optimized but starting from one image of the validation set. The results can be found in Figure 39 for the object localization use case and in Figure 40 for the IMAGENET use case. For both figures, the input has not been subtracted to the optimized image but a dark square is visible if no optimization was achieved. We can see the dependency on the input image on Figure 39 because of the same luminosity overall the image for each optimization. In the second use case, it is obvious that the initial input is an image of a dog. The recognition of a specific object is not as straightforward on the first use case, even on the last convolutional layer, because features are only looking at a small part of the image (46*46 pixels) whereas in the second use case, the last convolutional layer looks at parts of the image of size 196*196 pixels, which is nearly all the image seeing that the input shape is of 224*224 pixels.

It is anyway obvious that starting from one of the inputs is better than starting from a random image seeing that we are able to find solutions in more images but this visualization is highly dependent on the input being optimized. The second approach was, therefore, to take several images, take their mean and then perform the optimization. While the result would still be dependent on the input, it would not be as obvious and another step was taken: first remove the mean input image to that optimization, in order to gather only the changes that had taken place and then perform the optimization to that new image. In this way, the dependency on the input image would be reduced as much as possible. For the object localization use case, Figure 41, as well as the IMAGENET use case, Figure 42, more filters are visualized but there are still many filters that are not. In order to deal with that issue, another approach was taken,

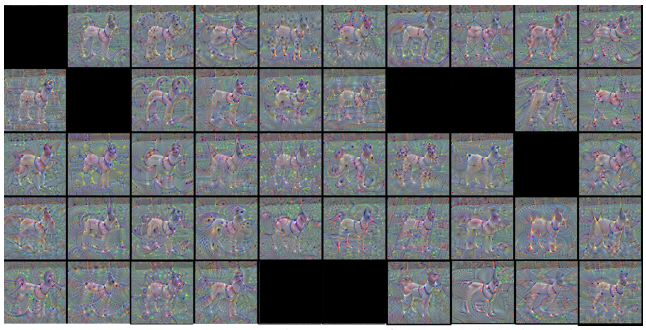


**Figure 40.** Feature visualization of 50 randomly selected filters from the last layer of the network trained with IMAGENET, starting from dog image

focusing on finding the right inputs for a specific filter: for each filter, the optimization was tried in the same way as previously. If the backpropagation would not result in a decreasing loss, meaning that the learned feature is not part of the input image, the input needs to be changed and new images are selected at random. This leads in the possibility of visualizing more input optimizations. The results for the IMAGENET use case are presented in Figure 43: all the filters are being optimized and, by looking separately at each filter, a clear feature can be detected. While this method is very efficient for this use case, seeing that the inputs are being optimized for nearly all the filters, it is not the case for the object localization use case where only half of the filters from the last convolutional layer are being optimized. The results are better than the previous methods but one needs to keep in mind that, an input image that has not been optimized with respect to a filter, does not mean that the filter is necessary dead. In fact, it could be that its feature learned is complex and not present in most of the images.

## C Computation of the effective filter size

**Assumptions** For this section, it will be considered that the spatial extent of all the filters are identical, of $F$ (usually 3)

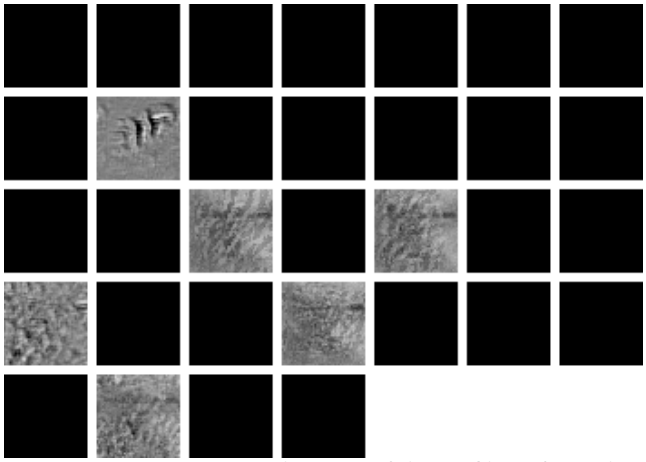


**Figure 41.** Feature visualization of the 32 filters from the last layer of the network trained with object localization, starting from mean image

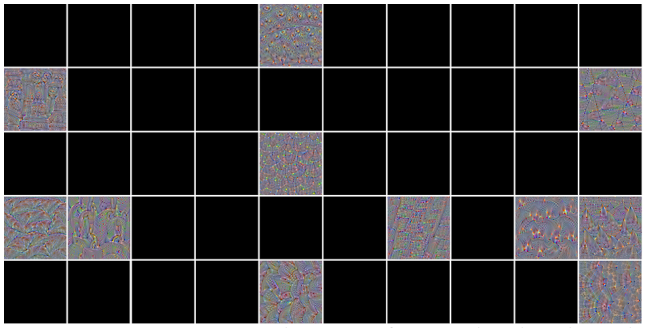


**Figure 42.** Feature visualization of 50 randomly selected filters from the last layer of the network trained with IMAGENET, starting from mean image

and that the stride used for convolutional layers are of 1 and of $S$ (usually 2) for maxpooling layers. The generalization can be done following the same way of reasoning but this hypothesis is made for simplicity in order to have one single effective filter size per layer and simpler functions but also because it is usually the case in the networks' architectures.

**Definition** The most common way of calculating the effective filter size of a filter from a specific layer is by going down in the previous layers. Typically, the effective filter size can be initialized to 1 just after the targeted layer and then update this value according to the layers you are going through until the input layer. When going back through a convolutional layer, the effective filter size (eSize) can be updated to $eSize = (eSize - 1) * s + F$ with s the stride, assumed to be equal to 1. When going back through a maxpooling layer, it should be updated to $eSize = eSize * S$.

By calling $u_n$ the value of the sequence for the $n^{th}$ layer, we have the following recurring sequence:

$$\begin{cases} u_n = 1 \\ u_{n-1} = u_n + F - 1, & \text{for a convolutional layer} \\ u_{n-1} = S * u_n, & \text{for a maxpooling layer} \end{cases}$$

with $u_0$ being the value of the effective filter size.

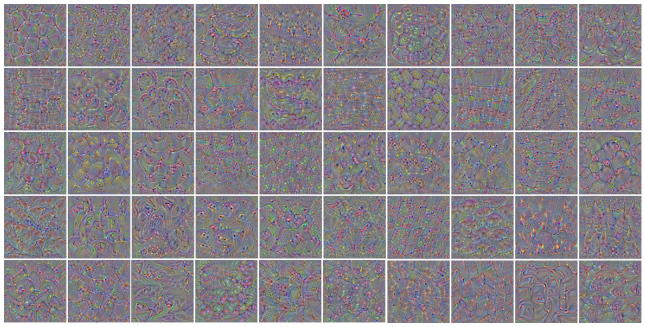


**Figure 43.** Feature visualization of 50 randomly selected filters from the last layer of the network trained with IMAGENET, starting from mean image, looping through initializations if necessary

The issue with such a computation comes from the fact that one needs to go back through all of the previous layers, not enabling the possibility to calculate the effective filter size of one layer, knowing the one of the previous layer. This section therefore explains how to "invert" this sequence and calculate the effective filter size according to the one of the previous layer.

**Proof** Let $f(x)$ be the function to find, to calculate the effective size, x being the initial effective filter size. This function is a composition of two functions: $g(x)$, representing a convolutional layer, therefore, $g(x) = x + F - 1$ and $h(x)$, representing a maxpooling function, with $h(x) = S * x$. It is obvious that

$$g(x + a) = g(x) + a \tag{2}$$

and

$$h(x + a) = h(x) + h(a) \tag{3}$$

By recurrence, since f is a composition of these functions, we get that

$$f(x + a) = f(x) + h^n(a) \tag{4}$$

with n the number of max pooling layers before the targeted layer.

Considering that the targeted layer is a convolutional layer and the previous layer also, if the effective filter size of the previous layer is $size_{n-1} = f(1)$, seeing that the effective filter size of the input can be size as of 1, then we get $size_n = f(F) = f(1) + h^n(F - 1) = size_{n-1} + (F - 1) * S^p$ with p the number of maxpooling layers. If, on the contrary, the previous layer is a maxpooling layer, then we have the following formulae: $size_{n-1} = f(1)$ and $size_n = f(F * S) = f(1) + h^n(F * S - 1) = size_{n-1} + (F * S - 1) * S^p$ seeing that the initial value of a convolutional layer going through a pooling layer is of F * S.

**Conclusion** Since we can get the same reasoning if the targeted layer is a convolutional layer, we get the following formulae:

If convolutional layer:

$$\begin{cases} size_{next} = & size_{prev} + (F-1) * S^p, \quad \text{if prev is a convolutional layer} \\ & size_{prev} + (F * S - 1) * S^p, \quad \text{if prev is a maxpooling layer} \end{cases} \tag{5}$$

If maxpooling layer:

$$\begin{cases} size_{next} = & size_{prev}, \quad \text{if prev is a convolutional layer} \\ & size_{prev} + S^p, \quad \text{if prev is a maxpooling layer} \end{cases} \tag{6}$$

with p the number of maxpooling layers before the targeted layer.