Visualisation and Visual Data Analysis
Milestone 4

NeuroVisualizer - Understanding Neural Networks Made Simple

Project Page:
`http://wwwlab.cs.univie.ac.at/~a1125503/vis/`

Oliver Guggenberger, a1125503
Christoph Seebacher, a1127870
Ernst Naschenweng, a0826057

January 2018

Contents

1 Motivation

Our application uses the dataset from the CASS 2017 - Summer School. This dataset contains random shapes and noise of 100,000 different topologies of neural networks. These topologies differ mainly in the number of hidden layers, number of neurons, the used activation function and the learning rate. The results are in the columns of the TPR (true positive rate) and FPR (false positive rate) which describe the effectiveness of the neural nets.

Our goal is to provide a tool which lets users explore the dataset to find out which parameters result in good neural networks. Two things are novel about our approach. Firstly, our application analyzes 100,000 different neural networks while other visualizations only focus on one topology (with sometimes very large datasets). Secondly, we do not visualize the output of different layers. Instead, we focus on correlations between different parameters to gain insights in which settings have the greatest impact, positive or negative, on the quality of a neural net. This is essential for a global understanding of neural network parametrization where the visualization of individual layers is irrelevant. However, we do provide a graphical output of the classification result for each neural network available on demand. Multiple images can be browsed simultaneously with all relevant data shown in the tooltip.

Our application provides two dashboards. A general dashboard which follows Shneiderman's mantra to provide an overview first, enable zoom and filter to delve deeper into the data and show details on demand through a tooltip for each topology. Since overfitting in the training of neural networks is a challenge that deserves special attention, we created a second dashboard which lets a user tackle this specific question in a specialized user interface. There is also a guided mode with explanations of the most important concepts of neural networks to make it easier to understand and use our NeuroVisualizer.

2 Related Work

2.1 General Research

Our reserach for related work did not find any similar approaches. There are publications which focus on single neural networks and visualize them for large datasets like in Zeiler and Fergus. What our application does, however, is correlate the parameter settings for 100,000 different neural networks. It appears that this is a novel approach for our specific use case.

2.2 VisRSeq

Figure 1: VisRSeq User Interface

VisRSeq is a tool originally developed to visualize gene sequencing data. It can be extended with user defined visualizations using R. We evaluated VisRSeq during the low-fi prototyping phase if it could be used for our visualization, but we decided to use Tableau for the platform independance (web browser) and efficiency of implementation.

2.3 Visual Parameter Space Analysis: A Conceptual Framework

Sometimes if you have to handle problems or models with input parameters and therefor you get output parameters, like it is usual in neural nets, often it isn't the fastest way to work with the trial and error method for every single setup. Instead it would be a good method defining a space where all your different input parameters were set and you can now let these bunch of setups running through your model over night and get all the results at the early morning. This paper describes a framework which provides this functionality and way more.

The main focus of our project was to create an environment that lets users explore the impact of different setup parameters in a structured way. So this literature wasn't directly fitting our project assignment, because we already used the dataset of 100,000k different network topologies.

3 Approach

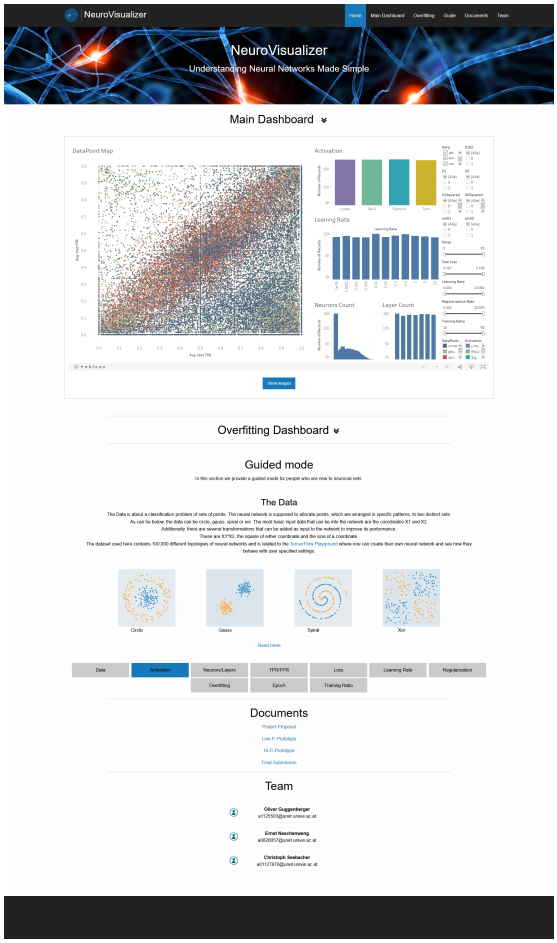

Figure 2: The NeuroVisualizer application integrated in the project website

3.1 Main Dashboard

The main view of NeuroVisualizer is a scatter plot called "DataPoint Map". It correlates all neural nets by the TPR and FPR (true/false positive rate, respectively) of the test data. A high TPR and a low FPR distinguishes "good" neural networks. Good neural nets are therefore in the bottom right corner while bad ones are in the top left. The view supports brushing so selecting a number of points will filter all other views.

The views to the right of the scatter plot show the number of selected points by various criteria such as the activation function or the value of the learning

rate. By selecting a number of data points one can see which parameter values provide the largest share of the number of selected points.

By having a scatter plot as the main view with the main quality parameters on both axis, one can easily identify "good" and "bad" neural networks which can then be analyzed in more detail. The scatter plot also highlights the different problem types like "spiral" or "circle" by color encoding. With the filters provided one can reduce the data space more and more until only the topologies of interest are left. From there, individual networks can be analyzed while returning to a more global view is possible without breaking the workflow in a simple click. The maximum detail with all parameters on the level of individual neural networks is available in the tooltip of each data point and the tooltip of the image related to the data point.

3.2 Overfitting Dashboard

The overfitting dashboard solves a specific question about overfitting a neural network during its training phase. In the scatter plots we correlate a quality score for the training and the test data. This quality score is the true positive rate (TPR) minus the false positive rate (FPR). Good neural networks have a TPR of 1 and a FPR of 0 for both the training and the test data and therefore are located in the top right corner. Encoding a score like this enables us to show more information and the score expresses the quality of a network better since a high TPR can mean little if the FPR is high too.

We selected the data type, activation function and regularization rate to provide different views on the same data space. According to our own research, these features have a significant impact on overfitting a neural network. These views, even though they look the same at a glance, are not logically connected. This is recognizable by the different color palettes. It is simply a slice through the data space with fixed x and y axis, but a different feature is highlighted in each view. Brushing works between the views to enable filtering down the data space. The bar chart to the right shows the learning rate distribution for the selected data points.

3.3 Guided Mode

Since the topic of neural networks in general and this dataset specifically is fairly complex and having a common understanding of the definitions is cruicial for successful usage of our application, we provide a little tutorial with explanations of important terms.

4 Implementation

4.1 Technologies used

NeuroVisualizer is implemented in Tableau and embedded in our project website. We also use the Tableau API to read the currently selected data points and show the respective images. These images then provide the most important information in a tooltip also via the API. This feature was implemented in Javascript without any other Javascript framework.

4.2 Implementation challenges

The most difficult implementation challenge was getting the images for each neural network on demand depending on the selected data points. We could not just ignore the images since they provide significant insights at the level of individual neural networks in combination with the information provided by NeuroVisualizer.

At first we tried to read the selected data points and the respective image path from the DOM tree with Javascript which was impossible due to the fact that tableau public doesnt create DOM objects for selected marks. After some research we found out that there exists a scriptin API provided by Tableau which allows to extract data and events from the dashboard and further process them in Javascript as needed. This made arbitrary data manipulation from the dashboard into the website much easier.

5 Results

5.1 Scenario of use - optimal parameters for spiral data type

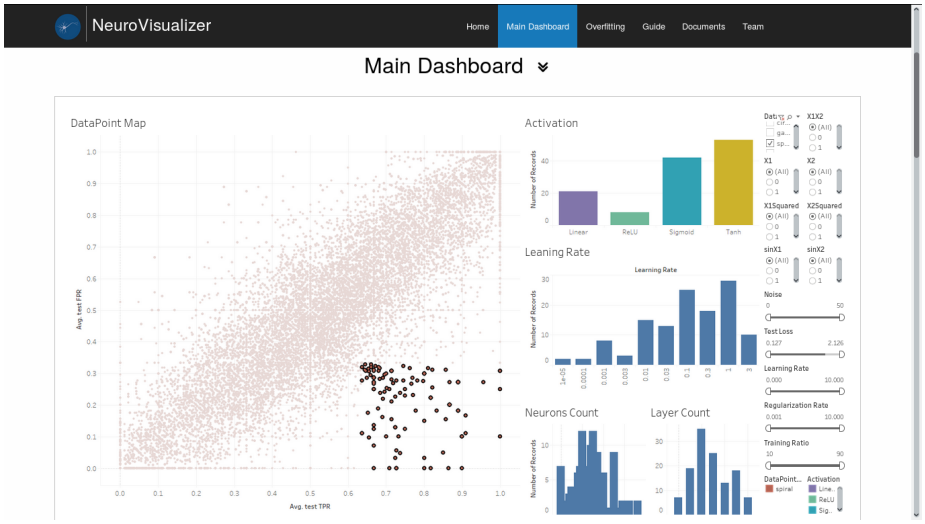

Figure 3: Good neural networks for data type "spiral"

The figure above is filtered to the spiral classification problem and as can be seen there are almost no very bad networks and a few very good ones. Good neural networks for the data type "spiral" mostly use the Tanh activation function followed by sigmoid. ReLU rarely resulted in good networks.

A higher learning rate is also favorable to a certain degree with a peak at values 0.1 and 1. Interestingly, the highest value at 3 is much less often a good choice for this data type. Two layers are the best choice followed by three layers. Overall, ten to twenty neurons are ideal. Figure 3 shows the tooltip of a selected point to have a look at the maximum detail of data available. Once points have been selected in the DataPoint Map, the button "show images" will become active and by clicking on it, it will list all images from the training phase as shown in Figure 4.

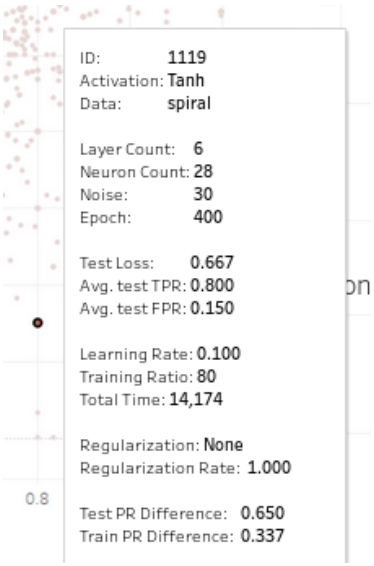

Figure 4: Tooltip of a selected data point of data type "spiral"

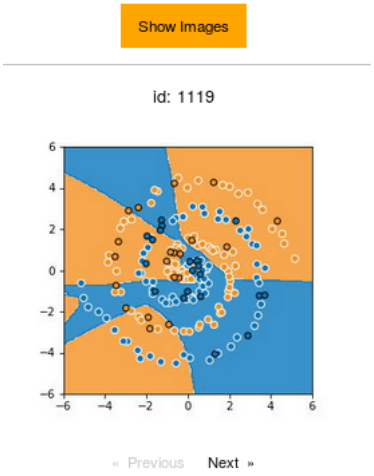

Figure 5: Image of the training phase of a selected neural network

5.2 The Kernel Trick

While exploring the dataset with NeuroVisualizer we found about 15.000 neuronal nets with zero neurons and zero layers. Our first and false delusion was that we should clean the dataset from those nets. Further investigations revealed the very interessting fact that some of those nets are highly efficent.

This was against our expectation and our previour knowlege about neuronal nets. In order to solve problems like the XOR problem at least one hidden layer is required [1]. It turned out that this is not entirley true and can also be solved by using the famous kernel trick. The kernel trick transforms the data into a higher dimension and avoids the explicit mapping that is needed to get linear learning algorithms to learn a nonlinear function or decision boundary [2]. Neuronal nets which leverage the kernel trick seems to work simmilar to networks with one hidden layer.

5.2.1 Insights

In the following subsections we discuss our insights about the kernel trick. We further investigated networks with zero neurons and zero layers and try to solve the given problem in the respective problem space.

5.2.2 XOR

Ca be solved using the Kernel trick by using X_1^2, X_2^2 and/or $Sin(x_1)$, $Sin(x_2)$

5.2.3 Gaussian

Ca be solved using the Kernel trick by using X_1^2, X_2^2 and/or $Sin(x_1)$, $Sin(x_2)$

5.2.4 Circle

Ca be solved using the Kernel trick by using X_1^2, X_2^2 and/or $Sin(x_1)$, $Sin(x_2)$

5.2.5 Spiral

Cannot be solved using the Kernel trick.

5.3 Performance

NeuroVisualizer is bottlenecked by the amount of data. While the data file is only 14 MB in size, it contains 100,000 records and the data transformations necessary for the different views are fairly complex. On top of that there is filtering which happens between the views.

The initial page load takes a few seconds and the response time of our application is around 5 seconds which is acceptable for an expert system. The resources for the project website are loaded asynchronously to speed up the loading time. The general processing time of the data is not related to the fact that our application is web based. The performance is the same if executed locally in the Tableau software so there is nothing we could improve performance using this technology.

[1] `http://toritris.weebly.com/perceptron-5-xor-how--why-neurons-work-together.html`

[2] `https://en.wikipedia.org/wiki/Kernel_method`

6 Discussion

6.1 Strengths

NeuroVisualizer is very easy to use and does not get lost in visual clutter or unnecessary features. We use various visualization techniques to encode information such as brushing, color encodings, position, filters and tooltips to convey insights efficiently. The user interface focuses on a quick overview with many zoom and filter options, but also the maximum detail on demand following Shneiderman's mantra. Since the implementation only requires a web browser, our application is available and usable anywhere.

6.2 Weaknesses

The performance and response times are not optimal, but we carefully optimized our application as much as possible with asynchronous loading of resources. However, we found nothing else to improve the runtime behavior given the tools chosen (Tableau and Javascript). An alternative implementation could have been done in D3, but considering that the Urbana crime statistics assignment was fairly slow with much less complex views, we were confident that a change of programming tools would not have alleviated the performance issues especially since running the Tableau workbook locally in the native client has a similar performance as the embedded version.

We also looked at color perception for different kinds of color blindness. Even though we carefully selected distinguishable colors in all kinds of scenarios of color perception, there are still some minor improvements that could be made especially in the overfitting dashboard with red-blindness (protanopia) and monochromacy in general. In case of monochromacy, most color encodings become unusable, but a color palette that is distinguisable in this case and also visually pleasing for people without impaired color perception, is very difficult to compose since in the overfitting dashboard there are more than one color palettes.

6.3 Lessons Learned

An important lesson we learned was the significance of the concept and prototyping phase. Whatever you don't account for in your prototype will cause more and more problems the later you discover the issue in your project. Thorough testing and questioning every approach over and over are cruicial to end up with a working user interface that solves the challenge described in the use case. Also the use case itself needs to be very well thought through and narrowed down. There is no point in starting with the implementation if it is still unclear what needs to be done exactly and why. That is, concept and prototype have priority before anything else. Luckily, we decided to do our presentation for M2 and got early enough feedback to completely re-do our prototype basically starting from scratch. We better defined the use case and iteratively improved our ap-

plication in many group meetings where we re-fined our concept and challenged each other's ideas until we could not find improvements anymore and ended up with the current design.

7 Task Separation

In both the concept (fine-tuning) and implementation phases every group member worked on many topics all over the project scope which makes it hard to itemize exact contributions per person.

- Literature research (1 paper per person): Everyone
- Implementation & Testing (Dashboards): Chris, Ernst
- Use case testing & Insights: Chris, Oliver
- Website programming: Oliver, Chris
- Tableau API programming: Oliver
- Report: Ernst
- Presentation: Everyone

8 References

- VIS lecture slides
- T. Munzner: Visualization Analysis & Design: Abstractions, Principles, and Methods
- Sedlmair, Michael et al.: Visual Parameter Space Analysis: A Conceptual Framework (link)
- Zeiler, Matthew and Fergus, Rob: Visualizing and Understanding Convolutional Networks (link)
- `https://github.com/hyounesy/cass2017_vis`
- `http://playground.tensorflow.org`
- `https://www.tensorflow.org/get_started/summaries_and_tensorboard`
- `http://visrseq.github.io/`