Visualization to scale and interpret large-scale Deep Neural Network Models

Report for the "Visualization & HCI" seminar (SS 2020)

Jagyan Prasad Mahapatro
Fachbereich Informatik, TU Kaiserslautern

Abstract. Deep Learning has been successful in many decision-making tasks in many different domains but the internal workings of these models still remain difficult to understand. Finding out which features influence more and which less has been a topic of ongoing research. Visualization tools have been the go-to for many years to be able to represent the learning process of a model and to interpret how and what do these models, with many layers and millions of weights, actually learn from the data. But so far, most of such tools have focused on smaller models and relatively small datasets. But as deep neural networks are becoming more and more prominent in industry level settings, there has been a need to understand these large scale industry level models with very large datasets and complex model structures. To this end, in this seminar, two such tools are presented. These tools, ACTIVIS[3] and SUMMIT[2], have their primary focus on large scale models and datasets but are designed for different kinds of data and tasks. **ACTIVIS[3]**, which is an interactive visualization system to represent industry-level deep neural network models, has been designed at Facebook and deployed on Facebook's machine learning platform. This is designed taking into consideration 15 different machine-learning users at Facebook and combines the results of both instance-level and subset-level model exploration. The second tool **SUMMIT[2]** is an open-source interactive system to analyse and scalably summarize the neuron activations and influences of features on the predictions made by deep neural networks. It uses activation aggregation and neuron-influence aggregation as the new scalable summarization techniques. In this seminar we go deep into the process of how these tools work. In later sections we also discuss the benefits of one tool over another and then see which tool suits which kind of scenario better. We shall also look at the future work in this field as presented in the respective papers. By the end of this seminar we shall be able to understand briefly the inner workings of these tools and make an informed decision on how these tools could help deep neural network users of large-scale, industry-based models in understanding and improving their previously mostly black-box based approach to deep neural networks.

1. Introduction

Machine Learning, Artificial Intelligence and Neural Networks are not new ideas and have been around for decades. But Deep Learning is a rather recent concept. Yet, it has become a wide spread technique because of its success in many different domains such as computer vision, natural language processing, signal processing, pose estimation and many more. This has led to a rise in interest in deep learning.

Despite so many people investing and working on deep learning, its internal working has remained elusive to most people. There are a few reasons for this. Firstly, deep learning as the name suggests is very deep, that is, has a lot of hidden layers as we are trying to replicate the neural network of a human brain. Hence different layers learn the features differently. Trying to analyze how that happens is difficult. Secondly, people do not usually write their own models for deep learning but tend to use pre-existing models as a black box. Hence you only get the desired results. But if the results are not satisfactory, it is difficult to understand the cause for it or how to fix it. This has led to an increase in the demand for developing visual tools to help developers and users interpret deep-learning models. But taking into account the complexity and variety of models, many tools are unable to provide desired insight into the models inner workings. Moreover, with more and more problems being solved using deep learning, the size of models keep on growing as well as these models are being deployed in industrial level large scale models with very large and varied datasets. Thus, basic visual tools are unable to tackle these issues as well as do not provide a general approach to handle any kind of models.

The tools that we shall look at in this seminar report are targeted specifically to solve these issues. These tools are flexible amd generalizable to the wide variety of models and datasets that companies use for many products and services. Also, these tools go beyond just the results to interpret a model which many other previous tools do.

ACTIVIS[3] is the first tool we shall look at in this seminar. This is a tool designed by Facebook to primarily visualize the internal workings of models within the Facebook AI or any external model. The major contributions of this tool are:

- A novel visualization technique that combines instance and subset-level inspections of neuron activations. This facilitates flexible comparison of neuron activation patterns for specific instances as well as specific custom defined subsets.

- Interface to unite graph representation as well as local inspection of complex models, allowing

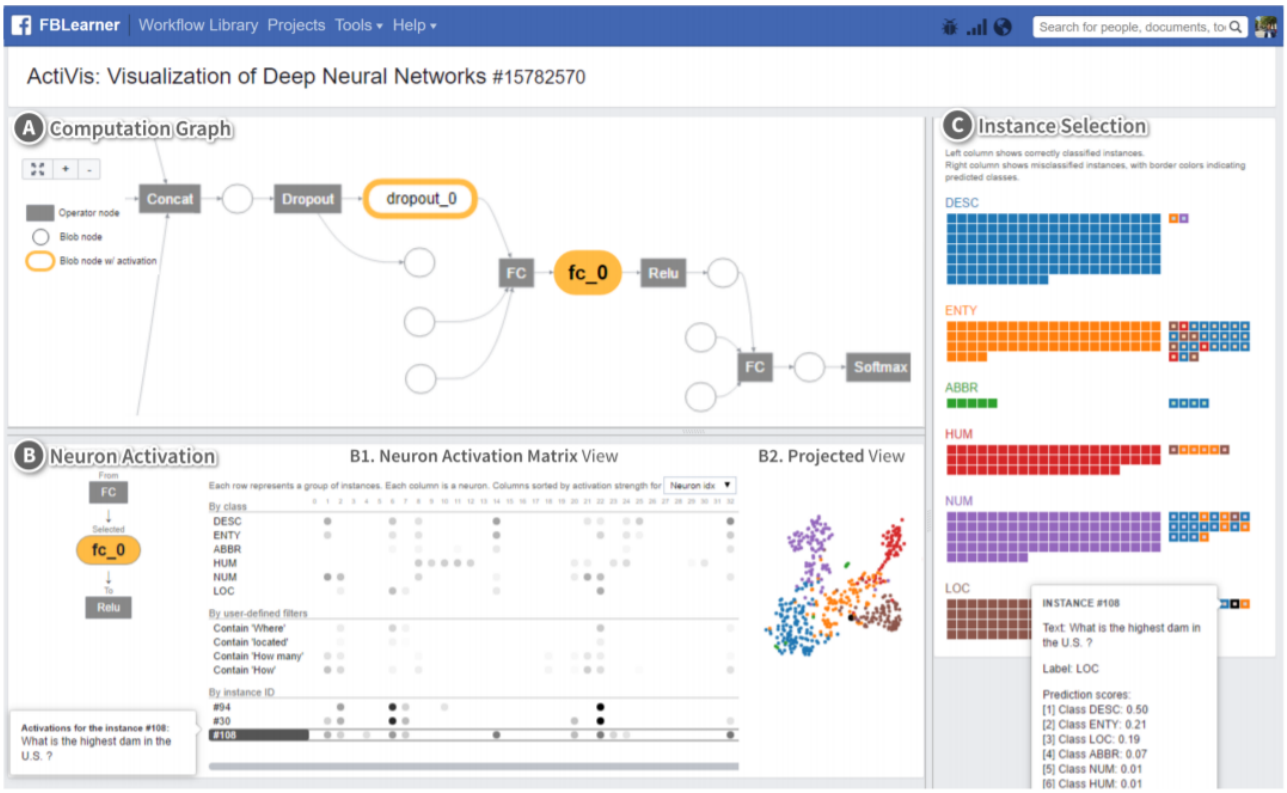

Figure 1: ACTIVIS[3] integrates multiple coordinated views. A. The computation graph summarizes the model architecture. B. The neuron activation panel's matrix view displays activations for instances, subsets, and classes (at B1), and its projected view shows a 2-D t-SNE projection[11] of the instance activations (at B2). C. The instance selection panel displays instances and their classification results; correctly classified instances shown on the left, misclassified on the right. Clicking an instance adds it to the neuron activation matrix view.

exploration of models at various levels of abstraction.

- Deployable model on large scale datasets and models.
- Facilitates model visualization within FBLearner[1] without and additional code and any external model with few lines of code.
- Case studies within Facebook with engineers as well as data scientists highlighting how ACTIVIS[3] helps them in their work.

The next model in our seminar is **SUMMIT[2]** which is an open-source tool to be deployed in any deep learning model by scalably and systematically summarizing what features a deep learning model has learned and how they help in making predictions. SUMMIT[2] gives the following contributions:

- An interactive system for scalable summarization and interpretation to explore learned classes in large scale models.
- Two new scalable summarization techniques namely *activation aggregation* to identify important neurons and *neuron-influence aggregation* to identify relationship between neurons.
- Graphical representation of the above mentioned techniques using attribution graphs. By using graphs the tool opens up the abundant graph algorithms to draw more insights into the model.
- An open-source, web-based implementation making it easier and accessible without many installation steps or any resources.

In this report, we shall look at these contributions in detail. We shall look at certain case studies done using these tools to evaluate their effectiveness. This report will show the interface features of these tools to see how they are implemented in real life. Also, we will compare and contrast these tools to see which is better and more suitable for which kind of model.

2. Methods

In this seminar, two visual tools have been covered. Both these methods use multiple strategies to visualize deep neural networks and combine these strategies to form an overall idea of the model in question. They handle multiple large scale datasets and are suitable for small as well as large and industry scale models. So, their end goals are the same but their approach is different as well as there are other differences which we shall highlight in next sections. In this section, we shall look at and summarize the

design challenges, goals, strategies and the interface.

2.1 ACTIVIS[3]

To design a tool to interpret models, 15 machine learning engineers were interviewed at Facebook AI to come up with a strategy. The conclusion drawn from this was that visualization has to be done broadly in two ways.

Instance-level analysis: In this strategy, a single chosen instance would run through the model and its behavior would be noticed. This would be helpful only in cases where data of a single instance such as status update of a user on Facebook. But this would not work where there is abstract data where the next strategy would work.

Subset-level analysis: The above status is however not useful in all cases. Where there is an article containing some complex numerical data, just instance-level analysis would not produce any useful insight. Thus looking at subset of dataset by date, author or publisher would make more sense.

2.1.1 Design Challenges

To accommodate both the strategies presented within an environment of ACTIVIS[3], some design strategies has to be formed which is summarized in six design challenges which are grouped into three labels, Data, Model and Analytics.

1. **Diverse input sources and formats - DATA:** Many different data formats such as text and numerical features are used in different deep learning models. Moreover, there are many models which utilize many different data formats at a time.

2. **High data volume - DATA.** Most industrial models use large volumes of data.

3. **Complex model architecture - MODEL:** In industrial scenario, most models are much more complex with many hidden layers which are wide as well.

4. **Many different variety of models - MODEL:** If the tool has to be general in nature, it has to be able to analyze many different kinds of models without much change in code.

5. **Diverse subset definitions - ANALYTICS:** The process has to be able to analyze many kinds of subset definitions so that it can flexible to a user's selection based on the task.

6. **Simultaneous need for performing instance- and subset-level analysis - ANALYTICS:** To analyze a model, the instance-based and subset-level analysis has to be done at the same time in order to be able to interpret the patterns better.

2.1.2 Design Goals.

In this section, design goals are defined to address the design challenges of the previous section.

1. **Unifying instance- and subset-based analysis to facilitate comparison of multiple instance activations:** ACTIVIS[3] aims to be able to provide flexibility to choose the subset. Also, different subset divisions have to be compared in a view. At the same time, large datasets have to visualized as well.

2. **Tight integration of overview of model architecture and localized inspection of activations:** ACTIVIS[3] has to be able to provide a summary of the whole model as well as zoom into the complex models and analyze each component.

3. **Scaling to industry-scale datasets and models through flexible system design:** The system has to be flexible, modularized system that allows developers to analyze models with simple API functions, while keeping it scalable.

2.1.3 Visualization of neuron activations at subset level.

ACTIVIS[3] does instance-level visualization using inspirations from past models. But as we discussed in the previous sections, just instance-level visualization with hundreds of neurons in hundreds of layers is not going to make much sense. Instead, if we see activation patterns of multiple related instances, then the user will be able to derive some meaning out of it. To do this, ACTIVIS[3] allows users to define instance subsets. Then average activations are computed and average vectors of multiple subsets are placed together and compared.

A neuron activation matrix is generated for comparing multiple instances and instance-subsets. Here each row represents either an instance or an instance-subset and each column represents a neuron and the color represents the activation level. The darker the circle, the stronger the activation.

ACTIVIS[3] allows users to define instance subsets flexibly. It can be defined using many different properties. In the neuron activation matrix, the classes are shown as default subsets. But further more features can be defined for subsets.

There is also the feature of sorting the rows by activation levels to see if any pattern is revealed such as spotting instances that are positively correlated with their true class.

To be able to find any patterns, ACTIVIS[3] does a two dimensional projection of high-dimensional data using t-distributed stochastic neighbor embedding (t-SNE) of activation functions. An example of it is shown in the figure. This projected view complements the neuron activation matrix as hovering over any specific row highlights the related instances in the

projected view and clicking on specific instance in the view adds it to the matrix view as shown in the figure.

2.1.4 Interface.

To help users interactively specify where to start the exploration of models, the system interface is designed and developed as shown in Figure 1. The interface has multiple components in different panels, which are discussed below.

1. **Overview of Model Architecture:** A very common way to represent deep neural networks is a computation graph. The way it is presented in ACTIVIS[3] is shown in the Fig. 1(A) on the top panel. The direction of the flow of data is from left to right. Dark rectangle represents an operator and circle represents a tensor. Hovering on a node highlights the full name and clicking on it shows the corresponding activation in the activation panel.
2. **Activation for Selected Node:** Upon selection of node in the overview graph, the neuron activation panel on the bottom gets populated as seen in Fig. 1(B). It consists of three sub-panels: (a) The names of the selected node and its neighbors(Fig, 1(B1)), (b) the neuron activation matrix(Fig, 1(B2)), and (c) the projected view as discussed in the previous section(Fig, 1(B3)). Multiple node selection is also an option for the user to compare activation patterns at different nodes.
3. **Instance Selection:** This panel combines the dual feature of ACTIVIS[3]. In this section, the user gets an overview of instances through their results and compare the activation results by selecting multiple instances and show it in the neuron activation matrix. Each square represents an instance in the panel in the right side(Fig. 1(C)). Upon hovering on the instance, the basic information about the instance is displayed.

2.1.5 Deployment Strategies:

ACTIVIS[3] can be used for any model by adding only a few lines of code to their model to create some information to be used by ACTIVIS[3] during training. Once training is done, model exploration can be done with a link to ACTIVIS[3] to visualize and explore the model in the FBLearner[1] Flow interface. In this section, we see how ACTIVIS[3] is built and deployed on FBLearner[1].

- **Generalizing to Different Models and Data Types:** The data generation process is modularized using API functions for model developers to simply call it in the code to activate ACTIVIS[3] for their specific models. Also, there are some user-defined functions to specify how subsets are defined in ACTIVIS[3] for more abstract and unstructured data types.
- **Scaling to Large Data and Models:** Some key ideas to scale ACTIVIS[3] to Large Data and Models are as follows:
  - **Manual selection of interesting variable nodes:** Rather than generating activations for all variable nodes, the model developers specify their own default set of variable nodes.
  - **Assisted sampling and visual selection of instances:** A sample of 1000 instances is presented by default within the interface which was decided by interviewing Facebook engineers as a number which would meet their practical needs.
  - **Neuron activation matrix computation for large datasets:** To reduce the time complexity of calculating the activation matrix over large datasets, a scalable approach is defined where a matrix S is created mapping all instances to subsets mapping. After labels are generated for all instances, an activation matrix A is produced for each variable node. Then the neuron activation matrix is produced by multiplying S and A. Its time complexity is linear in the number of instances[3].

2.2 SUMMIT[2]

The main goal of SUMMIT[2] is to build an interactive visualization tool to understand how neural network build their hierarchical representation. SUMMIT[2] introduces two new scalable techniques, (1) *activation aggregation* and (2) *neuron-influence aggregation* which we will see in further sections. The outcomes of these techniques are combined to generate attribution graphs which provide more information and helps to interpret model better.

The work presented in this paper is demonstrated using a model called INCEPTIONV1[10]. Although a choice is made, the summarization techniques work for other models in other domains.

2.2.1 Design Challenges:

To implement the summarization techniques of SUMMIT[2], these five key design challenges have been identified.

1. **SCALABILITY - Scaling up explanations and representations to entire classes, and ultimately, datasets of images:** Most existing visualization tools visualize neuron activations for only a single image. This maybe useful for small datasets but to see how the neuron activation takes place for entire class, the tool has to be able to allow image explanations for multiple images together.

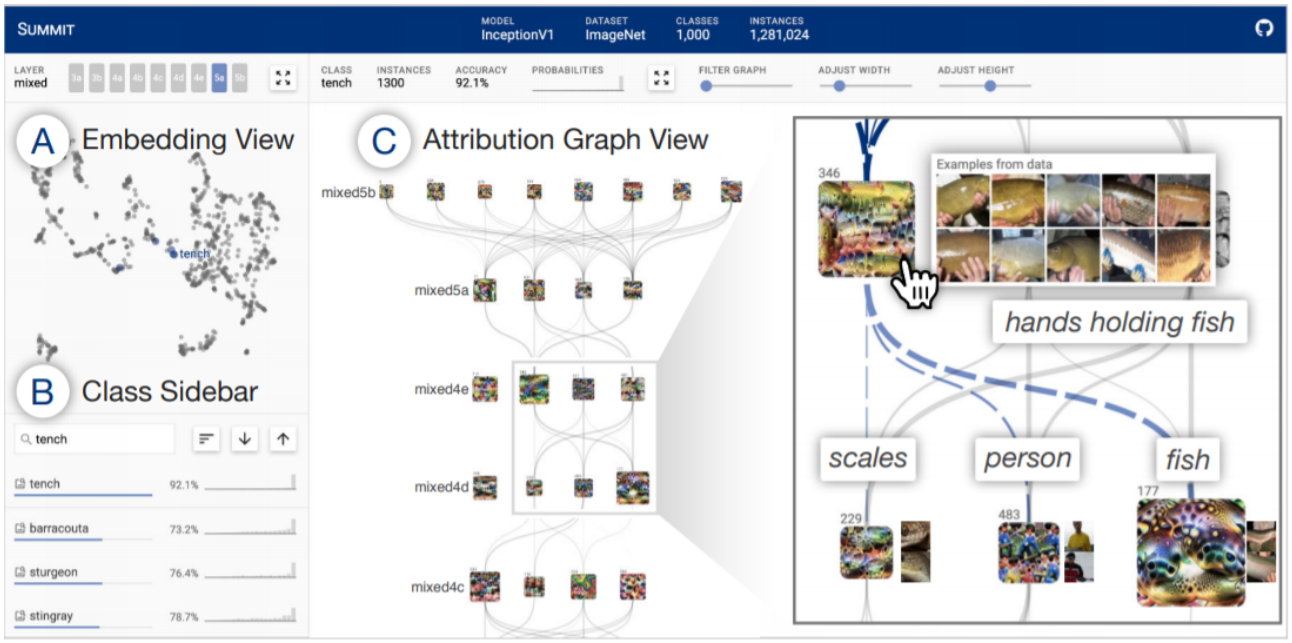

Figure 2: With SUMMIT[2], users can scalably summarize and interactively interpret deep neural networks by visualizing what features a network detects and how they are related. In this example, INCEPTIONV1[10] accurately classifies images of tench (yellow-brown fish). However, SUMMIT[2] reveals surprising associations in the network (e.g., using parts of people) that contribute to its final outcome: the "tench" prediction is dependent on an intermediate "hands holding fish" feature (right callout), which is influenced by lower-level features like "scales," "person," and "fish". (A) Embedding View summarizes all classes' aggregated activations using dimensionality reduction. (B) Class Sidebar enables users to search, sort, and compare all classes within a model. (C) Attribution Graph is used to visualize highly activated neurons as vertices(nodes) and their most influential connections as edges(dashed purple edges).

2. **INFLUENCE - Discovering influential connections in a network that most represents a learned class:** In dense convolutional neural networks, it is very important to determine the effect of a single convolutional filter's effect on later layers.

3. **VISUALIZATION - Synthesizing meaningful, interpretable visualizations with important channels and influential connections:** Just knowing which neurons are activated within different layers of convolution is not going to help unless the tool is able to combine all these observations into a meaningful holistic representation of the learning of the model would allow to interpret the model much better.

4. **INTERACTION - Interactive exploration of hundreds of learned class representations in a model:** Visualizing an overview is the challenge presented in the first point but in industrial and large-scale models, there are hundreds of classes which need to be visualized, so it is a challenge to present an interactive method to the user to be able to drill drown to classes as required.

5. **ACCESSIBILITY - Lower accessibility to understanding large scale neural network models:** Is it possible to provide an easy way for users to understand and interpret deep learning models without having the deep knowledge required to understand and train deep learning models?

2.2.2 Design Goals:

Identified challenges in the previous section, leads to the following five design goals for SUMMIT[2] to handle these challenges.

1. **Aggregating activations by counting top activated channels:** SUMMIT[2] aims to be able to identify the channels which have the strongest activation for any given image. This information can be then aggregated together to be able to understand which channel is most commonly activates to understand the impact of classes on neuron activations.

2. **Aggregating influences by counting previous top influential channels:** Aggregating top channels at each layer by neuron activation can help in understanding the impact of previous channels on future channels for each class.

3. **Points of interest for neural networks:** SUMMIT[2] aims to represent aggregated influences and activations in the form of graph to

be able to leverage abundant research in graph algorithms to analyze the model better.

4. **Designing of an interface to be able to visualize attribution graphs:** The goal of SUMMIT[2] is to support users to easily interact with the interface to select entire class from the neural network to check how predictions for a class are made by learning features.

5. **Deploying the tool on a lightweight universal platform:** The target of SUMMIT[2] is to make it feasible for any developer to use to interpret models without much need for computational resources by making it deployable on modern web browsers.

2.2.3 Creating Attribution Graphs by Aggregation:

The two new scalable summarization techniques introduced by SUMMIT[2] are (1) *activation aggregation* to discover important neurons, and (2) *neuron-influence aggregation* to identify relationships among such neurons, are combined to create a novel attribution graph to summarize crucial associations and substructures for model predictions. In the points below, we will see how these techniques are combined.

1. **Aggregating Neural Network Activations:** The method for aggregating neural network activations consists of the following steps:

   - Compute activation channel maximums for all images.
   - Filter by a particular class.
   - Aggregation Method 1: taking top k channels.
   - Aggregation Method 2: taking top k% of channels by weight.

   These steps generate an Activation Matrix to be later used.

2. **Aggregating Inter-layer Influences:** This process is done by taking the activations from previous layer and then convolving on them using a convolution kernel from the next layer. This results in 2D activation maps. We take the max of these maps and then select top most influential channels from previous layer and aggregate them into an aggregated influence matrix.

3. **Combining Aggregated Activations and Influences to Generate Attribution Graphs:** Then comes the novel graph created by SUMMIT[2] called attribution graph. Considering graph as the primary visualization allows us to utilize the abundant graph algorithms to better represent the activations and influences. SUMMIT[2] uses a Personalized PageRank Algorithm[4, 8] to score the importance of vertices which represent channels and edges which represent influences. Thus the attribution graphs use the activation matrix from the neuron activations and aggregated influence matrix from Inter-layer influences to run the PageRank Algorithm[4, 8] for 100 iterations and compute the graphs using the PageRank values to highlight importance of channels and the influences they have on the prediction of that particular class.

2.2.4 Interface:

Using the design goals and the aggregation methods described in the previous sections, we shall look at different aspects as part of the interface of SUMMIT[2] as shown in Figure 2.

The header shows metadata regarding the model visualized by SUMMIT[2]. Here we look at the model name, number of classes, number of images and the dataset name. As we see in the Fig. 2, the paper demonstrates the capabilities of SUMMIT[2] using a image classification model called INCEPTIONV1[10] using ImageNet dataset with 1.2 million images and 1000 classes[9]. Apart from the header, there are three more sections in the interface. We shall have a brief look at these sections as following.

1. **Embedding View: Learned Class Overview:** The first view of SUMMIT[2] as seen in the figure is an Embedding View which is a dimensionality reduction overview of all the classes in a model(Fig. 2(A)). It uses UMAP, an nonlinear dimensionality reduction that better preserves global data structure better than other techniques such as t-SNE[11]. In this representation, each dot corresponds to one class and the spatial positioning representing their similarity. Panning and zooming in is allowed and zooming enough gives some information about the class which can be used to check similarity of classes.

2. **Class Sidebar: Searching and Sorting Classes:** In the next panel below the embedding view is the Class Sidebar which has a list of all the classes(Fig. 2(B)). The top row is the selected class and below it are classes sorted by their similarity to the selected class using a cosine similarity. These row display some statistics about the class. The purple bar indicates how similar it is to the selected class. The histogram shows how well the probability of all predictions of the class. This allows the users to see if the model is biased towards any class or vice versa. There is also the facility of scrolling for similarity context and sorting and selecting of classes to display the attribution graph in the next panel.

3. **Attribution Graph View: Visual Class Summarization:** Attribution graph is the novel graph which is the primary view of SUMMIT[2] (Fig.

2(C)). Its header contains the similar information as the Class Sidebar and has a few controls to interact with it as described below.

- **Visualization strategy for attribution graphs:** SUMMIT[2] is inspired by other visualization tools such as CNNVis[6], AEVis[5] and Building blocks[7], to use graph-based representation for interpreting deep learning models. Here, the top row is the last mixed network layer and the bottom one is the first mixed network layer of the model. The size of the nodes represents the number of images within the class where this channel is the top-most. The edges shows the flow of data in the model and the thickness represents the influence of the channel over others.
- **Understanding attribution graphs:** The attribution graph is a novel approach to SUMMIT[2] which represents how classes behave inside a model. In some classes, we see that features are learned in the early layers itself, hence the node sizes are bigger in the lower layers where as in some cases, it needs many layers to become prominent features about the class. This also helps to understand how the model behaves overall by looking at how the node sizes increase as we go higher in the graph, i.e., as we go further deep into the model, mode complex features are learned and the classes become more distinguishable.
- **Inspecting channels and connections in attribution graphs:** Just visualizing the channels is not enough. So, SUMMIT[2] provides a few more interactions to the attribution graph to understand it better. Hovering over the channels highlights the path data takes to pass through this channel. Also, a window is also displayed beside it showing what features are learned from a few sample images. These and some more interactions, provide an in-depth idea about all the influences of channels on the predictions.
- **Dynamic drill down and filtering:** Attribution graph also supports panning and zooming in to filter out channels and connections of interest which assists the user to isolate the channels and understand their importance. The user experience within this panel is also meant to be able to navigate within views very easily.

3. Results

As this seminar covers visualization tools, for results, we are going to look at how these papers covered case studies to analyze the impact these tools had on the interpretation of models for developers and users.

3.1 ACTIVIS[3]

For case studies, three different types of users, an expert in natural language processing, a new software engineering working with natural language processing and a data scientist in 60 minute sessions and the audio was recorded for later observation and analysis. The findings present that, ACTIVIS[3] is effective for data scientists, experts and new developers. However, there is some scope of improvement which is presented as future work.

3.2 SUMMIT[2]

In the case studies section of this tool, many scenarios were taken into consideration and explained using examples on the model INCEPTIONV1[10] on the ImageNet dataset[9] and it is highlighted how certain problems could be diagnosed using SUMMIT[2] which would other wise be very difficult to find the source of. The title of these scenarios are listed below and are somewhat self explanatory.

1. **Unexpected Semantics within a Class.**
2. **Mixed Class Association Throughout Layers.**
3. **Discriminable Features in Similar Classes.**
4. **Finding Non-semantic Channels.**
5. **Informing Future Algorithm Design.**

4. Discussion and Future work

In this section, we shall look at the advantages and future work for each of the two tools discussed in this seminar. We shall look at the places where these tools could be best utilised. Also, we shall look at the primary differences between the tools.

4.1 Advantages and Future work for ACTIVIS[3]:

1. ACTIVIS[3] performs really well in understanding how the model learns by visualizing activations but this could be modified to visualize gradients by replacing the activations with gradient values. This would help developers to identify the specific neurons which lead to the model not performing well.
2. As ACTIVIS[3] has the feature of pre-defining the subsets, it is very fast. But for future work, the model would need real-time subset defining so as to make the filtering process fast.
3. Right now ACTIVIS[3] provides an initial preselection of 1000 instances which makes it easier for the user to be able to make their selections which users have actually found useful. But, a similar pre-selection of subsets would be assist the users even better. This selection could be based on heuristics or measures.

4. ACTIVIS[3] would become even more generalized if it could support input-dependent models, i.e., the number of neurons is dependent on the input.

4.2 Advantages and Future work for SUMMIT[2]:

1. The attribution graph is very useful but future work with SUMMIT could include multiple attribution graphs to have a comparison between classes using graph comparisons.

2. SUMMIT does a great job to adapting to simpler CNN models but it needs to be made adaptive to more complex models such as ResNets. After this, SUMMIT could also be upgraded further to include other types of models such as Generative Adversial Networks(GANs).

3. The attribution graph generation is done through PageRank[4, 8] which is a simple yet effective method. But other methods could be used to generate the attribution graph which would increase the value of the graph.

4.3 Similarities and Differences between ACTIVIS[3] and SUMMIT[2]:

- **Similarities:** As both the tools are for visualization of neural networks, there are a few similarities between the tools.

    1. Both the tools use neuron activations to visualize the model.
    2. Graph is used as an important tool for visualization albeit in different steps of the visualization
    3. Interaction is a very important part of the interface for both the tools. Hover, Pan and zoom perform specific tasks. There is also the feature of sorting in both the tools to reveal some patterns.
    4. The idea for both the tools is to make model interpretation easy for both expert and novice users.
    5. There has been interaction sessions with experts to identify the usefulness of both the tools.
    6. Both the tools are adaptive for both small-scale and industry-scale models.
    7. There is only a small addition of code to make models adaptable for both the tools.

- **Differences:** However, here is where the similarities end. Now let us list the differences between the tools.

    1. ACTIVIS[3] is primarily designed for text based models and SUMMIT[2] is designed for Image-based CNN models.
    2. ACTIVIS[3] uses both instance-based and subset-based visualizations but SUMMIT[2] does only subset-based visualization.
    3. FBLearner[1] is the specific platform to run ACTIVIS[3] but SUMMIT[2] runs on any modern web-browser.
    4. ACTIVIS[3] uses graph to visualize the overview structure of the model, whereas SUMMIT[2] is primarily defined by the novel attribution graph.
    5. The subsets in SUMMIT[2] are mostly images belonging to the same class but ACTIVIS[3] provides a wider range of subsets such as by year.
    6. SUMMIT[2] does not provide very good tools to compare subsets, only at the class sidebar level(Fig. 2(B)). Whereas, ACTIVIS[3] provides a matrix view of each subset to compare them(Fig. 1(B1, C)).

5. Conclusion

There have been many tools to perform visualization of internal workings of deep learning models but not for industry level models. This seminar gives a comprehensive analysis of two visualization tools to visualize and interpret activations of neurons in a deep learning model. First we look at ACTIVIS[3] which is a tool developed at Facebook for mostly text analysis based models. It uses the principle of instance-based and subset-based visualization to increase interpretability of models. We also see SUMMIT[2] which is an open-source web based visualization tool which provides a scalable way to visualize and interpret deep neural networks for image-based models. SUMMIT[2] uses graphs as the primary visualization and provides many interactions with the interface to understand the model even better. This report covers the design steps followed in making these tools and the describes the interface of the tools. We see the deployment strategies for these tools. We also looked at the way case studies were conducted on the tools to analyze their effectiveness in real time environment for both expert level users and new users. We also look at the similarities and differences of the tools. We come to a conclusion that visualization of deep learning models goes beyond just research and development of minor models. They also play a deep role in understanding and interpreting industry-level real-time deep learning models where we have complex structures and many data points and classes where subset-analysis is a very effective way to understand how models learn of features and make predictions.

References

[1] J. Dunn. Introducing FBLearner Flow: Facebook's AI backbone. https://code.facebook.com/posts/1072626246134461/introducing-fblearner-flow-facebook-s-ai-backbone/, 2016. [Online; accessed 26-June-2017].

[2] Fred Hohman, Haekyu Park, Caleb Robinson, and Duen Horng Chau. Summit: Scaling deep learning interpretability by visualizing activation and attribution summarizations, 2019.

[3] Minsuk Kahng, Pierre Y. Andrews, Aditya Kalro, and Duen Horng Chau. Activis: Visual exploration of industry-scale deep neural network models, 2017.

[4] Amy Langville and Carl Meyer. A survey of eigenvector methods of web information retrieval. *SIAM Review*, 47, 01 2004.

[5] Mengchen Liu, Shixia Liu, Hang Su, Kelei Cao, and Jun Zhu. Analyzing the noise robustness of deep neural networks, 2018.

[6] Mengchen Liu, Jiaxin Shi, Zhen Li, Chongxuan Li, Jun Zhu, and Shixia Liu. Towards better analysis of deep convolutional neural networks, 2016.

[7] Christopher Olah, Arvind Satyanarayan, Ian Johnson, Shan Carter, Ludwig Schubert, Katherine Ye, and Alexander Mordvintsev. The building blocks of interpretability. *Distill*, 2018.

[8] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. In *WWW 1999*, 1999.

[9] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge, 2014.

[10] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions, 2014.

[11] Laurens van der Maaten and Geoffrey E. Hinton. Visualizing data using t-sne. 2008.