An Interactive 3D Visualization Tool to Analyze the Functionality of Convolutional Neural Networks

by

Farnoosh Fatemi Pour

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

University of Alberta

Abstract

Convolutional Neural Networks have outperformed traditional image processing approaches for many image classification tasks. However, despite achieving high classification accuracy on multiple datasets, these machine learning approaches are not transparent and act more like a black box. For this reason, explaining how they work and how they perform classifications is essential for research and education.

This lack of interpretability prevents students and beginners in the field from fully understanding the internal functionality of these networks. Moreover, it makes it hard for an expert to debug the models and enhance their performance. Lastly, not explaining how a model produces an output decreases the user's trust in such machine learning models, especially in medicine.

Visualizing the internal variables of convolutional networks enables us to understand how the model processes the input data and generates the output classifications. It also allows us to show how neural networks work by exploring the sensitivity of the model to small changes in the input and how the internal variables affect the results of the classification.

These models usually have multiple convolutional and fully-connected layers with hundreds or even thousands of neurons in each layer. Visualizing this large number of variables intuitively is challenging as it is computationally expensive and hard to display with limited screen resolution.

This thesis proposes a novel interactive tool to visualize convolutional neural networks in using 3D graphics. The first part of our method partitions

the feature maps into clusters instead of showing all feature maps simultaneously. We use a K-Means clustering algorithm to cluster the feature maps and then use principal component analysis to represent each cluster's parameters. The tool also allows the user to create modified versions of the original input by applying binary masks and helps them to analyze the impact of removing different regions of the initial input data.

We evaluate this new tool by surveying eight participants with diverse backgrounds and asking them multiple questions about its effectiveness and user experience. In addition, we ask participants to compare the proposed tool with another baseline visualization method. Our evaluation indicates that clustering the feature maps in each layer is an effective and helpful way for the user to understand how convolutional neural networks work.

Acknowledgements

I would like to express my thanks to my supervisor, Professor Pierre Boulanger, for his constant support and guidance throughout this project.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Problem Definition

Machine learning models play a critical role in today's technology, from self driving cars to health care. Deep learning models like neural networks are the latest effort to improve the accuracy of these classifiers and extend their applications. These machine learning models can use supervised and unsupervised algorithms to work with tabular, visual, textual, and graph data. Supervised algorithms assume that data are labelled where the input is associated with classifications performed by humans as a training set. Once the neural network is trained, it can then classify new data into the labels defined during training. Unsupervised learning works by automatically discovering patterns and information using unlabelled data that can then be labelled later by humans. The most common supervised algorithm uses deep neural network architectures with thousands of weights and multiple layers to classify the input data into the labels specified by the training set. However, in both cases, because of their inherent complexity, these neural networks are seen as a black-box model, making it hard to understand how they generates the classification output [38]. Visualizing the internal variables of deep neural networks could give us an insights on how the model processes the input data and generates the output classification. Visualizing their functionality can help us investigate the model's sensitivity to small input changes or how the internal variables affect the results of the classification. However, visualizing this large number of variables intuitively is challenging as it is computationally expensive and hard

to display with limited screen resolution [9], [32]. This thesis proposes a novel interactive tool to visualize convolutional neural networks using 3D graphics [31], [33] and a clustering approach.

1.2 Visualizing Convolutional Neural Networks

Convolutional Neural Networks (CNN) [22] are deep learning models that are designed to analyze images specifically. Such models benefit from local properties in the input images to extract representative features extracted by convolutional layers. Today's CNN architectures have several non-linear transformations layers [21], [22]. Visualizing the internal variables of a CNN model and its functionalities is helpful for both beginners and experts in deep learning. CNNs are usually the first deep learning model that students learn, and their complexities due to having multiple computational layers make it challenging to comprehend [50]. In addition, visualizing the network functionality can help students understand better the network [43], [50].

A good visualization tool for CNNs can provide means to customize the network's input and analyze the robustness of the network output. For example, the user can modify the values of some of the input pixels and see how the network output responds to such modification. This way, the user can build an intuition of how the network processes the input and predicts its output [43]. Understanding how CNN works could also helps experts optimize these networks in real-world applications. For example, allowing them to improve their performance by understanding the possible reasons why a network might fail [32] in some conditions. Furthermore, visualizing a CNN can help experts to analyze its training process [24], [36] or find its vulnerabilities [7], [23].

Neural networks are black-box by nature, and therefore, visualizing them is challenging. Here are the main challenges that must be addressed by a new tool:

1. **Size of the representation:** The number of neurons in CNNs is enormous, and humans can only process a limited amount of data and in-

formation at the same time [14], [30]. Therefore, visualizing all of the CNN's neurons prevent the user from exploring them effectively [25]. For example, the depth of the last layers of a VGG16 network [41] is 512 layers, which is too deep to be visualized effectively.

2. **Complexity:** Extracting meaningful information from the complex layers of a deep network is challenging. Each convolutional layer has tens or hundreds of filters and feature maps, and visualizing such complex data is challenging. Moreover, the input of each layer is the output of the previous non-linear layer, and layers are different in terms of abstraction level. Lastly, each layer creates its representation of the input data, and it is not possible to explain a deep network by its structure only [3].

Various factors contribute to the effectiveness of a neural network visualization tool. An effective tool should show the structure of the network [50] and help the user gain insight into the relationship between the input features and the model output [19]. The user needs to see an overview of the features of neurons and how the high-level features are created after the low-level features [25]. Since deep neural networks have a large number of neurons and layers, the limitations of the human perception for processing vast amounts of information should be considered [34]. An effective visualization tool should also engage users [13], [15] and let them experiment directly with their input and parameters to build their mental model of how the network works without coding [43]. It is also essential for the user to gain a general sense of the network's functionality and explore the input data individually. For example, an analyst might be most interested in a particular subset of the data, such as a specific type of disease rather than a particular patient [51].

1.3 A New Visualization Tool

In this thesis, we propose an interactive visualization tool to analyze the internal characteristics of CNNs for a given input image. The purpose of this tool is to help the user gain an understanding of how the CNN model processes the

input image and performs classification. We solely focus on image recognition tasks. Our tool consists of two main components, (1) a 3D visualization of CNN layers and (2) an interactive tool to explore how the network output is modified by input images.

1.3.1 3D Visualization of CNN Layers

Since CNN consists of a large number of neurons, visualizing the neuron's states and feature maps is challenging and in order to prevent cognitive overload, the information being shown must be simplified [30]. Gestalt principles [4] describe the way human minds organize visual information. According to this principle, related items should be grouped to be visualized more effectively. Our approach is to simplify the visualization of CNN layers by grouping similar feature maps. First, we cluster the feature maps of each layer and then use Principal Component Analysis (PCA) parameters to represent each cluster state. Using this approach each layer of the network is viewed as a cluster instead of showing all the feature maps. At anytime, the user can still see all the feature maps in each cluster on demand.

1.3.2 Visualization of Modified Input Images

Exploring the predictions of a CNN on different modifications of the input image offers an effective way to understand how the black-box model [52] works. We let the user explore the network's behaviour on modified versions of the input image to give insights into how the network works. We use multiple binary masks to generate these modified versions [52]. After creating the modified images, we calculate the label probabilities in CNN output. We then select the top-four classes with the highest probabilities and visualize the altered images in the 3D display space according to their label probabilities. This visualization enables a comparison of the network output when different regions of the original image are masked. This way, the user can understand the impact of each layer on the outputs of the CNN model. Our visualization approach also uses a binary classification model (SVMs [6] are used for simplicity) to find the separation border of modified images that are recognized with a label the

same as the original image and the ones that are identified with a different label. The SVM model demonstrate the critical regions that have a direct impact on the decision of the CNN model. In addition to the predefined masks for creating modified images, we also allow the user to define their masks and altered images to investigate the impact on the CNN output.

1.4 Thesis Contributions

The contributions of this thesis are as follows:

1. We propose a novel tool to visualize the convolutional layers of CNNs using feature maps clustering and the first PCA component of each cluster.

2. We extend the capabilities of our interactive visualization tool to study the impact of each region of the input image in the decision of CNN models using predefined and user-provided masks for the input image.

3. We evaluate the tool by interviewing eight participants with diverse backgrounds to ensure the usability of our method and to find out the advantages and weaknesses of our approach compared to alternatives.

1.5 Thesis Structure

The rest of the thesis is organized as follows. We first review how other researchers have dealt with this problem in Chapter 2. Chapter 3 introduces how the proposed interactive visualization tool was designed. Chapter 4 describes how the interface was evaluated and analyze the results of the experiments for eight participants. Finally, we discuss the implications of our work and future steps in Chapter 5.

Chapter 2

Related Work

2.1 Explaining Deep Neural Networks

We first summarize the existing techniques for interpreting how Deep Neural Networks work. One of the earliest attempts in visualizing and explaining how neural networks work was Activation Maximization (AM) proposed in 2009 by Erhan et al. [10]. This approach aims at finding an artificially generated input that maximizes the activation of a particular unit in the network. This is done by keeping all the network parameters fixed and then running an optimization algorithm that generates an input image that maximizes the network activation values. The generated input image can be visualized as the ideal input for any neural network unit. Some approaches try to use the network's activation values and create a heatmap on the input image that the network uses to generate its output. Using deconvolutional neural networks was proposed by Zeiler et al. [52]–[54]. This method uses deconvolutional layers and unspooling layers to reverse the convolution and pooling operations of a CNN. This method projects the feature map of a given layer back to the input image dimension. It highlights parts of the input image that contributed to the activation of a given layer. Class Activation Maps (CAM) [55], calculates a map for each class using the feature maps of the last convolutional layer and up-samples each map to the size of the input image to produce the CAM. The result then displays how much each spatial region of the input image contributed to the network's output. Layer-Wise Relevance Propagation (LRP) [1], identifies the importance of each pixel with a backward pass in the neural network. Then,

it uses the activation values of the last layer to assign a relevance score to each neuron, propagates the scores back through the layers up to the input images, and calculates a relevance score for each pixel. The earlier methods explain how the network generates its input, but they have drawbacks. First, they fail to show the internal process of a CNN and second, the explanation they provide is for experts to understand and diagnose the network. For beginners, they need to get more information on how the network uses its connections to calculate the output and the step-by-step process of extracting features from the input and moving through the network to generate output.

Perturbation methods gain insight into the CNN calculation by comparing the network output for a specific image with a modified version. Analyzing the change in the network output when the input is altered can help calculate the feature relevance. To find the sensitivity of a network classification to different regions of an input image, Fong et al. [12] repeatedly cover-up square areas of the input image and compare the network outputs. They define three ways to delete information from the input image: (1) blur, blurring the area, (2) constant, using a mask with constant values on the area, and (3) noise, adding noise to the area. By observing the change in behaviour of the model, they find regions in the input that are sensitive to the model output. Zintgraf et al. [56] consider an input image region as important in the network output for a specific class if (1) removing it makes a big difference in the output, and (2) it is independent of its neighbouring area. This means that if an image region does not have much influence on the network output or it can be predicted from its surrounding pixels, the relevance value of that region will be low. Therefore, they observe the average effect of removing an input image region over all the input images.

In this thesis, we apply multiple masks on the input image and visualize its effect on the network output. We use a sliding window with multiple sizes to create various masks. We then select several images that hugely change the network output. The user can click on each of these images to see the inner network values. We also choose the four top-rated classes among the selected images and visualize the images with how much they belong to each of the

four classes. This way, the user can observe the network behaviour change in the neighbourhood of the image.

Another way to interpret a black-box model is to use a surrogate model [51]. In this approach, an interpretable model, such as a decision tree or linear classifier, is trained to mimic the behaviour of the black-box model. This distilled model then offers explanations for the classifications of the original model. Although the distilled model may not reveal much information about the decision process, it can shed some light on the features and their relations that potentially led to a model's final output [51].

Model distillation methods can be categorized into two groups: (1) Model Translation (MT) and (2) Local Approximation (LA) [51]. MT methods train a simpler model on the entire dataset using the input/output set of the trained model. The new model can be further analyzed to help explain the original model's computation [51]. The assumption behind LA methods is that the model has a linear discrimination behaviour in each small local space of the input [51]. Therefore, they use multiple simple models to explain the original model in a local area of the input space. This thesis focuses on the LA methods since we aim to visualize the network in the neighbourhood of a specific input.

One of the visualization methods based on LA is Local Interpretable Model-agnostic Explanations (LIME) proposed by Ribeiro et al. [37]. The explanation that LIME generates is defined to find an explainable model to approximate the original model in a neighbourhood of the input image with a minimum loss while being as low complex as possible. Furthermore, LIME is model-agnostic, which means it makes no assumptions about the function being interpreted [37].

This thesis uses SVM and trains it with the previously generated masked images to draw a line between the images that cause the network to generate different outputs. We mark the images that are the support vectors of the trained SVM in our visualization to let the user know which images were critical for the surrogate SVM model to follow the local behaviour of the network.

2.2 Visual Analytics for Understanding Neural Networks

There are various Visual Analytics (VA) tools developed to help analyze deep learning models and their predictions. VA tools can be employed to visualize the structure and layers of neural networks and their learned features to help with understanding and interpreting them. For example, LSTMVis [46] visualizes the long short-term memory networks structure and inner state. Summit [17] visualizes the internal values of a CNN and its structure. GANVis [49] shows the learned features of a trained generative adversarial network.

VA can also be used to visualize the training process and provide interactive experimentation and exploration of the network. DGMTracker [24] helps with understanding the CNNs, and DeepEyes [36] progressively shows the activations and values of GANs during training epochs to support the experts with the design of the network architecture. These analytical tools are helpful for experts to analyze their networks. But cannot help beginners to gain an intuition of how the network works quickly.

2.3 Visualization in Education of Deep Learning

Wang et al. [50] identify several challenges of deep learning education from interviews and surveys with instructors and students. CNN is mostly the first network taught in deep learning courses and its various computational layers make it complex for beginners to learn. Deep learning libraries and frameworks such as PyTorch [35] need prior knowledge of deep learning concepts. Therefore, the students cannot use these libraries to manipulate the network to understand the concepts. The students need to be able to learn about the network and build a mental model of how the network works without dealing with these challenges.

There are many attempts in providing a visualization tool that helps with deep learning education. These tools can visualize the network in a 2D screen,

3D screen, or in Virtual Reality (VR) space. A 2D and a 3D visualization output shows everything on a 2-dimensional screen. The difference is that the output of the 2D visualization has two dimensions, height and width, while the 3D visualization also uses the third dimension by creating a projective image. VR is a simulated environment that lets the user immerse in the 3D environment provided by stereo displays and 3D interactive wands. Multiple 2D visualizations of neural networks let the user view the network and its training process. TensorFlow playground [42] is an interactive environment for defining and training neural networks. Users can set the network structure parameters, including the number of hidden layers, the number of nodes in each hidden layer from a certain predefined range, and the network parameters such as learning rate, activation function, etc. The user then selects which dataset they want to use to train the network from the available two-dimensional datasets provided by the application. The values of the network parameters are shown in real-time during the training process. ConvNetJS [20] is a JavaScript library for visualizing the training process of neural networks. This library provides the network training statistics and layer information such as activations and weights during the training process. Furthermore, the user can select multiple deep neural networks such as CNN and Auto Encoders and design the network structure with coding in the UI. ReVACNN [5] shows the training process of a CNN in real-time together with a 2D embedding view of individual filters/nodes at a particular layer generated by t-SNE [27]. This 2D embedding view helps understand the relationships of filters in each layer and the redundancy of the filters' captured information. ActiVis [19] provides both instance- and subset-level visualizations for interpreting deep learning models and their outputs. ActiVis shows the computational graph of the network, which summarizes the model architecture and neuron activations matrix, for instance, a class or a subset of instances. The user can select to see both instance and subset-level visualizations. The average of the activation vectors for the user-defined subset of instances is shown for the subset-level visualizations. ActiVis uses t-SNE to show the projection of datapoints activation vectors in 2D space. Using t-SNE, points with similar activations are placed

closer to each other with high probability.

While 2D representations can effectively show smaller networks, the desktop size limitations make it challenging to visualize more extensive networks since they may not fit into the available screen space. Most interactive visualizations only depict fully-connected networks or only visualize too small networks to solve computer vision problems effectively. Since the network is large, all layers do not fit in the screen for showing deep networks, and the user should scroll the page to see all layers. In addition, when the number of filters/nodes is large, they may largely overlap when mapped to a 2D space, making the embeddings less informative.

3D representations take advantage of the third dimension to describe the aspects of the system. The third dimension allows for features such as rotation and visualization from multiple views [28].

TensorSpace.js [47] provides multiple "playgrounds" for exploring deep learning models. These playgrounds provide an interactive 3D visualization for a pre-trained deep learning model. The network layers are first shown as cubes, and the user can click on them to see the details of the layers and hover the mouse on the nodes to see their connections. The user can view the network from different angles and zoom levels in the 3D space controlled by mouse movements. The user can draw and feed their digit to the model via a drawing pad provided in the playground or select an image from the list to see the network prediction and layer activations.

A 3D interactive visualization of neural networks that are trained to classify 28×28 pixel images of handwritten digits is proposed by Harley [16]. This work aims to show what the network has learned and its behaviour with new input. It allows the user to interact with the network through a drawing pad to draw their digit by mouse and see the network output and activations. The visualization shows the node activation levels by colour. The user can hover over a node and see the edges leading to that node from the previous layer and their corresponding strengths. Clicking on a node reveals more detailed information about a node, such as its numerical input and output.

VR is a simulated environment that users can explore and interact with.

VR can benefit the visualization of neural networks. An example of using VR for developing deep learning models is VR4DL by VanHorn et al. [48]. This work provides an environment for biomedical professionals to select layers using their hands, construct a neural network structure, train it, and check its accuracy on a test set in real-time. In this framework, details of the layer activations are displayed as requested by the user. However, this visualization does not show the dense layers since they do not fit into the available space.

Bock et al. [2] employ VR and propose an interactive visualization of CNNs. This work starts by displaying the network as a cube with no detail. Then, the network details are shown based on how much the user approaches the network. If the user gets close enough, then details of the layers and activation values of the dense layer are also shown. The user can also apply any of the convolution operators in the network on test data and see the result to gain insights about the network operations.

Unlike the above educational tools, our proposed tool shows the network structure and inner states and provides some manipulated versions of the input image to help the user with their instance-based exploration. We apply various masks on the input image and select some that hugely change the network output, and show them in a 3D space for the user to see how much each image belongs to different classes. The user can then apply their mask and explore the network with their masked image. This helps the user get an initial intuition of what to explore and how to analyze the input image.

One can see in Table 2.1 a comparison of the various visualization tool and their classifications.

Table 2.1: Comparison of Neural Network Visualization Approaches

Approach	**Space**	**Support Customizing the Network**	**Support Customizing the Input**
TensorFlow playground [42]	2D	Yes - with limitation on number of hidden layers and neurons	No
ConvNetJS [20]	2D	Yes - with code	No
ReVACNN [5]	2D	Yes	No
ActiVis [19]	2D	No	No
TensorSpace.js [47]	3D	No	Yes - User can upload input image
Harley [16]	2D	No	Yes - User defines input by a drawing pad
VR4DL [48]	VR	Yes	Yes - User can upload input image
Bock et al. [2]	VR	No	No

Chapter 3

Proposed Method

In this chapter, we introduce our approach to visualize CNN as demonstrated in Figure 3.1. This tool has two main parts: (1) a tool to visualize convolutional layers (Section 3.1) and (2) a tool to modify the input image and visualize its effects on the outputs (Section 3.2). Our interactive visualization approach aims to handle large CNN and enable users to interact with the CNN with their own modified image as input. For example, for the input image x and its modified versions ($\tilde{x}_k$), our method visualizes the feature maps of the CNN model for x and $\tilde{x}_k$. In addition, it performs post-processing analysis to provide further insights into the CNN model using a clustering technique.

3.1 Part One: Visualization of Convolutional Layers

To visualize Convolutional layers, we first partition the feature maps using a clustering algorithm, and then generate a cluster representation that can then be viewed in 3D.

Feature maps clustering The reason that we partition the feature maps of each layer and only show one representative for each group is to simplify the visual clutter. We group the feature maps into clusters using Shannon entropy [40]. Shannon entropy is a measure of how much information is included in data such as random variables and distributions. It quantifies information to a number between 0 and 1. The entropy value is at maximum of 1

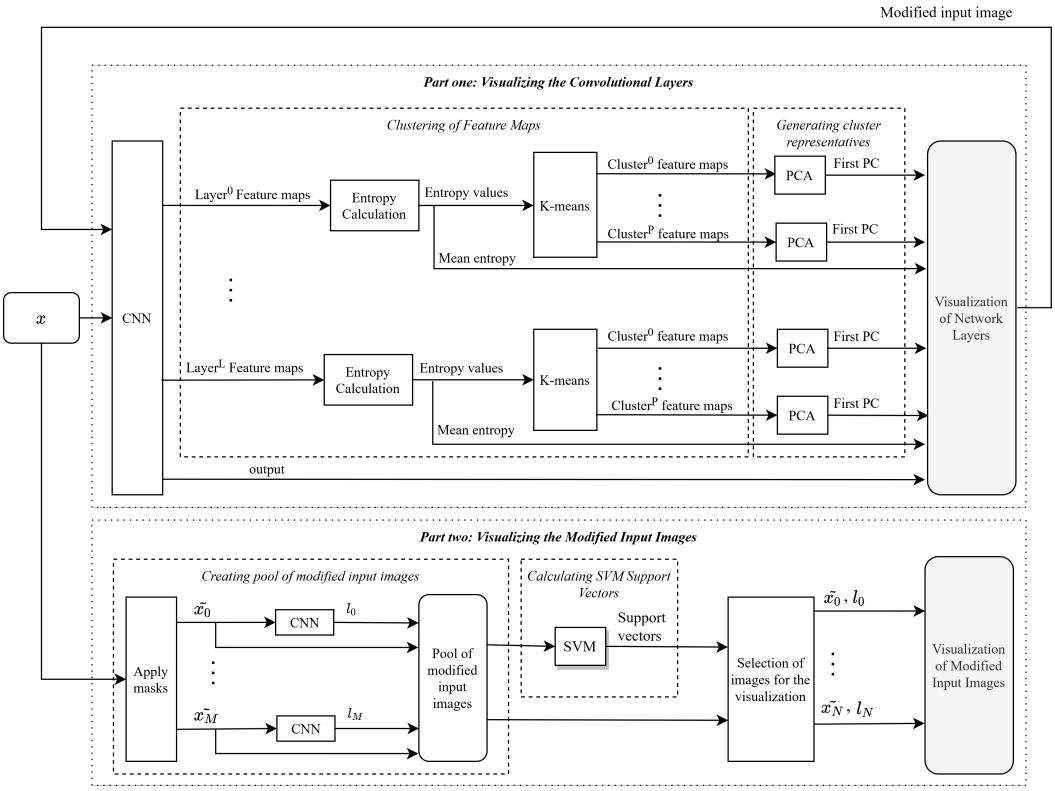

Figure 3.1: Our proposed method to visualize CNN

when randomness of the data is maximum. A higher value of entropy means less information in the data and lower entropy means that the data is more organized.

For simplicity, we use K-means [26] for partitioning the feature maps. K-means measures the similarity of points based on their Euclidean distances and assigns points to a cluster based on their similarity. We apply K-means on the entropy values of the feature maps. Therefore, we partition the feature maps with similar entropy values into the same group.

Generating cluster parametrization After clustering the feature maps, we generate a single parametrization for the feature maps that are in the same cluster using the first component of a Principle Component Analysis, PCA [18]. PCA is vastly used in literature for converting multi-channel images into single-channel e.g. RGB to grey conversion [39], [44]. PCA is a linear transformation that converts data from the original space to a more ordered sub-space. This

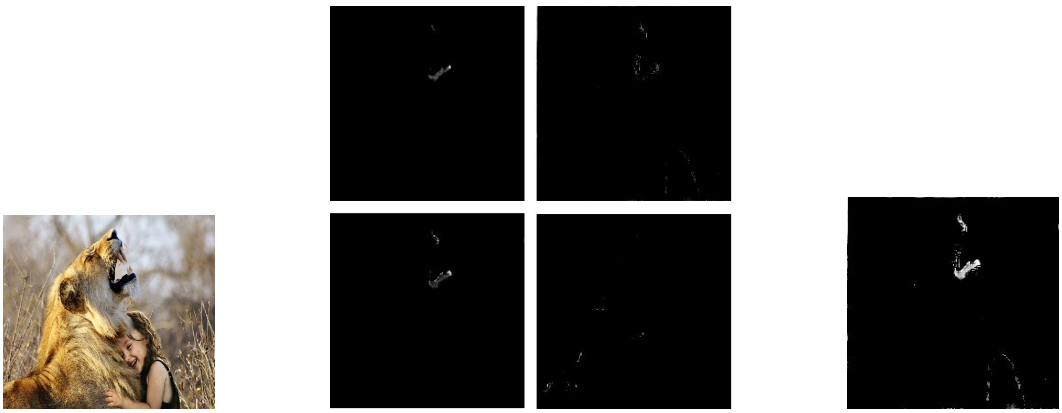

Figure 3.2: Representation of four sample feature maps (middle) of a sample image(left) with their first PC (right).

new sub-space usually has a lower number of dimensions compared to the original space. The First Principal Component (PC) of PCA is the dimension that explains the maximum variability in the data. In each cluster, we apply PCA on all feature maps within that cluster and use the first PC to represent the feature maps with a single image. Figure 3.2 is an illustrative example of how the first PC represents a set of feature maps. The figure shows 4 sample feature maps from the first layer of a VGG16 [41] model and their first PC. The first PC explains 54.8% of variability in feature maps.

We then visualize the following information for each layer:

1. Average entropy of each cluster of feature maps;
2. Representative image of each feature map cluster;
3. The percentage of variability of feature maps in each cluster that is explained by its representative.

Figure 3.3 shows examples of representations for the VGG16 CNN model. Each column visualizes a convolutional layer of the model from left to right. The right-most column shows the top 10 classes in the network's output. The feature maps are clustered based on their entropy to 3 clusters in each layer, and the first PC is shown as the cluster representative. The average entropy in each cluster and the amount of variability each PC explains is shown on top of each representative.

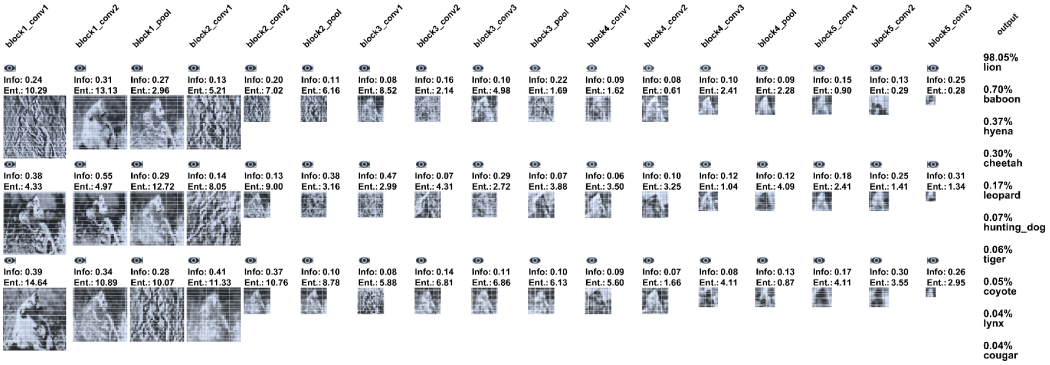

Figure 3.3: Visualization of the VGG16 model layers with the proposed visualization approach.

3.1.1 Definition of User Interactions

The user can interact with our visualization tool in three different ways.

- First, by hovering over each pixel, the user can track the corresponding pixels from the previous layer that are involved in calculations of that pixel (Figure 3.4).
- Second, by clicking on each cluster's representative, the user can see all the feature maps within that cluster (Figure 3.5).
- Third, the user can put a customized mask on the input image to hide some pixels and see the change in network layers and output for the new modified image.

As shown in Figure 3.1, we process the user's modified image in the same way we processed the original image: feed it into the CNN, group the feature maps of each layer, generate the representative feature maps, and update the visualized layers for the user. This way, the user can get insights into what regions of the original image impact the network's output more. For instance, the user can put a mask on a car's tires and see if the network still labels the input image as a car.

Figure 3.6 shows a sample scene where the user applied their customized mask on the image. The white rectangle on top center of the image shows the region that the user-defined mask removed from the image. The visual-

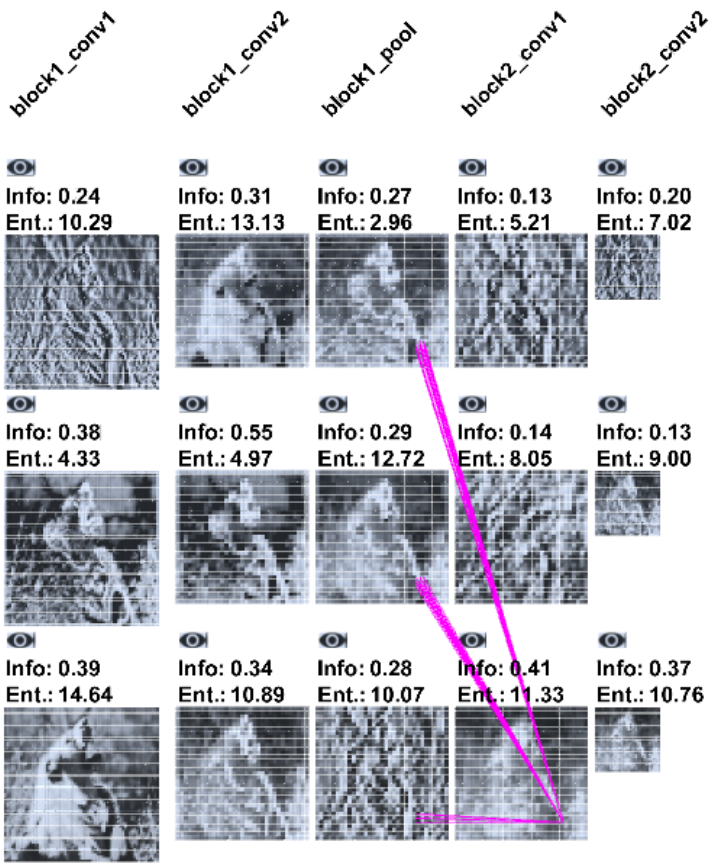

Figure 3.4: Tool to track the corresponding pixels from a previous layer.

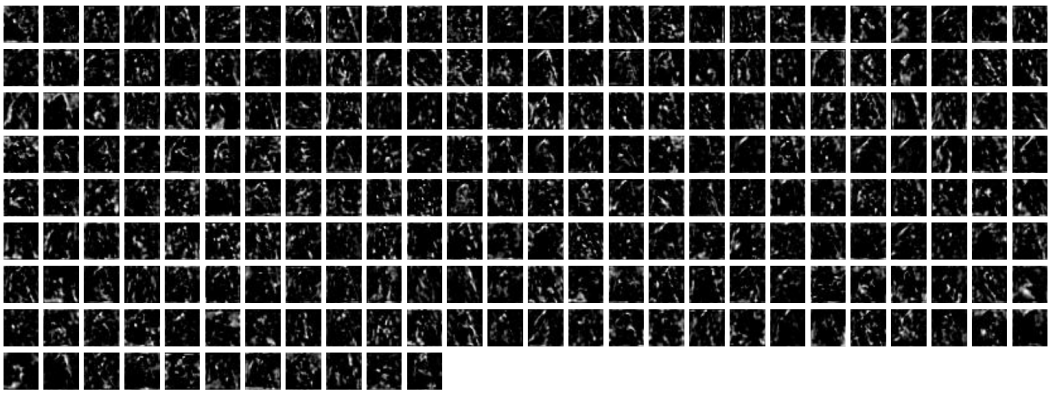

Figure 3.5: Visualization of all the feature maps of a cluster.

ization of the network layers for this user-defined modified image is shown in Figure 3.7. The user can generate multiple modified images. Our visualization shows all of these user-defined images to the user as shown in Figure 3.8. The

Figure 3.6: Visualization of a user-defined mask and removing the pixels that fall under the mask

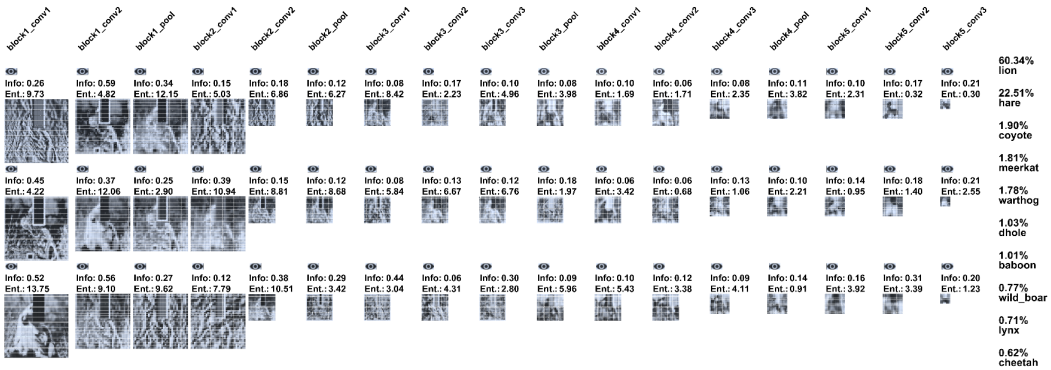

Figure 3.7: Visualization of the CNN model layers after the user defined a modified input image.

user can select any of these images and see the network layer visualization for it.

Figure 3.8: Visualization of various modifications of the input image using a mask.

3.2 Part Two: Visualization of the CNN Model based on Modified Images

Our tool creates a pool of modified versions of the original image and selects a subset of them to show to the user, as it is illustrated in Figure 3.1. We show these images in a 3D VR scene (Figure 3.9) to provide insights into what regions of the original image play a more important role in the decision of the CNN. In the following, we explain how we generate the pool of modified images, select a subset of them, and visualize this subset.

Assume x as the original input image of CNN model C and $l_x = C(x)$ as its output. Let $m^i, i = 1, ..., M$ be a set of binary masks where the shape of m^i is as same as x. Applying these M binary masks to the original image x by $x \odot m^i$ (element-wise multiplication) results in M modified images. These modified images $\tilde{x}_i, i = 1, ..., M$ are similar to the original image except for the masked regions that are in black colour (zero values). Formally, these modified images are in the neighborhood of the original image with radius r, $|x - \tilde{x}_i| \leq \varepsilon, i = 1, ..., M$.

To select a subset of modified images, we choose the top-four labels in the pool of modified images and select a subset of modified images. The subset represents each of these four labels equally.

After selecting a subset of modified images, we visualize them in a 3D scene to enable the user to compare the output of the CNN model for them. For this purpose, we first place the four selected classes in the corners of a 3D equilateral triangle (tetrahedron). Then we set the position of each of the selected modified images inside the tetrahedron according to label probabilities. In other words, the distances between label probabilities to each of the four

vertexes determine the position of each image. Figure 3.9 shows a visualization of the modified images in 3D space.

3.2.1 CNN Decision Boundary for Modified Images

After visualizing the modified images in a tetrahedron, we train a SVM [45] model to separate the correctly and incorrectly labeled modified images. We use the output of the last fully-connected layer for each modified image as the input data for SVM. If the CNN model labeled the modified image the same as the original image, we ask the SVM model to classify it as class 0, and if their labels are not the same, we consider class 1 as the label for training the SVM model.

After the SVM model is trained, we mark its results for the user in our visualization. Support Vectors are a subset of the input data that define the decision boundary of the SVM model. The SVM model separates the data of two classes with its Support Vectors. Therefore, the SVM model classification is sensitive to its Support Vectors and not the rest of the data. When a modified image is marked as a Support Vector in our trained SVM model, it means that modifying the mask applied to that image is more likely to change the output of the CNN model compared to other images. Marking the Support Vectors for the user helps them in finding the modified images (and their underlying masks) that are more sensitive to change from the CNN viewpoint.

Figure 3.9 shows an example of this visualization. As the figure shows, one of the images has a black border around it as an indicator that the CNN model's decision for this modified image is more likely to be sensitive to changes.

Figure 3.9: Visualization of modified images in 3D scene.

Chapter 4

Interface Evaluation

To evaluate our visualization tool, we have designed a set of experiments and a questionnaire form that will allow us to evaluate its effectiveness. The following hypotheses need to ve confirmed:

1. CNN visualization with grouping the feature maps of each layer is more informative than showing all feature maps simultaneously;

2. Letting the user apply customized masks on the original image is more insightful on how the network performs.

We start the experimentation with a comprehensive tutorial. Because of COVID 19 limitations, only eight participants performed three experiments and completed the questionnaire. We repeated each experiment three times using three different images. Here is the list of three experiments the users were asked to perform:

1. *Non-clustered visualization*: In the first experiment, we visualize the feature maps of a CNN without any clustering and ask participants to describe the information they gain from this visualization. As shown in Figure 4.1, we display the name of all CNN layers in order on the screen. The user could click on any layer to reveal the feature maps in that layer;

2. *Clustered visualization*: This is our proposed visualization tool in which we cluster the feature maps and also visualize the modified images on a tetrahedron based on their class probabilities;

3. *Reference visualization*: In this baseline experiment, we use CNN Explainer [50] to compare the information that we visualize for the user with another CNN visualization method. CNN Explainer is one of the recent efforts in visualizing CNNs. This interactive approach lets the user use their image as input. It visualizes how the image gets transformed in CNN layers. This approach shows the mathematical details behind each calculation in each layer on demand when the user clicks on the neurons and connections.

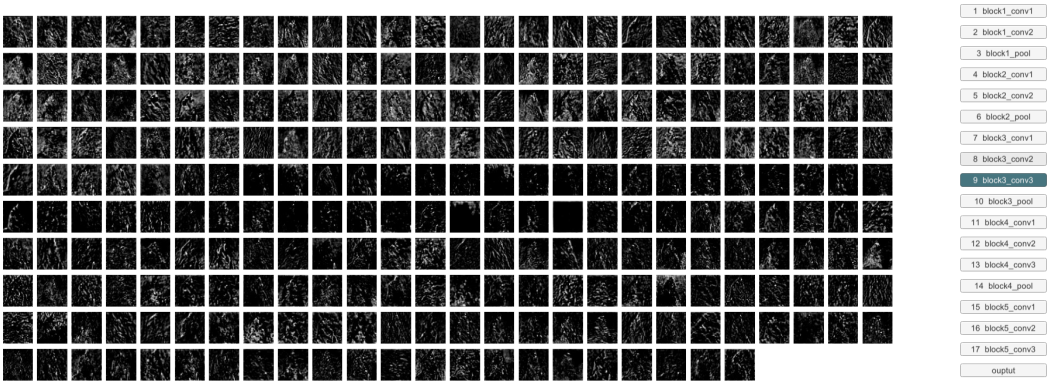

Figure 4.1: Visualization of VGG16 model layers without clustering feature maps.

We implemented the proposed visualization tool on a computer with the following configuration: the RAM is 32GB, the CPU is an AMD 5800X running at 3200 MHZ, and the graphic card is an RTX 3060 RGB. We used a VGG16 model trained on the ImageNet dataset [8] for both *Clustered visualization* and *Non-clustered visualization* visualizations. VGG stands for Visual Geometry Group, a research group at Oxford. This group introduced the VGG CNN architecture with 16 layers as VGG16 in 2014. We use TensorFlow [29] to load a pre-trained VGG16 model. ImageNet database is composed of over 14 million images with 1000 labels.

We implemented both *Clustered visualization* and *Non-clustered visualization* in the VR game engine Unity3D. Figure 4.2 shows the diagram of this implementation. The 3D visualization applications are consisted of two scenes:

1. Network structure: This scene visualizes the CNN model layers. This scene also provides the ability to modify the input image and update the visualized layers for the new image.

2. 3D visualization of masked images: This scene visualizes the modified images in the 3D space. The user can rotate this page and zoom in/out to explore the modified images and compare the CNN model output for them.

The user can switch between these two scenes at any time. In the main menu, the user selects an input image for visualization. The unity program consists of multiple components that retrieve the CNN model information and modified images and visualize them. The CNN model functionalities are implemented in python. The python scripts run along with the unity program and provide the data requested by the unity components. For example, whenever the user modifies the input image, the unity program sends the new modified image to the running python script to calculate the CNN model feature maps and output values for.

In the following paragraphs, we will describe the experimental setup and the questionnaires used and then present the experiment results and its analysis.

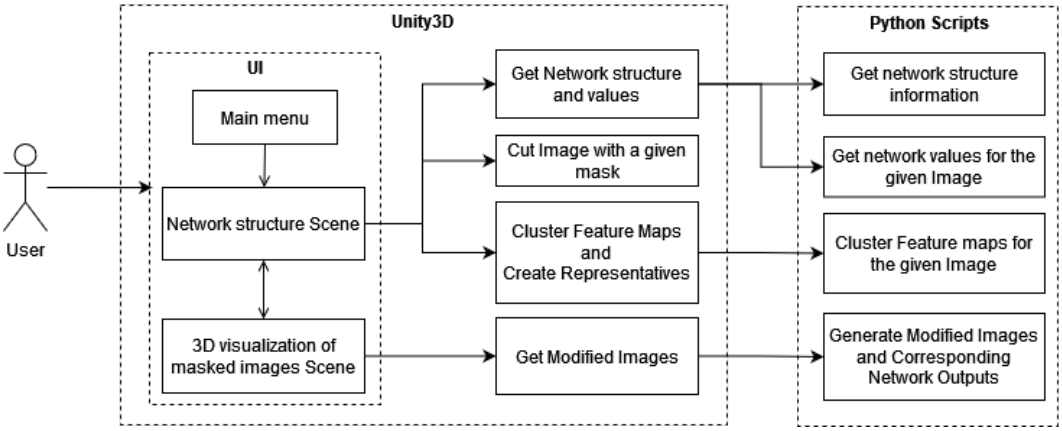

Figure 4.2: Block diagram of the Unity scene layout and back-end components

4.1 Limitations of the Experimental Study

The proposed visualization tool may benefit experts and non-experts differently. Based on the level of user's knowledge of deep neural networks, they use the visualization tool differently. For example, an expert may prefer to get more detailed information of the mathematical operations of the network compared to a user with no neural network knowledge. Therefore, analyzing the feedback that the expert and non-expert participants provide separately could give us more insight into the effectiveness of the app for each kind of user. However, because of the COVID-19 situation limitations, the number of participants who performed our study was small and this kind of detailed analysis was not possible.

4.2 Experiment Setup

Before starting the experiments, the participants fill out an information sheet that provides their demographic information, as shown in Table 4.1. Then the participants worked for 15 minutes with each of the three visualizations tools to get familiar with their functionalities. For the *Clustered visualization* and *Non-clustered visualization*, we provided tutorial panels that explain elements of the visualization tool. While going through the tutorials one by one, the participants had to also work with the tool and try each element to get familiar with it. After getting acquainted, the participants were asked to watch the video[1] that explains CNN Explainer and works with it with a sample image to get familiar with its functionalities. After the participants got familiarized with the three visualization experiments, we start the experiments. The participants work with three images shown in Figure 4.3, each with three different visualization approaches: We randomly change the order of the experiments for each image to mitigate the learning effect users may get from the first and second approaches. After the experiment, the participants had to fill out the questionnaire.

[1] https://www.youtube.com/watch?v=HnWIHWFbuUQ

Figure 4.3: Three images used for the experiments.

Table 4.1: Demographic information questions

No.	Question
1	What is your age?
2	What is your gender? (1) Male (2) Female (3) Prefer not to declare (3) Other
3	What is your background? If you choose other, please explain. (1) Mathematical sciences (2) Physical sciences (3) Biological Sciences (4) Other
4	How do you assess your knowledge level of neural networks? (1) I understand how neural networks work and I work with them regularly in my job/studies (2) I understand how neural networks work (3) I have some general knowledge of how neural networks work (4) I have no knowledge

4.2.1 The Questionnaire

Our questionnaire, shown in Table 4.2, consists of 10 questions in two sections. The first section evaluates the tool for ease of use, learnability, and intuitiveness. The questions in this section are closed with a certain number of pre-defined answers from strongly disagree to agree strongly. The second section asks open questions about the visualization tools. These questions are designed to assess how the visualization tool helped participants gain insights into the network functionality. It also evaluates how much pre-knowledge about neural networks is required to use the tool, which visualization tool

they preferred, why, and what new features they would like to be in a new version.

4.3 Participants Demographic Information

Our evaluation consists of eight participants, 75% female and 25% male aged between 25 to 40 years old. 50% have a background in Mathematical Sciences, 37.5% in Biological Sciences, and 12.5% in Social Sciences. 25% understand how neural networks work and regularly work with neural networks in their daily work. 37.5% assess their knowledge of neural networks as they have some general understanding of how neural networks work, 12.5% understand how neural networks work, and 25% have no knowledge.

4.4 Usage Analysis

In both *Clustered visualization* and *Non-clustered visualization*, we log the user interactions with the visualizations. We log the times that participants start or quit the app, apply a custom mask on the image, recalculate the network layers, or start any network layers or 3D modified images scenes. From this log, we analyze the usage of the visualizations tools by the participants. On average, each participant spent 19:39 minutes on tutorials to get familiar and 37:3 minutes on the experiments. From the available scenes in the apps, they spent on average 20:9 minutes on the network layers in the *Clustered visualization* and 11:39 minutes in *Non-clustered visualization*. In addition, they spent 6 minutes on average on the 3D modified images visualization tools for both approaches. They also defined on average 7.3 customized masks on the original images and updated the network layers visualization for them.

4.5 Evaluation Results

Tables 4.3, 4.4, and 4.5 show the number of answers we received for each multiple choice questions for the *Clustered visualization*, *Non-clustered visualization*, and the *Reference visualization* tools. In the following, we explain the

Table 4.2: The designed questionnaire for evaluating the apps

No.	Question	Assessment area
1	It was easy to learn how to work with the app	`Learnability`
2	The app was user-friendly	
3	App visualizations were intuitive	`Intuitiveness`
4	The information shown in the app had visual clarity	`Helpfulness`
5	The visualized information was useful to become familiar with the network's structure and decision making process (e.g. the network's structure, the flow of information through the network, the effect of change on the network output, etc.)	
6	Can you explain how you used the visualized information to extract insights on the network functionalities? Please share some of the insights you gained. (1) Allow me to understand how the NN compute the features used for classifications (2) Allow me to visualize the architecture of the network better (3) Allow me to understand what is the influence of the network's parameters on the results (4) Allow me to understand what parts of the input data matter for the network (5) Other: ...	
7	For each of the three images that you tried: (Please include the image name) What ratio of the areas in the input image that you think to matter the most for the classification tasks the network also paid attention to? Did the network flag any areas in the input images that you consider irrelevant for recognizing the objects? For example, highlighting a tree behind the cat for a cat image.	
8	For each of the three images that you tried: Do you think the size of the areas that the network highlighted for classification was correct? For example, highlighting half of a cat's eye means the size is smaller than needed.	
9	I think pre-knowledge about deep neural networks is required to understand the visualized information	`Learnability, Intuitiveness`
10	If you agreed to the previous question, what pre-knowledge do you think was necessary to understand the visualization?	
11	From the following scenes, which scene did you find more useful and why? (a) 3D visualization of masked images (b) Network structure	`Scene preference`
12	From the following scenes, which scene did you find more informative and why? (a) Showing clustered network layers (b) Showing full feature maps layer by layer	
13	What features do you like to be added to this app? Why?	`Improvements`

Table 4.3: Summarized results of the questionnaire multiple choice questions for the *Clustered visualization*. SD: Strongly Disagree, D: Disagree, N: Neutral, A: Agree, SA: Strongly Agree

No.	Question	SD	D	N	A	SA
1	It was easy to learn how to work with the app	0%	0%	0%	**50%**	**50%**
2	The app was user-friendly	0%	0%	0%	**62.5%**	37.5%
3	App visualizations were intuitive	0%	0%	0%	**50%**	**50%**
4	The information shown in the app had visual clarity	0%	0%	0%	**50%**	**50%**
5	The visualized information was useful to become familiar with the network's structure and decision making process	0%	0%	0%	**62.5%**	37.5%
6	I think pre-knowledge about deep neural networks is required to understand the visualized information	0%	**62.5%**	0%	12.5%	25%

results for each of the assessment areas.

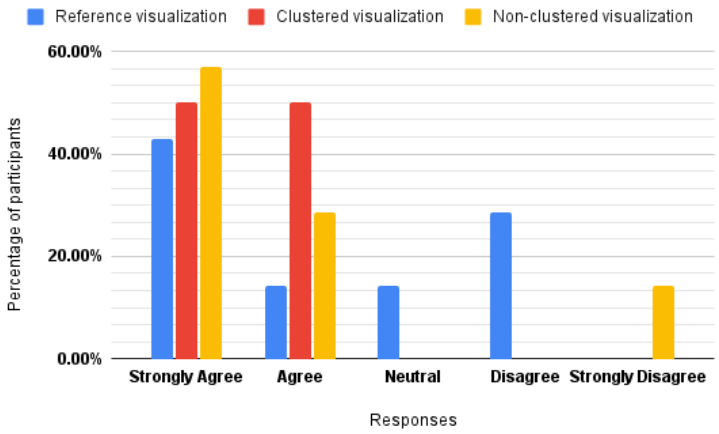

Figure 4.4: Comparison of participants responses to the question "it was easy to learn to work with the visualization tool"

4.5.1 Learnability and Intuitiveness

Ease of Use The first question in this category is about how easy it is to learn working with the application (Figure 4.4). Participants' answers demon-

Table 4.4: Summary of the results of the questionnaire multiple choice questions for the proposed app with non-clustered layers. SD: Strongly Disagree, D: Disagree, N: Neutral, A: Agree, SA: Strongly Agree

No.	Question	SD	D	N	A	SA
1	It was easy to learn how to work with the app	14.2%	0%	0%	28.5%	**57.1%**
2	The app was user-friendly	0%	14.2%	0%	**42.8%**	**42.8%**
3	App visualizations were intuitive	0%	28.5%	14.2%	**57.1%**	0%
4	The information shown in the app had visual clarity	0%	14.2%	14.2%	**57.1%**	14.2%
5	The visualized information was useful to become familiar with the network's structure and decision making process	0%	42.8%	0%	**57.1%**	0%
6	I think pre-knowledge about deep neural networks is required to understand the visualized information	0%	14.2%	14.2%	0%	**71.4%**

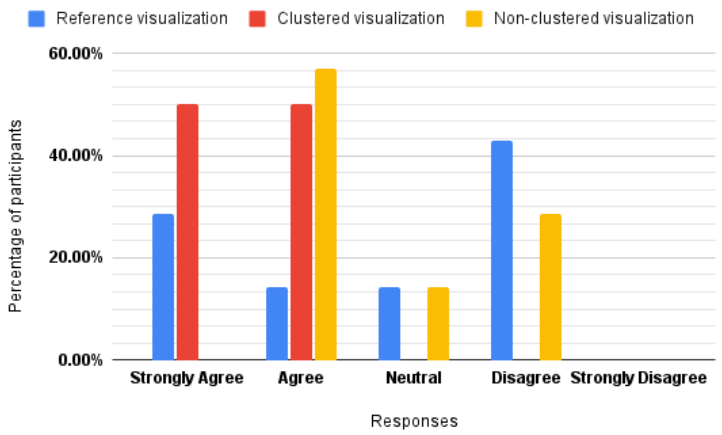

Figure 4.5: Comparison of participants responses the question "the visualization tool was intuitive"

strate that the application with clustered layers is easier to learn to work with (answered as Agree or Strongly Agree). When the feature maps are not clustered for the application, 85% of the participants specified that learning to

Table 4.5: Summary of the results of the questionnaire multiple choice questions for the baseline approach, CNN Explainer. SD: Strongly Disagree, D: Disagree, N: Neutral, A: Agree, SA: Strongly Agree

No.	Question	SD	D	N	A	SA
1	It was easy to learn how to work with the app	0%	28.5%	14.2%	14.2%	**42.8%**
2	The app was user-friendly	0%	**42.8%**	0%	14.2%	**42.8%**
3	App visualizations were intuitive	0%	**42.8%**	14.2%	14.2%	28.5%
4	The information shown in the app had visual clarity	0%	**57.1%**	0%	0%	42.8%
5	The visualized information was useful to become familiar with the network's structure and decision making process	0%	**42.8%**	0%	**42.8%**	14.2%
6	I think pre-knowledge about deep neural networks is required to understand the visualized information	0%	42.8%	0%	0%	**57.1%**

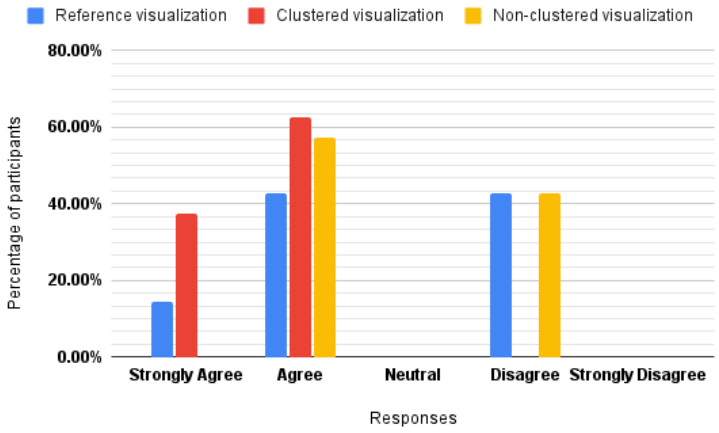

Figure 4.6: Comparison of participants responses the question "pre-knowledge is required to wok with the visualization tool"

work with the app was easy (answered as Agree or Strongly Agree). For the baseline application, 42% of the participants specified that learning to work with the app was easy (answered as Agree or Strongly agree) and 28% found it hard (answered as Disagree), with the remaining participants (14%), stay

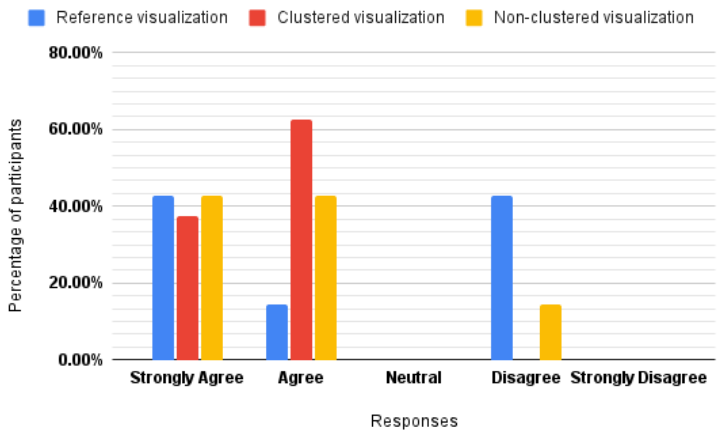

Figure 4.7: Comparison of participants responses the question "the visualization tool was user-friendly"

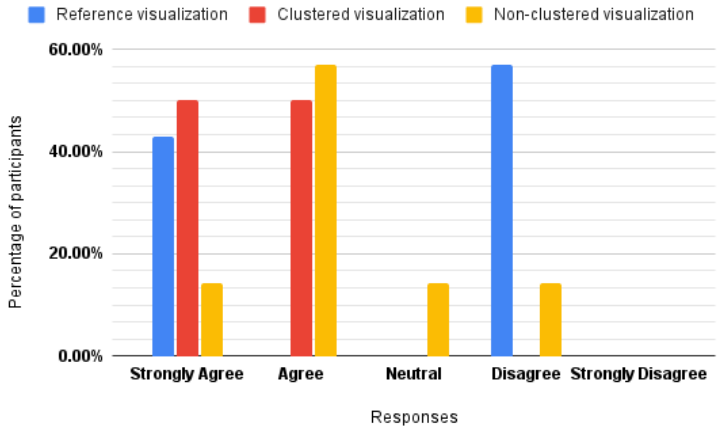

Figure 4.8: Comparison of participants responses the question "the visualization tool had visual clarity"

neutral.

User Experience For the user experience evaluation, all participants stated that clustering the feature maps makes our visualization more user-friendly (Figure 4.7). However, most participants (84%) stated that our visualization is user-friendly, even without clustering the feature maps. For the baseline application, 42% of the participants disagreed that the app is user-friendly. A similar result was also reported for the intuitiveness of the provided visualizations (Figure 4.5).

The answers to the questions indicate that both versions of our visualizations (with and without clustering the feature maps) are user-friendly and adequately intuitive once the users get familiar with using them. Our visualization tool is also easy to learn, and the users' learnability could also improve. However, the baseline visualization tool is less intuitive and not easy to work with compared to the proposed visualizations. This might relate to the knowledge required to work with the this visualization tool.

Background Knowledge Ash shown in Figure 4.6, The majority (71%) of the participants strongly agree that a minimum amount of background knowledge about CNNs is required to be able to gain the most benefits from our visualization tool when the feature maps are not clustered. Interestingly, only 37% of participants think background knowledge is needed to work with our proposed visualization method when the feature maps are clustered. This number increases to 57% for the baseline visualization tool.

Based on the participants' responses to the open-ended question of what pre-knowledge they think is required for understanding the visualizations. They said that *general knowledge of neural network algorithms and mathematical calculations* is required for the baseline tool, and being familiar with *how neural networks work in general*, *feature extraction*, and *classification* are required for working with the proposed clustered and non clustered approaches.

4.5.2 Helpfulness

Clarity of Visualization The first question in this category is about the visual clarity of the information displayed as shown in Figure 4.8. By asking this question, we aim to evaluate the effectiveness of our proposed visualization in terms of clarification of feature maps that are not easy to understand. Our survey shows that all participants agree or strongly agree that clustering the feature maps makes the information provided by the visualization clearer. However, the number goes down to 71% without clustering the feature maps and 42% for the baseline visualization. The low visual clarity is when the feature maps are not clustered because the CNN model (VGG16) (and CNN

models in general) has small feature maps. Therefore, showing them all simultaneously reduces the clarity of information for the end-user.

Understanding Neural Networks The second question in the helpfulness category is about how helpful the participants find our visualization in gaining a deeper understanding of how neural networks work. By this question, we ask the participants how much the visualization helped them become more familiar with the model's decision-making process, architecture, the information flow through the network, the feature maps in each layer, and the effect of internal variables on the model output. All participants agree or strongly agree that clustering the feature maps helps them better understand the CNN model, and this number goes down to 57% for not clustering the feature maps and the baseline visualization tool.

In the following question, the participants explain the visualization's areas that helped them learn. This question includes some responses to choose one or more items from, and they can also provide their answer. Table 4.6 shows the ratio of the participants that voted for each of the pre-defined items for this question in each approach. For the baseline visualization, we can see that most of the participants voted that it helped them visualize the architecture of the network. For both clustered and non-clustered approaches, most of the votes go to understand how the features are computed and what parts of the input network pay attention to.

As shown in Table 4.2, questions 7 and 8 of the questionnaire asks the participants to explain the insight they found from the visualization about the parts of the input that the model pays attention to. For example, question 7 asks if the areas that the model considers match their expectation, and question 8 asks if they could realize if the size of those areas was correct.

For the baseline tool, 85% of the participants could not answer these questions, and 15% replied that the model worked well and the areas it pays attention to match their expectation for one of the images. However, it does not work for the other two images. The reason may lie that this approach does not let the user change the input and play with the model. To realize this

Table 4.6: Percentage of participants that voted for each item in each approach. The items assess how the visualization helped the participants become familiar with different aspects of the model.

Item	**baseline approach**	**clustered approach**	**non-clustered approach**
Allow me to understand how the NN compute the features used for classifications	37%	62%	28%
Allow me to visualize the architecture of the network better	50%	37%	14%
Allow me to understand what is the influence of the networks parameters on the results	0%	37%	14%
Allow me to understand what parts of the input data matter for the network	12%	75%	57%

information, one should pay enough attention to the visualized inner calculations. Given that many of the participants do not have detailed knowledge of the functionality of neural networks, a few could reply to these questions for the baseline approach.

Let's now discuss the insights that the participants reported for the visualization with clustered layers. For all three images, we have only a few (25% for the red car and 12% for the rest) participants who were not sure about their findings and reported no insight. The rest of the participants shared the insights they gained about what input image regions matter the most for the model. For instance, for the blue car, most participants (87%) stated that the network considered most of the input regions they expected, and 50% think the attention regions were sufficient or almost sufficient. On the other hand, 25% of the participants found that the model also considered irrelevant input regions for its classification. Below are some samples of their responses to these questions for the three images they tried.

- *Blue car: I think the presence of the street was very effective in diagnosis because by removing the street, the percentage of car detection decreased.*

- *Blue car: By masking any areas, all the anticipations were including the car.*
- *Removing the car plate made it critical for CNN according to the 3D visualization [of masked image]. The street lines seemed to matter, but I didn't expect this.*
- *Red car: Since it has several dimensions, it brings up different recognitions by each masking scenario. (Even the box behind the car was included in the network recognitions.*
- *Red car: The box behind the car is considered as the image details, which has been recognized by the network correctly.*
- *Cat: It was highlighting the wheel rather than the cat*
- *cat: I think it can improve its functionality for the complex images as well*
- *Cat: The initial recognition was incorrect due to the image complexity; however, it recognized three types of cats in the next steps ... by masking the cat, I expected the network to recognize the bicycle wheel.*

For the proposed visualization without clustering the feature maps, 28% of the participants reported no insights about any of the images. 14% commented that they could get no information from the app. The 72% who responded to the questions did not provide an explanation.

Overall, the results indicate that the participants could gain insights into the model by using customized masks and manipulating the model input and observing the model changes. They could get more insight from the visualization with clustered feature maps compared to the visualization without clustering the feature maps. On the other hand, they got the least insight from the baseline approach. The reason could be that the baseline approach does not let the users make changes to the model input. The following section reviews what visualization approaches the participants preferred and why.

4.5.3 Scene Preference

The first question in this category compares the two main scenes provided in the proposed visualization: (a) 3D visualization of modified images and (b) Network layers visualization. For the *Clustered visualization*, 37% of the participants replied with the model layers scene, 50% replied with the 3D modified images scene, and the rest replied with both. For the *Non-clustered visualization*, 27% of the participants replied with the network layers scene, 42% replied with the 3D modified images scene, 14% replied with both, and the rest did not reply.

This is an open-ended question, and the participants provided their reasons why they think each scene is more useful. Here are the reasons participants provided for their choices:

- *The [clustered] network layers are useful for understanding the changes of input in different layers. For example, the reduction of the input dimension in the end layers was obvious.*
- *The [clustered] network layers were more user-friendly. It provides much more comprehensive and clear experimental results by an accurate masking process.*
- *The [clustered] network layers allows for a better understanding of how the detection (of the main object) might have interfered*
- *In the [non clustered] network layers, we can see the changes of the input in each layer.*
- *The [non clustered] network layers did not add any extra information about the net flow.*
- *The 3D visualization is suitable for comparing model performance and inference for different images*
- *3D [visualization] was also very helpful to distinguish/differentiate between these interfering factors*

- *3D visualization gives me insight into how much each image can belong to what labels and also helps to compare the model performance of different images simultaneously.*

We notice from the results that most participants prefer to have both the model layers and the modified images being provided together. They reported that each has its advantages, and the visualization of the modified image adds valuable information to the network layers visualization scene.

The second question compares the two different visualizations of the network layers: (a) Showing clustered feature maps and (b) Showing all feature maps simultaneously. The majority (75%) of the participants replied that they preferred the clustered feature maps, and 25% selected all feature maps simultaneously. On the other hand, those who liked the feature maps without being clustered stated that they preferred to see them simultaneously. What follows are the main reasons they provided for choosing the clustered feature maps:

- *Because you did not need to go to each layer separately.*
- *I could compare different images in different layers. Also, clustering the images gives me better visual insight into their main differences.*
- *I do not have the image processing experience and the relevant insight. Showing clustered [feature maps in] layers helps me better understand the differences between input categories in each layer. I don't have to see the full feature map.*
- *Allows for better visualization of the whole procedure at the same time*
- *It helped see the flow of the feature maps through the network layer by layer. Looking at all feature maps layer by layer doesn't make any connections in mind.*
- *Full feature maps don't give me an idea of how the network works, and it is hard to go through layers one by one. If the network is bigger, then this visualization could be even worse.*

These results confirm our hypothesis that clustering the feature maps and showing a concise visualization of the network layers is more valuable than showing all feature maps simultaneously. The main reason would be that the visualization with clustered feature maps shows all network layers on the screen and enables comparison. Also, both the network layers scene and the modified images scene provide the most information.

4.5.4 Improvements

Participants were asked to provide what improvements they suggested for the proposed visualization tool. Here, we discuss the features they suggested. For the *Non-clustered visualization*, some participants noted that many images on the screen were confusing for them, and it was hard to get information from the visualization. The other feedback is about showing the connection of the layers simply so that the structure can be better understood. This feedback matches what we expected. The *Clustered visualization* conceals a large number of feature maps in each layer and is less confusing to work with. It also visualizes all the layers beside each other simultaneously, which makes it easier to understand the network structure than the *Non-clustered visualization*. *Clustered visualization* hides the layer connections since the size of the links is enormous. For the *Clustered visualization*, we have some suggestions for improving the UI and speed of the visualization. Our visualization interface has room for improvement. Each time a user feeds a modified image to the model, recalculating the layer's information and updating the visualized model takes about 20 seconds. It can improve to work close to real-time.

Some suggestions also relate to the information we show. For example, some participants suggested that we display the connections of different layers and the calculations of each layer in a simple way. We did not include the layer connections in our proposed visualization approach to avoid the mathematical details. However, visualizing the network connections simply can further improve our approach. Also, some of the label words, such as 'red panda', were unfamiliar to the participants, and they had to search for their meanings while working with the visualization. Therefore, participants suggested that

we provide more information on the meaning of the labels.

They also suggested letting the user merge some labels into one. The VGG16 model has 1000 classes, and some of these classes are very close. For example, the user's network application might not be sensitive to differentiating a car, a convertible, and a sports car. By playing with the network, the user might realize that merging some of the output labels into one could increase the model accuracy.

The other improvement suggestions in the proposed approach are to (1) provide multiple CNN models so that the user can compare them, (2) show the CNN training information such as the loss rate and feature maps development during the training, (3) ability to modify the CNN structure and retrain it, and (4) the ability to compare a specific layer for multiple modified images. All these suggestions can improve our visualization approach.

We also had some feedback from the participants who work in the medical field on how such visualizations could improve for use in medical applications. First, the visualization response time when applying changes should improve, as suggested above. The second suggestion is to let the user play with the visualization to ensure its ability for human body recognition. For example, they are differentiating between children and adults. This can be done by letting the user choose their own set of images and try the visualization. Also, in medical imaging, distinguishing between the artifacts and the tissue is vital to avoid mistakes [11]. To help analyze the network's performance for avoiding artifacts, the visualization would better allow for more detailed modifications. This way, the user can remove some artifacts/tissue and analyze the model response. One way to allow this is to add an image segmentation technique to the visualization approach to support applying sensitive and small masks with mice.

Chapter 5

Conclusion

CNN has shown to be a powerful method to solve multiple image processing tasks. However, despite being so powerful, they are black-boxes, making them hard to interpret and analyze their behaviour. Interpreting and understanding how CNNs process the input image and generate their output help us teach these models to students more effectively and improve them faster. Visualization is a powerful tool to understand such black-box models. Visualization of CNN models is challenging due to their size and complexity. CNN models have a massive set of neurons and apply many nonlinear transformations on the input. Visualizing such an enormous set of variables makes it challenging for the user to gain a deep understanding of the network. Also, the complexity of the calculations makes it difficult to extract relevant and informative information to visualize.

This thesis proposes a visualization tool for CNN models based on partitioning the feature maps and showing only a single representative of each cluster. In addition to visualizing the feature maps in each convolutional layer, we apply binary masks on the input image to generate modified images and visualize the model output for these altered images. This helps the user further investigate the impact of each region of the original image on the decisions of the CNN model. We evaluate our proposed new tool by running an experimental study. We design our experiments to involve our proposed visualization approach with and without clustering the feature maps, and a recent public domain visualization tool as our baseline. Finally, we develop a

questionnaire that evaluates the effectiveness of our proposed approach. We asked eight participants of our study to experiment with the provided visualizations to understand how the model processes the input image and fill in the questionnaire to provide their feedback.

Our experimental study shows that clustering the feature maps is more effective than showing all the feature maps simultaneously. In addition, we found that beginner users gain a deeper level of understanding about the model by investigating its sensitivity.

For future work, we suggest improving our tool by supporting more models and letting the user compare multiple models with the same input image. This can help the user compare how different models process the input image. Furthermore, we recommend adding visualization of the network's training process to show how the feature maps change during the training process. This can help with gaining information on how each feature maps evolve to their final form. Interactive visualization of training process can also let the user pass a set of images through the network and learn how the network inner values change while training.

References

[1] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek, "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation," *PloS one*, vol. 10, no. 7, 2015.

[2] M. Bock and A. Schreiber, "Visualization of neural networks in virtual reality using unreal engine," in *Proceedings of the 24th ACM Symposium on Virtual Reality Software and Technology*, 2018, pp. 1–2.

[3] K. Browne, B. Swift, and H. Gardner, "Critical challenges for the visual representation of deep neural networks," in *Human and Machine Learning*, Springer, 2018, pp. 119–136.

[4] D. Chang, L. Dooley, and J. E. Tuovinen, "Gestalt theory in visual screen design—a new look at an old subject," 2002.

[5] S. Chung, S. Suh, C. Park, K. Kang, J. Choo, and B. C. Kwon, "Revacnn: Real-time visual analytics for convolutional neural network," in *KDD 16 Workshop on Interactive Data Exploration and Analytics*, 2016.

[6] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[7] N. Das, H. Park, Z. J. Wang, *et al.*, "Massif: Interactive interpretation of adversarial attacks on deep learning," in *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems*, 2020, pp. 1–7.

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, Ieee, 2009, pp. 248–255.

[9] F. Doshi-Velez and B. Kim, "Towards a rigorous science of interpretable machine learning," *arXiv preprint arXiv:1702.08608*, 2017.

[10] D. Erhan, Y. Bengio, A. Courville, and P. Vincent, "Visualizing higher-layer features of a deep network," *University of Montreal*, vol. 1341, no. 3, p. 1, 2009.

[11] M. K. Feldman, S. Katyal, and M. S. Blackwood, "Us artifacts," *Radiographics*, vol. 29, no. 4, pp. 1179–1189, 2009.

[12] R. C. Fong and A. Vedaldi, "Interpretable explanations of black boxes by meaningful perturbation," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 3429–3437.

[13] E. Fouh, M. Akbar, and C. A. Shaffer, "The role of visualization in computer science education," *Computers in the Schools*, vol. 29, no. 1-2, pp. 95–117, 2012.

[14] S. Guberman, "On gestalt theory principles," *Gestalt Theory*, vol. 37, no. 1, pp. 25–44, 2015.

[15] S. Hansen, N. H. Narayanan, and M. Hegarty, "Designing educationally effective algorithm visualizations," *Journal of Visual Languages & Computing*, vol. 13, no. 3, pp. 291–317, 2002.

[16] A. W. Harley, "An interactive node-link visualization of convolutional neural networks," in *International Symposium on Visual Computing*, Springer, 2015, pp. 867–877.

[17] F. Hohman, H. Park, C. Robinson, and D. H. P. Chau, "Summit: Scaling deep learning interpretability by visualizing activation and attribution summarizations," *IEEE transactions on visualization and computer graphics*, vol. 26, no. 1, pp. 1096–1106, 2019.

[18] I. T. Jolliffe, "Principal component analysis," *Technometrics*, vol. 45, no. 3, p. 276, 2003.

[19] M. Kahng, P. Y. Andrews, A. Kalro, and D. H. P. Chau, "Activis: Visual exploration of industry-scale deep neural network models," *IEEE transactions on visualization and computer graphics*, vol. 24, no. 1, pp. 88–97, 2017.

[20] A. Karpathy, "Convnetjs: Deep learning in your browser (2014)," *URL http://cs. stanford. edu/people/karpathy/convnetjs*, vol. 5, 2014.

[21] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[22] Y. LeCun, B. Boser, J. S. Denker, *et al.*, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[23] M. Liu, S. Liu, H. Su, K. Cao, and J. Zhu, "Analyzing the noise robustness of deep neural networks," in *2018 IEEE Conference on Visual Analytics Science and Technology (VAST)*, IEEE, 2018, pp. 60–71.

[24] M. Liu, J. Shi, K. Cao, J. Zhu, and S. Liu, "Analyzing the training processes of deep generative models," *IEEE transactions on visualization and computer graphics*, vol. 24, no. 1, pp. 77–87, 2017.

[25] M. Liu, J. Shi, Z. Li, C. Li, J. Zhu, and S. Liu, "Towards better analysis of deep convolutional neural networks," *IEEE transactions on visualization and computer graphics*, vol. 23, no. 1, pp. 91–100, 2016.

[26] S. Lloyd, "Least squares quantization in pcm," *IEEE transactions on information theory*, vol. 28, no. 2, pp. 129–137, 1982.

[27] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[28] A. Marcus, L. Feng, and J. I. Maletic, "3d representations for software visualization," in *Proceedings of the 2003 ACM symposium on Software visualization*, 2003, 27–ff.

[29] Martín Abadi, Ashish Agarwal, Paul Barham, *et al.*, *TensorFlow: Large-scale machine learning on heterogeneous systems*, Software available from tensorflow.org, 2015. [Online]. Available: `https://www.tensorflow.org/`.

[30] N. Meissler, A. Wohlan, N. Hochgeschwender, and A. Schreiber, "Using visualization of convolutional neural networks in virtual reality for machine learning newcomers," 2019.

[31] S. Mohseni, N. Zarei, and E. D. Ragan, "A multidisciplinary survey and framework for design and evaluation of explainable ai systems," *arXiv*, arXiv–1811, 2018.

[32] C. Molnar, *Interpretable Machine Learning, A Guide for Making Black Box Models Explainable*. 2019, `https://christophm.github.io/interpretable-ml-book/`.

[33] W. J. Murdoch, C. Singh, K. Kumbier, R. Abbasi-Asl, and B. Yu, "Interpretable machine learning: Definitions, methods, and applications," *arXiv preprint arXiv:1901.04592*, 2019.

[34] E. Olshannikova, A. Ometov, and Y. Koucheryavy, "Towards big data visualization for augmented reality," in *2014 IEEE 16th Conference on Business Informatics*, IEEE, vol. 2, 2014, pp. 33–37.

[35] A. Paszke, S. Gross, F. Massa, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

[36] N. Pezzotti, T. Höllt, J. Van Gemert, B. P. Lelieveldt, E. Eisemann, and A. Vilanova, "Deepeyes: Progressive visual analytics for designing deep neural networks," *IEEE transactions on visualization and computer graphics*, vol. 24, no. 1, pp. 98–108, 2017.

[37] M. T. Ribeiro, S. Singh, and C. Guestrin, "" why should i trust you?" explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1135–1144.

[38] C. Rudin, "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead," *Nature Machine Intelligence*, vol. 1, no. 5, pp. 206–215, 2019.

[39] J.-W. Seo and S. D. Kim, "Novel pca-based color-to-gray image conversion," in *2013 IEEE international conference on image processing*, IEEE, 2013, pp. 2279–2283.

[40] C. E. Shannon, "A mathematical theory of communication," *The Bell system technical journal*, vol. 27, no. 3, pp. 379–423, 1948.

[41] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[42] D. Smilkov and S. Carter, "Tensorflow-neural network playground," *Playground. tensorflow. org*, 2017.

[43] D. Smilkov, S. Carter, D. Sculley, F. B. Viégas, and M. Wattenberg, "Direct-manipulation visualization of deep networks," *arXiv preprint arXiv:1708.03788*, 2017.

[44] M. Song, D. Tao, C. Chen, X. Li, and C. W. Chen, "Color to gray: Visual cue preservation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1537–1552, 2010.

[45] M. Stitson, J. Weston, A. Gammerman, V. Vovk, and V. Vapnik, "Theory of support vector machines," *University of London*, vol. 117, no. 827, pp. 188–191, 1996.

[46] H. Strobelt, S. Gehrmann, H. Pfister, and A. M. Rush, "Lstmvis: A tool for visual analysis of hidden state dynamics in recurrent neural networks," *IEEE transactions on visualization and computer graphics*, vol. 24, no. 1, pp. 667–676, 2017.

[47] *Tensorspace playground*, `https://tensorspace.org/`, Accessed: 2020-12-25.

[48] K. C. VanHorn, M. Zinn, and M. C. Cobanoglu, "Deep learning development environment in virtual reality," *arXiv preprint arXiv:1906.05925*, 2019.

[49] J. Wang, L. Gou, H. Yang, and H.-W. Shen, "Ganviz: A visual analytics approach to understand the adversarial game," *IEEE transactions on visualization and computer graphics*, vol. 24, no. 6, pp. 1905–1917, 2018.

[50] Z. J. Wang, R. Turko, O. Shaikh, *et al.*, "Cnn explainer: Learning convolutional neural networks with interactive visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 2, pp. 1396–1406, 2020.

[51] N. Xie, G. Ras, M. van Gerven, and D. Doran, "Explainable deep learning: A field guide for the uninitiated," *arXiv preprint arXiv:2004.14545*, 2020.

[52] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*, Springer, 2014, pp. 818–833.

[53] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus, "Deconvolutional networks," in *2010 IEEE Computer Society Conference on computer vision and pattern recognition*, IEEE, 2010, pp. 2528–2535.

[54] M. D. Zeiler, G. W. Taylor, R. Fergus, *et al.*, "Adaptive deconvolutional networks for mid and high level feature learning.," in *ICCV*, vol. 1, 2011, p. 6.

[55] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2921–2929.

[56] L. M. Zintgraf, T. S. Cohen, T. Adel, and M. Welling, "Visualizing deep neural network decisions: Prediction difference analysis," *arXiv preprint arXiv:1702.04595*, 2017.