

Visualization of Neural Networks

David Bauer*
TU Wien

Lukas Maximilian Masopust†
TU Wien

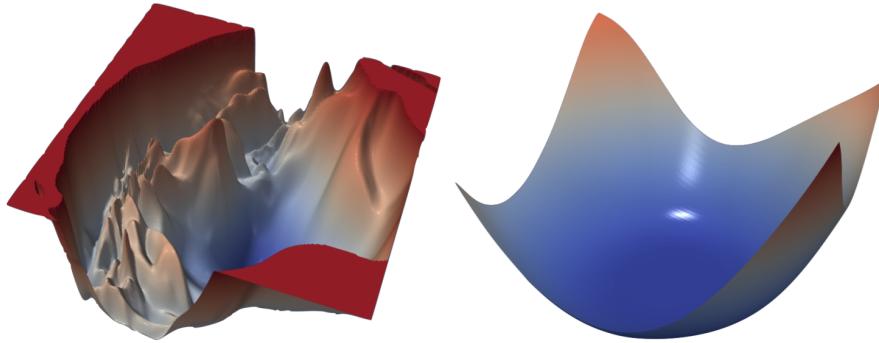


Figure 1: [Li et al. 2017]

Abstract

Recent years have seen a surge in the application of machine learning - especially deep learning. These methods enable developers and researchers to solve problems which seemed too complex for traditional methods just ten years ago. However, these advances do not come without the cost of a set of unique problems. Neural networks are highly complex structures. Their internal configuration changes constantly during training and with increasing numbers of neurons. It is hard to comprehend how each individual component influences the outcome of predictions. In this paper, we look at several ways to manage this inherent complexity by visualizing different aspects of neural networks. Approaches range from displaying a simplified version of the graph-like structure of a network to visualizing activation maps of single neurons. There are also ways to generate terrain-like plots of error functions. This is especially useful in situations where finding local minima might not be trivial. This state-of-the-art report gives a rough overview of the wide variety of approaches to the ever increasing complexity inherent to neural networks.

Keywords: machine learning, neural network, deep neural network, deep learning, heatmap, activation, graphs, visualization

1 Introduction

1.1 Background and Motivation

As the both of us work in the field of medical informatics where we currently develop neural networks for bone assessment, we oftentimes wonder what actually happens inside the networks we train and evaluate. Since, as of now, there does not appear to be lots of focus on neural visualization (at least in our working field), we thought it to be interesting, not only to us, but for everybody involved in working in machine learning on what current approaches to visualization might be.

1.2 The Problem

Data, especially large collections of analytical data in various fields of science, can pose enormous challenges for the people involved in analysing these data. Datasets often exceed the limits of intuitive comprehensibility. One approach, that is frequently used to surmount these large amounts of data are computer aided visualizations. They are a powerful tool to categorize, classify, analyse and visually process data to make it more understandable. For instance, when looking at data in tabular form,

* e-mail: e01442385@student.tuwien.ac.at

† e-mail: e01427382@student.tuwien.ac.at

it is typically harder to grasp its meaning as opposed to looking at the same data processed as a graph or bar chart.

The problem we will be discussing in this paper is the usage of visualization techniques for processing data, that comes from neural networks. Aside from solely working on the visualization on a layer-level (which will still be the main focus of this work), we also want to give readers a more meta-level picture visualizing neural networks as a whole.

1.3 Our Contribution

As we soon saw, this is a rather niche problem to which good literature is scarce. We contribute to the stated problem by giving a comprehensive overview on the matter and explaining different ways in which aspects of neural networks can be visualized.

Following this introduction, we will explain how we came to find the literature we are reviewing and see which results we obtained from our work. Furthermore there will be a discussion as to how to interpret our findings and how visualization techniques could be employed in the field of future neural network visualization.

2 Terms and Definitions

In this section we will define the two most important concepts used in this work.

2.1 Machine Learning (Li2017ML)

Traditionally, computers were designed to be programmed with specific instructions, which had been executed in an orderly fashion. Machine learning changes these paradigms drastically. The core principle of machine learning is that computers are shown data which they are then to interpret. Through adequate ground truth data and error functions, the computer is given feedback about its performance and can therefore adjust. After a varying amount of time, the trained computer will have adapted to the underlying structure of the data enough, so that it can make educated predictions about previously unseen input. What these procedures are programmed to do is to approximate a pre-defined goal. This concept is best portrayed by an example: Assuming the goal is to identify handwritten numbers, the machine learning algorithm is provided with the raw image of the written symbol. At first, it will guess at random which number was represented by the drawing. After each guess, it receives the correct answer. It then adjusts its parameters to fit the desired outcome. This process is repeated many times with different symbols until the parameters are adjusted so precisely that the algorithm – which is called a "model" in the machine learning domain – can detect any arbitrary handwritten number.

The example above is fairly simple but this process works for any task designed to identify or classify data, given enough raw input material. One maybe more significant application, is to classify histological images in terms of cancerous cell expression, which could lead to early diagnosis of tumors.

2.2 Deep Neural Network (DNN)

A deep neural network is a network of linked computational nodes, aiming to mimic the way our brain works. Each node has input and output channels and a function F to determine output from input. Neural networks are a sub-category of machine learning that build upon its basic concepts. Only in recent years it has become possible to use vast DNNs due to the increased performance of modern-day GPUs. Since those networks can house millions of parameters and with deep learning being applied in nearly every area of today's problem-solving, the need for visualizations in this area is steadily increasing. Therefore, DNNs are sometimes regarded as *black-boxes* with no clear insight into their decision-making process.

3 Method

While machine learning has been applied to lots of problems in the past, recent years have seen an unprecedented rise in the application of neural networks for problem solving, especially with the ongoing hype of deep learning. Due to this rise, visualization approaches have lagged behind, rendering machine learning to be this oftentimes so-called *black box*. Hence, publications that are older than a few years may not portray the current landscape of machine learning visualizations correctly. This is why in this state-of-the-art report, we aimed to find the most recent papers on the topic and also tried to include approaches from Google, arguably one of the most important players in today's machine learning field.

3.1 Sources and Search Strategies

For choosing the works we wanted to include in this STAR, we chose IEEE Xplore, Google Scholar and Elsevier as our primary sources. Relevant publications were mainly obtained from the websites of Springer, the Cornell University Library and IEEE Xplore.

In order to make our results reproducible, we list our search terms in a list below.

- machine learning
- deep learning
- neural network
- deep neuronal network
- artificial intelligence
- layer visualization
- neuron visualization
- deep visualization
- data visualization
- graph visualization
- interactive visualization
- user interaction
- flow layout

Those terms were either used on their own or in combination with others.

3.2 Criteria and Constraints

The search terms listed above reveal a large number of publications relevant to neural network visualization,

which is why we will briefly explain how we decided on the papers that were included in our work.

Since not every paper about machine learning is also discussing visualization, we discarded those that were not specifically focussed on the visual representation of data. Further, as discussed above, we only took papers into consideration that had a fairly close publication date to 2019, which is why the *oldest* publication we used for this STAR was released in 2015. Also, we aimed at only using papers that had a high citation count which is a common measure to evaluate the importance of a publication.

To choose literature relevant for this STAR we first read through the abstract of every publication on our shortlist. Then, by also reading the conclusion we decided which publications were significant to this work.

4 Overview of Works

4.1 Network Structure Visualization

4.1.1 TensorFlow's Visualization API

Because Google is one of the major players in today's machine learning field, this section will address how their platform TensorFlow incorporates visualizations to aid its users in better understanding the underlying metrics and parameters of their deep learning models.

In their paper, Wongsuphasawat et al. explain the design and background of the visualization process Google's API uses to give users a visual overview of their underlying neural network structure and dataflow [Wongsuphasawat et al. 2018].

Because these kinds of models can contain well over thousand of low-level components – each with individual parameters – standard layout techniques such as flow layouts can produce diagrams that are difficult to read. Therefore, TensorFlow is bundled with a Graph Visualizer that, according to user feedback included in their paper, enhances the comprehensibility of model structures.

Goals

To gain a good understanding of what potential users might find helpful when visualizing their networks, Wongsuphasawat et al. consulted people involved in machine learning. Results suggested the importance of the following points:

- Overview of high level components
- Differentiation and similarities of components
- Examination of nested structures
- No referring back to code to gain knowledge of details

TensorFlow Graph Visualizer

Wongsuphasawat et al.'s visualization approach is based on a standard flow layout algorithm that was introduced by Sugiyama et al., representing directed graphs in a

flow layout [Sugiyama et al. 1981]. This method of visualizing graphs (Figure 2) first removes cycles as well as layers and orders nodes. Afterwards, coordinates are assigned.

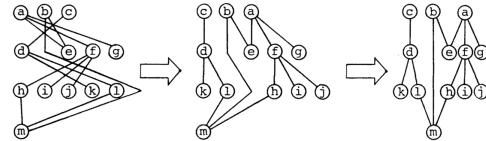


Figure 2: Sugiyama's method for visualizing graphs [Sugiyama et al. 1981]

After this pre-processing phase, a series of graph transformations is applied to make illegible graphs easier to read (Figure 3).

First, non-critical operations are de-emphasized by extracting them from the layout and depicting them as either small circles or bar chart icons, depending on their role. The paper states, that those operations account for approximately 30% of a typical network.

In the second phase, edges are bundled and a clustered graph is formed. Here, edges only connect nodes that are siblings in the hierarchy. This ensures that on interaction only subgraphs need to be refreshed, saving resources and improving responsiveness. Rounded rectangles represent group nodes, that are grouped by unique namespaces, such as "train".

As in the first step, in the third stage, non-critical nodes are extracted from each subgraph, rather than from the graph as a whole. Extracted nodes are thereby placed on the right side, as shown in Figure 3-d. Connecting nodes are also represented and give additional information, when hovered over. A set of heuristics is used to determine non-critical nodes, including searching for declaring and initializing variables.

As can be seen in Figure 3, this process declutters the graph by a high degree. Since Wongsuphasawat et al. use heuristics in some cases, unwanted results can be overruled by the user afterwards.

Feedback

Wongsuphasawat et al. gathered internal and external feedback regarding this visualization approach. Unfortunately their paper doesn't provide any kind of statistical evaluation, thereby rendering a scientific comparison useless. According to them, internal feedback gathered by questionnaires was overwhelmingly positive, whereas anonymous feedback found online had a "positive undertone" [Wongsuphasawat et al. 2018, p. 9].

4.1.2 TensorFlow Playground

In their paper, Smilkov et al. describe how Google's TensorFlow Playground aims at giving non-experts a quick and easy way into deep learning. Since the inner workings can sometimes become hard to read and interpret, the goal was to offer an interactive demo of an already trained deep neuronal network. TensorFlow

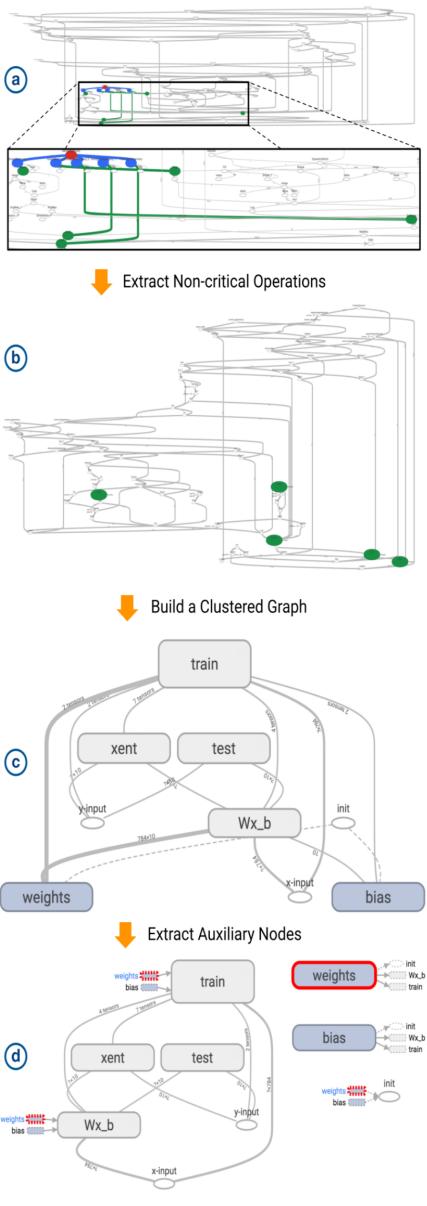


Figure 3: TensorFlow’s graph transformations [Wongsuphasawat et al. 2018]

Playground is an open-source in-browser tool that wants to give “novices with mathematical and practical intuition” [Smilkov et al. 2017, p. 1] a shortcut to understanding the basics of machine learning.

Their tool offers a network diagram of a DNN that is designed to solve classification or regression problems. As we can see in Figure 4, a circular shape with blue and orange dots can be used as training data. Two abstract features, x_1 and x_2 are then used to generate the gradient that can be seen in the neurons. In this way, users are able to see the different stages of data discrimination within the structure of a neural network. The combination of all neurons is then used to generate a heatmap on the right side that represents the output of the network. With more iterations and therefore more training, one

can see, that the blue dots are separated better from the orange ones over time, giving a visual representation of the mathematics underlying the network.

Smilkov et al.’s playground offers people, as for example students, an easy way to experiment with a network without any prior coding experience. While the mathematics behind deep learning can sometimes feel overwhelming for newcomers, their website offers a much more straight-forward approach to visualization of complex functions. According to their paper, TensorFlow Playground has gained positive reactions on the internet, although no statistical evidence was offered to support their claim.

4.2 Network Activation Visualization

4.2.1 Multi-Neuron Activations

Another way to interpret visualization of DNNs, is to give a visual representation of the layers a network uses for its decision making.

In their paper, Yosinski et al. describe tools they developed, that enable the user to see what each neuron “sees”, or expects to see, thereby making it easier to understand what a network actually interprets on the level of single neurons [Yosinski et al. 2015].

Concept

Yosinski et al.’s method uses the input of a webcam to demonstrate the activation function, which is the optimal activation for each neuron respectively. This can be thought of as the input (image) that a neuron expects to see. These visual cues are then used to generate the final decision a network makes about a certain input.

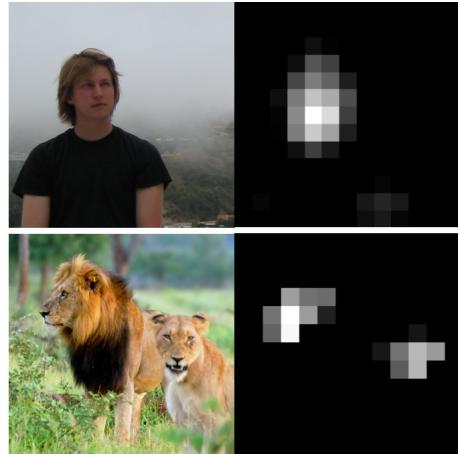


Figure 5: Activations for faces in a DNN trained to recognize human and animal faces [Yosinski et al. 2015]

As an example, a network that is trained to classify human and animal faces can be used to gather knowledge about the role each neuron plays in a network. In Figure 5, parts of the relevant input images can be seen. The images on the right side show which parts of the input images (left) matter for the classification of

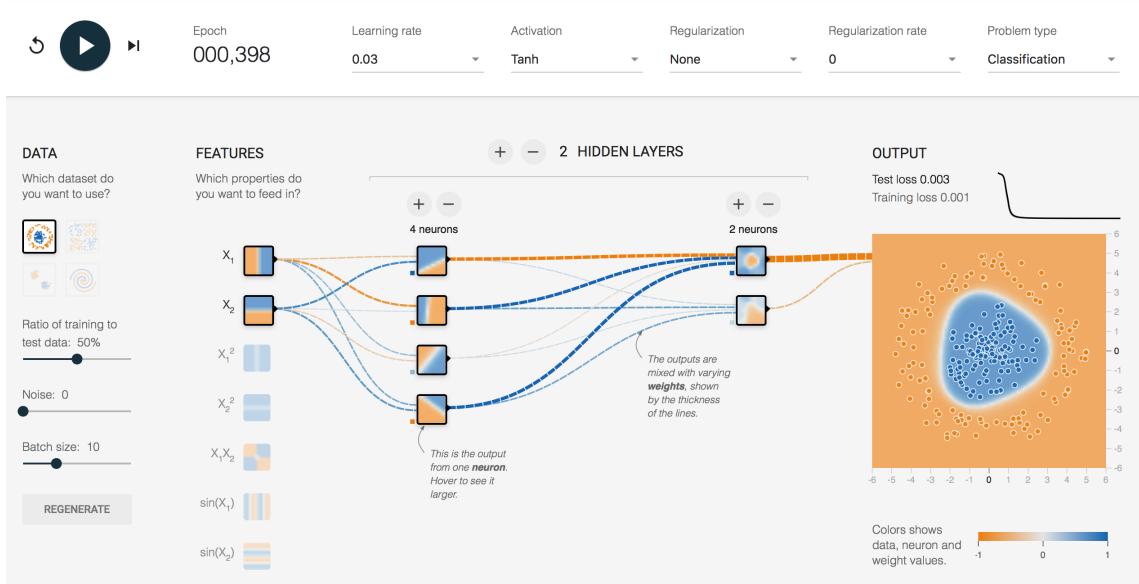


Figure 4: TensorFlow Playground [Smilkov et al. 2017]

this DNN. The conclusion the authors drew from this analysis is that layers appear to be quite local, meaning that instead of the expected distribution on all layers, there are specific neurons that show a higher reaction to, for example edges. Thereby, they create detectors for more complex concepts together with other neurons. Different local feature detectors can then make up whole objects, such as animals.

Visualizing these attributes of machine learning models can yield unexpected results. For example, a network that was trained to detect environments, such as playgrounds, shows some layers that are primarily activated by faces of children, even if this particular feature was not trained intentionally.

The second matter of interest of their work was the introduction of regularization options that help generating images that are visually better interpretable. Firstly, a decay function is used to decrease the importance of high values that otherwise would change the bias into an unfavorable direction. Secondly, a Gaussian blur is applied to filter high frequencies out of the input. If included, those high frequencies would result in generally favorable high activations but would at the same time decrease interpretability of the neurons attributes. Next, pixels that contain information close to zero are thresholded, since they usually do not contain relevant features. Further, the authors clipped pixels that only led to low activation to zero.

Yosinski et al. offer their project as open source interactive tools, thereby making DNNs more accessible for everybody interested in seeing what a DNN actually sees.

4.2.2 Regional Relevance Visualization

Similar to the paper discussed above, Samek et al. are also proposing an approach to visualize regional perturbation leading to a recognition decision via heatmaps [Samek et al. 2017].

To demonstrate what specific regions or features of an

image are of greater or smaller importance, they compare three ways of generating heatmaps:

- Sensitivity Analysis
- Deconvolution Method
- LRP Algorithm

Sensitivity heatmaps utilize sensitivity analysis. This means that the local sensitivity is used to demonstrate what, for example, makes a bird more or less likely to be interpreted as a bird. Therefore this method shows how small changes in a single pixel value influence the output of a network as a whole. Large values indicate that a pixel has a severe effect on the classification function. Since this approach focusses on local features, global features are underrepresented, leading to the drawback that the generated heatmap does not fully explain a classification of an image.

Deconvolution heatmaps are based on the concept of mapping activations of a network's output to pixel space via backpropagation. This method will, at first glance, give reasonable explanations of a decision, since the heatmap shows a matching input pattern for a classification object, meaning that the contours of a bird will be highlighted. The downside is, that, according to Samek et al., there is no direct correspondence between the generated scores and the contribution of an individual pixel to the overall decision.

Relevance heatmaps appear to be the most promising approach. They use Layer-wise Relevance Propagation (LRP) to deconstruct a decision into the relevances on a pixel level, therefore showing the contribution of an individual pixel to the end-result. This technique, compared to sensitivity heatmaps, is far more favorable towards global features and shows both, local and global impact in its heatmap. The most apparent benefit of LRP heatmaps is, that they not only visualize the positive evidence for a certain classification, but also

show what pixel data had a negative impact on the decision. In the example shown below (Figure 6) we can see that LRP, compared to deconvolution, highlights positive evidence as red, but also negative evidence, that, for example, speaks against the decision that the given image represents a 9.

4.2.3 Multifaceted Feature Visualization

In neural networks, each neuron detects one or more features that are further propagated for the decision-making process. Individual features can be visualized as can be seen in the previously discussed publications, but so far we have not covered the visualization of multifaceted features in a single neuron. In their paper, Nguyen et al. discuss the importance of visualizing those for more complex scenes, where multiple features achieve the highest activation in a neuron [Nguyen et al. 2016]. Previous work, they state, did not take those multiple facets into consideration, which led to mixes of e.g. colors or objects. Therefore, their work focusses on demonstrating an algorithm that produces synthetic images that correspond with the highest activation of a neuron (Figure 7). For example, a bell pepper classifier has to deal with red, green and yellow bell peppers. So there may be different images (facets) that achieve optimal activation. In this case, differently colored and oriented bell peppers. So there is no one single image that creates maximum activation. The aim of this work was to create representations that take this attribute of classifiers into account and give a more accurate depiction of optimal, multi-faceted inputs.

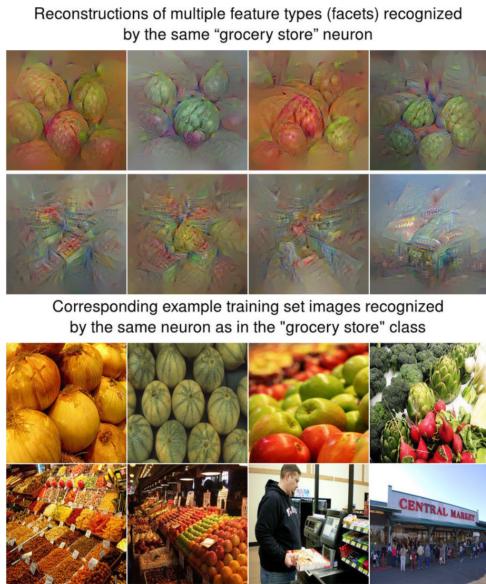


Figure 7: On the top, there are eight feature facets that activate the same neuron which was trained to identify grocery stores. Those images were synthetically created. On the bottom, the corresponding training images that were used for initial training [Nguyen et al. 2016].

Concept

Nguyen et al.'s deep visualization approach relies on a CaffeNet-based architecture DNN that was trained on the ImageNet dataset. They use an optimization algorithm that aims at optimizing the color-values of the pixels in an image to achieve maximum activation in a given neuron.

Simplifying the approach for the sake of this state-of-the-art report, their algorithm uses a DNN's abstract knowledge of a class. This information is then clustered, which yields several clustering centers, which all represent different facets of this particular classifier. To obtain the maximum activation image of a facet, the centroid of the cluster is calculated and conventional input maximization techniques are used to produce a part of the final representation. This process is repeated for all cluster centroids to obtain the final, combined representation of the multi-faceted maximum input.

An example for such a facet cluster can be seen in Figure 8.

4.3 Network Error and Loss Visualization

In some cases it is more desirable to explore neural networks in terms of training data and loss rather than looking at an already trained model. The inherent difficulty when choosing a particular network architecture is knowing how hard it will be with given training data to reach a global minimum in the error function and therefore reach optimal network performance.

4.3.1 Loss Landscapes

Efforts have been made to visualize these so-called loss landscapes, which are essentially three-dimensional plots of loss functions [Li et al. 2017]. One problem that typically arises from these kinds of visualizations is that it is not immediately apparent to what degree changes in the network weights influence the change in error. In other words, it is hard to estimate how rough or smooth the loss landscape is. This problem is due to the scale invariance of network weights, since most networks use normalization techniques. In that way, relative changes in weights and activation intensity do not play a role in the actual performance of the network. Li et al. have shown how to solve the problem of scale invariance [Li et al. 2017]. In general, a smooth landscape is more desirable since chances of getting "stuck" in a local minimum are smaller than in really "noisy", "spiky" environments. Sharpness or flatness of the minima themselves also plays a role and is, in most cases, directly connected to the roughness of the loss surface.

It is important to note that the morphology of loss landscapes highly depends on the chosen network architecture. Small changes can have a great impact on the outcome loss-wise. Figures 9 and 10 show the same ResNet with (10) and without (9) so-called skip connections, which are structural modifications of a standard network [He et al. 2015].

It becomes immediately clear from these examples that visualizing these landscapes can be a great benefit when developing new kinds of networks. This is because it enables researchers to see the impact of their modification at first glance. Conventionally, the use of mere mathematical descriptions of the changes may not convey this

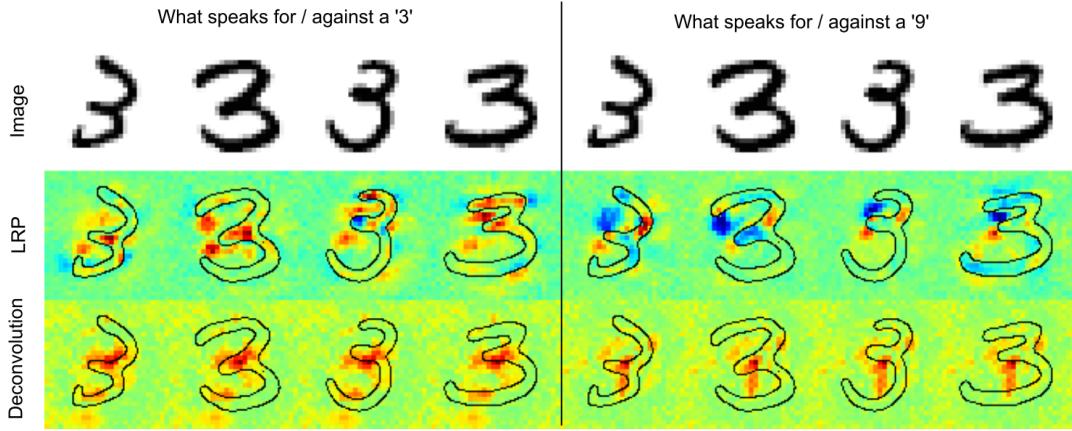


Figure 6: Heatmaps for classification of numbers where red means that these areas strongly speak for the requested pattern whereas blue denotes areas that speak against that pattern [Samek et al. 2017]



Figure 8: Illustration of the facet cluster from which the final facetted representation is generated [Nguyen et al. 2016]

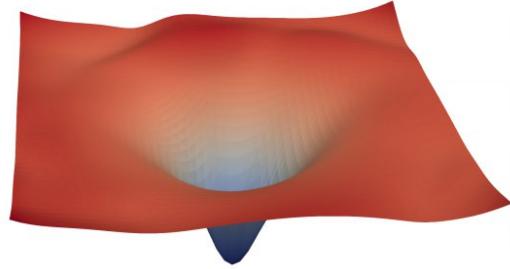


Figure 10: Smooth loss landscape of a ResNet-56 with skip-connections [Li et al. 2017, He et al. 2015]

$$d \leftarrow \frac{d}{\|d\|} \|\theta\| \quad (1)$$

Figure 11: Simplified version of the original normalization equation, where θ denotes the network weights [Li et al. 2017]

information as easily.

4.3.2 Filter-Wise Normalization

To address the issue of scale invariance, Li et al. proposed a method called Filter-Wise Normalization. In this approach, the loss surface surface is generated by evaluating the error function in random directions d . Conventionally, this was prone to the particular scale of the neurons. To avoid this, instead of using these directions d directly, they are first normalized with respect to each individual filter/neuron.

This new method allows the creation of plots that are comparable among each other.

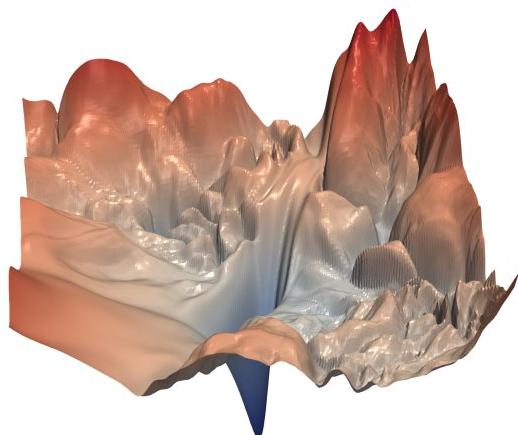


Figure 9: Rough loss landscape of a ResNet-56 [Li et al. 2017]

5 Discussion

In the results section of this report it could be seen that there are three main categories of visualizations of

neural networks. A neuronal network can be visualized on a meta level by representing the structure of its nodes and high level constructs in the form of graphs. Compared to that, the direct input, output and everything that happens in between can also be visualized to give the user a better understanding of the actual inner workings of a network. In addition, the underlying loss function also lends itself to visualization as it provides insights about the feedback behavior of the network.

In this section, we will discuss the pros and cons of each approach.

5.1 Network Structure Visualization

We have looked at the way Google’s TensorFlow utilizes algorithms on directed graphs to make complex neuronal networks easier to read [Wongsuphasawat et al. 2018]. Because being available within their API, this is a feature that is likely to be beneficial for everyone who uses TensorFlow as their machine learning platform. As seen in Figure 3, Google’s approach makes convoluted graphs very much readable, which allows for an improved analysis of neural networks. Since there is no obvious downside to their approach, it can only be noted that it would be beneficial if they would provide detailed comparisons with other emerging techniques in this area.

As seen in the work of Smilkov et al., there is a great need for visualizing the so-called *black box* that neural networks can sometimes seem to be [Smilkov et al. 2017]. Therefore, we discussed their method of visualizing networks for teaching and learning, TensorFlow Playground.

5.2 Network Activation Visualization

Yosinski et al. offer an open source project that aims to show the visual input and interpretation from every layer to every neuron of a DNN [Yosinski et al. 2015]. They therefore offer a very interesting insight into how a neuronal network actually generates an output and therefore classifies an image.

With their work on heatmaps, Samek et al. have shown, as Yosinski et al., that the knowledge about the workings of individual components can be quite insightful for engineers of a DNN. They proposed different ways of representing the decision-making-process of a neural network as heatmaps [Samek et al. 2017]. We have seen, that there were clear advantages with some approaches, such as LRP, where even negative evidence against a decision can be shown. It seems clear that the usage of discussed techniques would be greatly beneficiary for people working on optimizing the performance of specific neural networks.

5.3 Network Error and Loss Visualization

Li et al. have shown how visualizing loss functions can aid and accelerate the development of new network architectures [Li et al. 2017]. With their example about skip connections [He et al. 2015], they show impact on generalization performance of a particular network can be easily determined by employing visual representations of the loss function.

Their contribution of normalized evaluation of loss surfaces in the form of Filter-Wise Normalization further improves the comparability of different network architectures. This greatly aids researchers that try to quickly evaluate proposed changes to already established architectures.

5.4 Summary

The discussed papers provided an interesting look at the current state-of-the-art in the visualization of deep learning models. In general, we can discern that visualization in machine learning is seen as a tool to make DNNs more easily accessible and better modifiable. Whether on a meta-level or down to the visualization of a single neuron, the approaches discussed above all focus on making deep learning more understandable and more beginner-friendly. Since the wide applications of deep learning are just at the beginning of being fully understood and utilized, visualization will most probably play a major role in the neural networks of the future, especially since increased computational power will lead to more complex and advanced models in the future.

6 Conclusion

This state-of-the-art report, presented a range of novel ways to visualize DNNs. We discussed the benefits and shortcomings of approaches. Those approaches went from high-level to low-level and tried to solve different problems of the visual representation of neuronal networks.

We selected examples about how neural networks are represented on a meta-level and how a network can be seen on the level of a single neuron. We have also presented the benefits of visualization of entire networks for optimizing and studying DNNs more closely. Furthermore we looked at ways to visualize loss and error functions of particular architectures.

The aim of this paper was to give a comprehensive overview on the current state-of-the-art in neural network visualization. This was achieved by carefully selecting relevant and recent work in the field.

The discussed papers provided an interesting look at the current state-of-the-art in the visualization of deep learning models. In general, we can discern that visualization in machine learning is seen as a tool to make DNNs more accessible and easier to modify. Whether on a meta-level or down to the visualization of a single neuron, the approaches discussed above all focus on making neural networks more understandable and more user-friendly. Since the wide applications of machine learning are just at the beginning of being fully understood and utilized, visualization will most probably play a major role in the development of DNNs in the future, especially because, with increased computational power, neuronal networks will grow even more complex.

The visualization of neural networks is important and will become more relevant in the future, as machines and networks advance over time. There is still a lot of

potential but promising approaches are already emerging and there is quite a lot of research in the field. Significant global companies like Google or Facebook are also interested in pushing this aspect of neural networks. Further, we want to highlight, that a lot of the projects regarding visualization in this field are open source and easily accessible for anybody interested in getting started with deep learning.

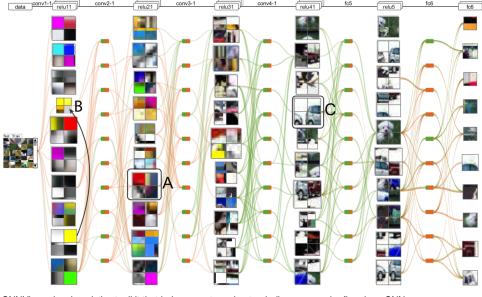


Fig. 1. CNNVis, a visual analytics toolkit that helps experts understand, diagnose, and refine deep CNNs.

Figure 12: Sample image of CNNVis. [Liu et al. 2016]

7 New Papers - Overview

7.1 Towards Better Analysis of Deep Convolutional Neural Networks

In their paper, Liu et al. introduce a visualization approach for visualizing deep convolutional neural networks (CNNs) [Liu et al. 2016]. CNNs present a major challenge for scientists and experts working with them for a number of reasons as detailed in their work. First, neural networks can consist of hundreds of individual layers that themselves are made up of thousands of neurons. This inherent problem makes analysis on a neuron-level difficult. Secondly, training a CNN is a time-intensive task, where training can last for hours, days, or even weeks. Therefore, the correct identification of potential problems is key to saving time and resources.

With their approach, CNNVis (12), Liu et al. try to tackle those challenges in order to give experts a tool for an easy visual analysis of their networks. Their text describes how CNNVis works and evaluates it in three user-studies where they measure performance improvements across multiple neural networks.

7.1.1 Approach

CNNVis uses a number of different steps to achieve a comprehensible CNN visualization. Those include the following.

- The network is converted into a directed acyclic graph (DAG)
- Layers are clustered
- Representative layers are chosen
- Neurons of those layers are clustered
- Representative neurons are chosen
- A hierarchical rectangle packing algorithm is applied
- A matrix reordering algorithm helps with visualizing activations
- A bi-clustering-based edge bundling method is applied

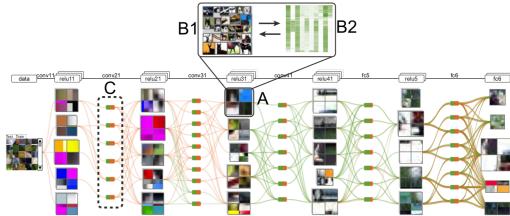


Fig. 6. Visualization overview.

Figure 13: Sample image of CNNVis. [Liu et al. 2016]

Since we don't want to focus on every detail of how their process works, we'll outlay some of the steps listed above.

In figure 13 we can see a detailed look at the visual output their program produces. As the input data is passed through the network, more and more features are being learned in accordance with their previous layers. Left of Fig. 13C we can see a representative network layer with some representative neurons. The neurons are chosen by sorting for maximum activation. A pool of five neurons-clusters is picked which can be switched by the user. When clicking on a individual neuron-cluster (Fig. 13A), the user can see how certain input images (which themselves are user-swappable) effect the activation of a single cluster (Fig. 13B1 and B2). Through that, the features a network has learned can be understood and decisions for retraining can be made. Finally, the biclustering-based edge bundling algorithm allows for decluttering the vast number of inter-neuron connections.

To summarize, Liu et al.'s application is a good way to visualize deep neural networks. Their clustering techniques allow for a stripped-down exploration of a network, where only necessary components can be viewed. User interaction allows for a flexible alteration to specific user needs. To demonstrate that their approach improves many aspects of CNN trainging, they conducted multiple user studies. Those studies were generally favorable of CNNVis and found it to be helpful for detailed network analysis.

7.2 Towards Better Analysis of Deep Convolutional Neural Networks

7.3 Analyzing the Training Processes of Deep Generative Models

In their paper, Liu et al developed the tool DGMTracker, which allows experts to analyze deep generative models – the most popular version being the generative adversarial network (GAN) – at different granularities [Liu et al. 2018]. DGMTracker offers the following three modules. Snapshot-, layer- and neuron-level analysis. To achieve this, DGMTracker is divided into two modules. Data flow visualization and Training dynamics analysis. Data flow visualization focuses on the general flow of input data towards the output layer. Experts can analyze hotspots and bottlenecks using this flow data. DGMTracker offers snapshot- and neuron-level analysis

of data flow data. Training dynamics are a collection of multiple parameters of interest to the expert. These include loss and average weight per layer. Those data are visualized in the Training dynamics analysis module of DGMTracker. Using these modules the expert can then perform analysis on different levels of detail in different modalities. An overview if DGMTracker's architecture can be seen in Figure 14.

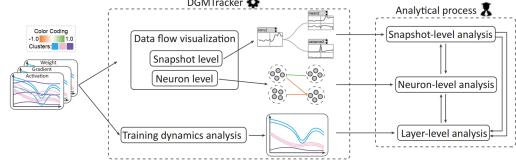


Figure 14: Overview of DGMTracker's architecture [Liu et al. 2018]

7.3.1 Snapshot-Level Analysis

Network states can be captured at different points in time during training resulting in a time series of snapshot data. This data is usually visualized either by using dimensionality reduction methods such as t-SNE or by employing multiple line charts. Liu et al relied on the latter approach, showing temporal changes in parameters such as loss or average weight in a layer (see Figure 15)

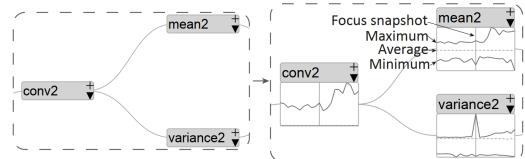


Figure 15: Snapshot-level analysis – each layer is portrayed as a separate area and metrics are shown within its boundaries [Liu et al. 2018]

These data cumulatively represent the overall training dynamics of a network.

7.3.2 Neuron-level Analysis

On this level, the expert can see how each neuron of one layer contributes to the input of the following layer. The contribution is assessed by using a credit assignment formula which maps each node's influence to a scale of minus one to one (where negative values mean backward contribution and positive ones denote forward contribution). These values are mapped to visual representations of the nodes through a color map. Connections between nodes follow the same scheme. Their thickness additionally shows the magnitude of contribution to individual connections. Since there are usually too many nodes in each layer to visualize them all separately, Liu et al have used K-Means clustering to reduce visual clutter. Neurons are then placed on a feature map to further reduce complexity (see Figure 16).

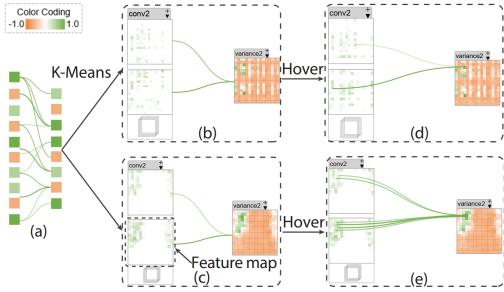


Figure 16: Neuron-level analysis – single node contributions are clustered using K-Means and are then mapped to feature maps to enhance legibility [Liu et al. 2018]

7.3.3 Training Dynamics Visualization

This second module of DGMTracker was designed to enable experts to analyze training processes on a temporal level. Snapshot data is visualized using different line charts showing metrics such as loss and layer activation over time. Again, visualizing all the data directly will usually lead to visual clutter. In this case, Liu et al propose blue-noise sampling to extract a representative overview of the raw data (see Figure 17). The advantage of blue-noise sampling over the conventional random sampling of snapshot data is that the former preserves outliers. This is especially important if the expert is to find layers and neurons that may cause training failure.



Figure 17: Training Dynamics – relevant data is picked from the raw data (left) using blue-noise sampling to preserve outliers (right) as compared to random sampling (middle) [Liu et al. 2018]

8 New Papers – Comparison

In this chapter, we are comparing the papers presented in the previous section. The categories are loosely based on the work of Hohman et al who did a comprehensive survey of the state-of-the-art of visual analytics in deep learning [Hohman et al. 2018]. In their paper, they treat the WHY, WHAT, WHEN, WHO, HOW and WHERE of visual analytics in this area. We will focus on the WHAT, HOW and WHEN in this chapter. This means that the papers of this work will be compared and categorized based on what specific data domain of deep learning they are visualizing and what means have been used to visualize that data. Additionally, it is important to understand if this is done during or after training (WHEN).

Hohman et al propose several categories of which we will use the following for our purpose.

8.1 What Data Is Visualized?

Papers in this report can be categorized as follows based on what kind of data they visualize.

8.1.1 Computational Graph and Network Architecture

8.1.2 Neural Network Edge Weights

8.1.3 Convolutional Filters

8.1.4 Activations

8.1.5 Error Gradients

8.1.6 Model Metrics

8.2 How Is the Data Visualized?

Papers in this report can be categorized as follows based on how the data is visualized – in other words, what kind of techniques are used.

8.2.1 Node-Link Diagrams

8.2.2 Dimensionality Reduction and Scatter Plots

8.2.3 Line Charts

8.2.4 Interactive Experimentation

8.2.5 Algorithms for Attribution and Feature Visualization

8.3 When Is the Data Visualized?

Papers in this report can be categorized as follows based on when the data is visualized. Visualizing the data during training can be a benefit since the expert can intervene as soon as anomalies are detected. However, visualizing data after one or multiple trainings have finished often yields a better general overview of the data [Hohman et al. 2018].

8.3.1 During Training (Online)

8.3.2 After Training (Offline)

References

- HE, K., ZHANG, X., REN, S., AND SUN, J. 2015. Deep residual learning for image recognition. *CoRR abs/1512.03385*.
- HOHMAN, F. M., KAHNG, M., PIENTA, R., AND CHAU, D. H. 2018. Visual analytics in deep learning: An interrogative survey for the next frontiers. *IEEE transactions on visualization and computer graphics*.
- LI, H., XU, Z., TAYLOR, G., AND GOLDSTEIN, T. 2017. Visualizing the loss landscape of neural nets. *CoRR abs/1712.09913*.
- LIU, M., SHI, J., LI, Z., LI, C., ZHU, J., AND LIU, S. 2016. Towards better analysis of deep convolutional neural networks. *IEEE transactions on visualization and computer graphics* 23, 1, 91–100.
- LIU, M., SHI, J., CAO, K., ZHU, J., AND LIU, S. 2018. Analyzing the training processes of deep generative models. *IEEE Transactions on Visualization and Computer Graphics* 24, 1 (Jan), 77–87.
- NGUYEN, A., YOSINSKI, J., AND CLUNE, J. 2016. Multifaceted Feature Visualization: Uncovering the

Different Types of Features Learned By Each Neuron
in Deep Neural Networks.

SAMEK, W., BINDER, A., MONTAVON, G., LA-
PUSCHKIN, S., AND MULLER, K.-R. 2017. Evaluat-
ing the Visualization of What a Deep Neural Network
Has Learned. vol. 28. nov, 2660–2673.

SMILKOV, D., CARTER, S., SCULLEY, D., VIÉGAS,
F. B., AND WATTENBERG, M. 2017. Direct-
Manipulation Visualization of Deep Networks. aug.

SUGIYAMA, K., TAGAWA, S., AND TODA, M. 1981.
Methods for Visual Understanding of Hierarchical
System Structures. vol. 11. 109–125.

WONGSUPHASAWAT, K., SMILKOV, D., WEXLER, J.,
WILSON, J., MANE, D., FRITZ, D., KRISHNAN,
D., VIEGAS, F. B., AND WATTENBERG, M. 2018.
Visualizing Dataflow Graphs of Deep Learning Mod-
els in TensorFlow. vol. 24. jan, 1–12.

YOSINSKI, J., CLUNE, J., NGUYEN, A., FUCHS, T.,
AND LIPSON, H. 2015. Understanding neural net-
works through deep visualization. *arXiv preprint*
arXiv:1506.06579.