Computer Vision Lab Dresden

&

Computer Graphics and Visualization

Internship Report

# Visualization of Convolutional Neural Networks

*Author:*
Dhruv Jain
Abhinav Sharma
Aparna Balagopalan

*Supervisors:*
Prof. Stefan Gumhold
Prof. Christin Seifer
Prof. Carsten Rother
Dr. Dmitrij Schlesinger
Dr. Omid Hosseini

July 25, 2016

# 1 Introduction

In the recent past, CNN models have been shown to outperform state-of-the-art models in the field of object classification, semantic segmentation, genomics etc. However, our understanding of how these models work, especially what computations they perform at intermediate layers has not been complete. The main motivation is to **Visualize and Interpret the neural nets for a deeper understanding**. A better understanding of these models could lead to a more handcrafted design of CNN's, higher efficiency or better performance as achieved by [1] We can divide the visualization tasks completed into two broad categories -

- **Activation of filters**
- **Gradient Based**

Both 2D and 3D visualization was performed in each case.

# 2 Dataset and CNN Model

The ImageNet dataset is used. ImageNet populates 21,841 synsets of WordNet with an average of 650 manually verified and full resolution images. As a result, ImageNet contains 14,197,122 annotated images organized by the semantic hierarchy of WordNet. ImageNet is larger in scale and diversity than the other image classification datasets which is the reason for choosing this dataset for our research. We used the training images in ImageNet since the training process was our major area of analysis.

The CNN model used is Caffenet, a modified version of the popular AlexNet with the order of the normalization and pooling layers reversed.

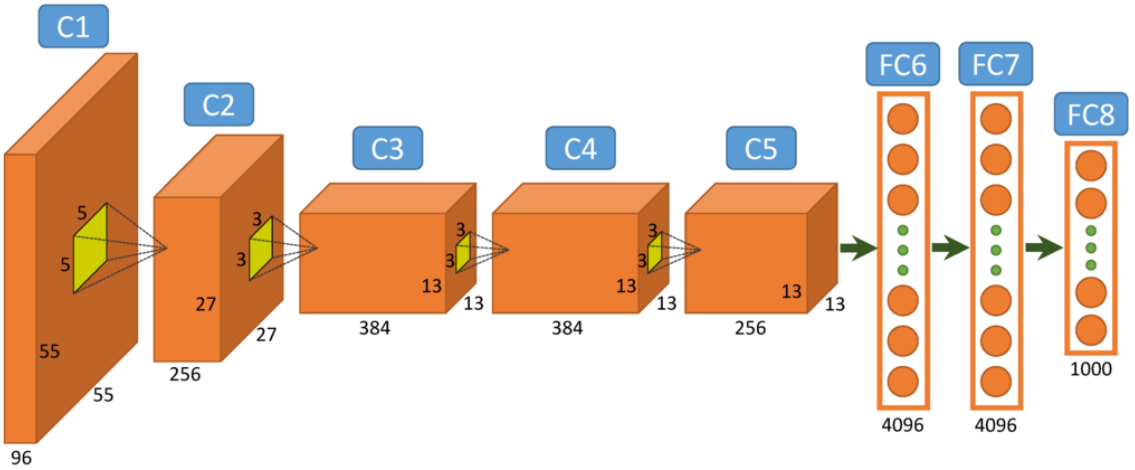



Figure 1: Caffenet Model

# 3 Visualization

## 3.1 Activation Based Visualization

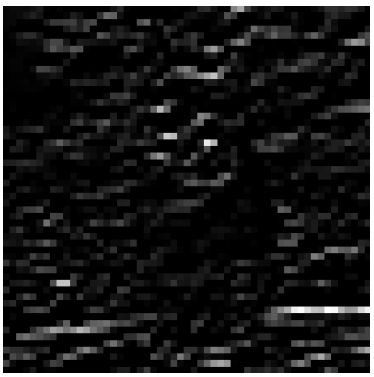
### 3.1.1 2D Unordered Visualization

A sample image from a class of Imagenet database is taken as input and fed to the above network. The filter activations of each layer is visualized as images by appropriate reshaping of the output activations. A closer look is taken at the output images of the convolution layer 1 filters as they are easily interpretable as compared to other layers. Here we gained insights as to what functions different filters of the layer are performing. A few filter output images are shown below with the functions performed by them written below the images:

Input Image (Class: Tabby Cat)

Vertical edge detection

Foreground-background separation



Horizontal Edge Detection

### 3.1.2 3D Unordered Visualization

The filter output images of conv1 layer is visualized using the CGV visualization tool to obtain a 3D visualization of the output in order to have a complete view of the layer as a whole. The output images of each filter are stacked over one another to generate this volume. A screenshot of the volume is shown below:

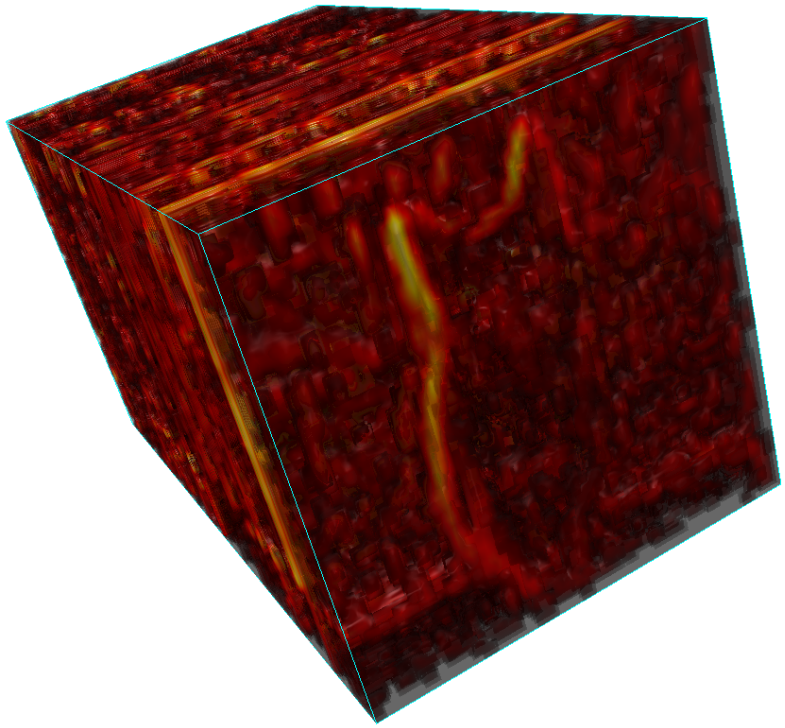


Figure 2: 3-D Visualization of Conv1 Layer Output

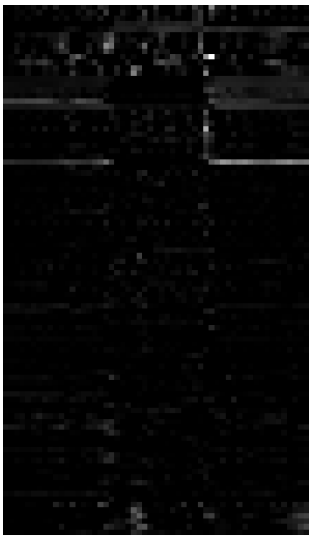
### 3.1.3 Finding patterns between filter output images

In an image the neighbouring rows have a degree of correlation or pattern amongst themslves.In order to arrange the layer outputs such that the neighbouring filter outputs have similarity between themselves different methods were tried.

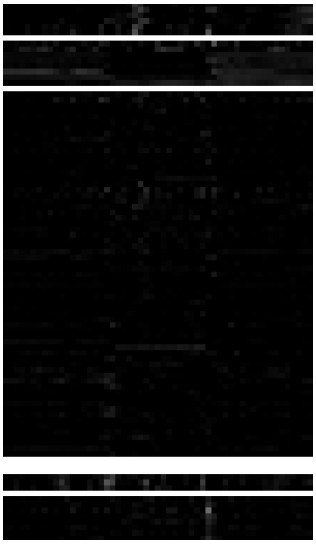
- **Slice-Based Reordering:** The following measures were implemented in order to find patterns in the space of a particular layer output on different image slices in order to see if the order obtained was similar for different sets of slices.

– **K-means clustering:** A vertical slice from the volume of filter output images of a layer is taken and K-means clustering on rows of the image slice performed. In this method, clusters of similar rows were obtained. Example: Results for conv1 layer for the 15th slice are shown below:

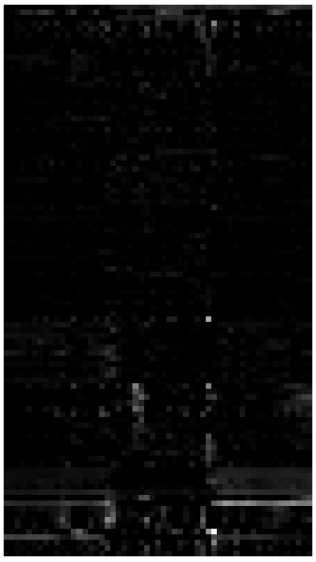
Input (Slice 15)



10 Means



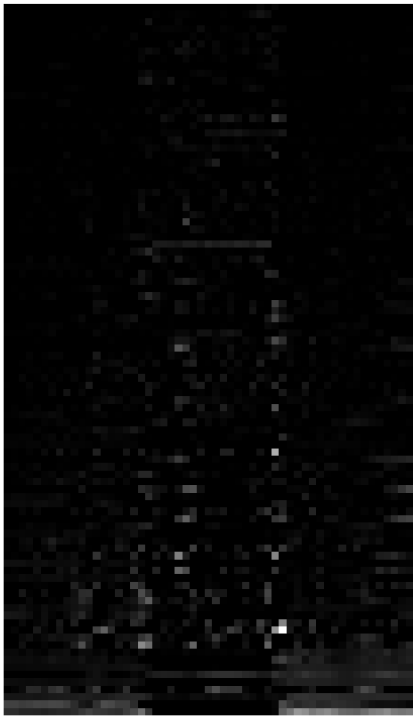
7 means (Clusters separated by White bars)



20X5 Means

- **PCA reordering:** A vertical slice from the volume of filter output images of a layer is taken and **Principal Component Analysis(PCA)** on rows of the image slice. The rows are then reordered on the basis of the values of projection on the principal component in increasing order. eg. Results for conv1 layer for the $15^{\text{th}}$ slice are shown below:

Input (Slice 15)



PCA reordered image

Since the order from different slices were different, though consecutive slices produced almost similar ordering, the method of PCA was extended to each filter output.

- **Filter output - based reordering:** This method was implemented to find patterns between whole filter output images instead of between rows in a slice from the volume. Here the output images are stacked column wise to form a vector and Principal Component Analysis (PCA) is done on them. The vectors are then reshaped back to images and are reordered accordingly in the volume. This volume of reordered images are visualized using the cgv-viewer. A screenshot of the reordered volume visualization of conv1 layer is shown below:

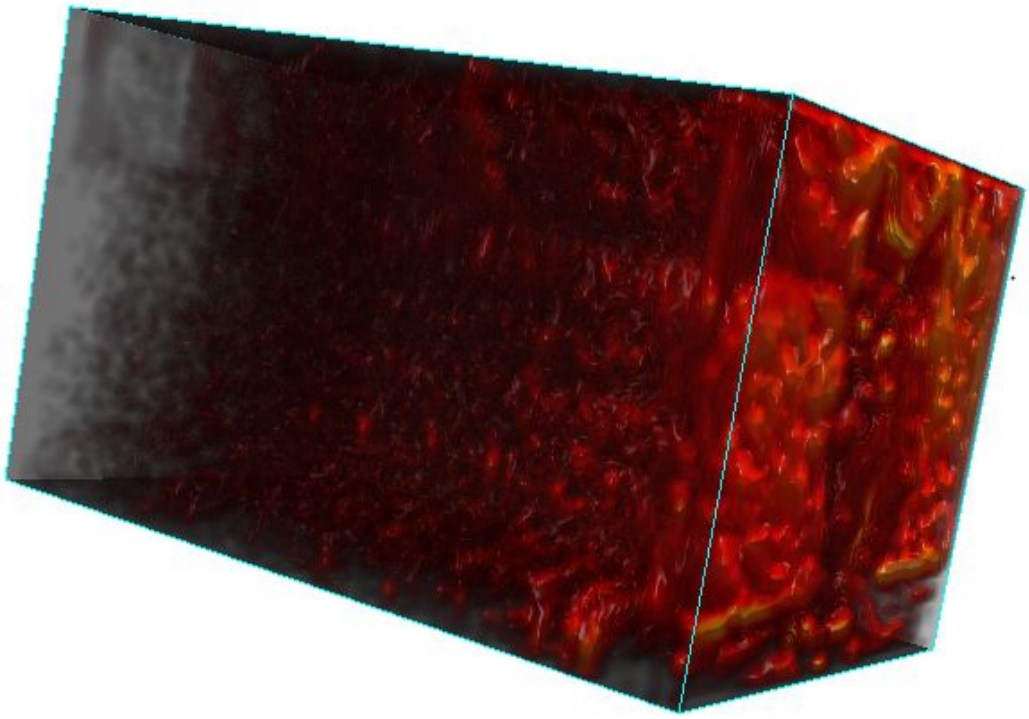


Figure 3: 3-D Visualization of PCA Reordered Conv1 layer Output

## 3.2 Gradient Based Visualization

In the paper [2], LRP algorithm is used to compute scores for image pixels and image regions denoting the impact of the particular image region on the prediction of the classifier for one particular test image. This is done using the activation values of neurons. Since the gradient of the output of the network or the final layer neuron, responsible for classifying an image to a particular class, with respect to the filter weights is a direct reflection of how sensitive they are to change in the weights and is easily interpretable, this was focused in our approach in order to get a similar "importance" measure or score for each filter in the CNN. This was computed using backpropagation and slight modifications to the loss layers of the network chosen. In effect, the loss was made a function of only the neuron in analysis and the gradients obtained.This was also validated by computing an approximation of the gradients by changing the filter weights and calculating the corresponding change in the activation of the neurons.

### 3.2.1 Gradient Calculation

Gradient of a neuron in the final fully-connected layer with respect to filters in a particular layer was computed using backpropagation and slight modifications to the loss layers of the network chosen. In effect, the loss was made a function of only the neuron in analysis by modifying the caffe prototxt file and setting the loss weight corresponding to the neuron under analysis to unity and the rest to zero. The corresponding gradients were obtained and further analyzed by visualizing them.

**Validation:**

This was also validated by computing an approximation of the gradients by changing the filter weights and calculating the corresponding change in the activation of the neurons.

$$Gradient = \frac{\Delta NeuronActivation}{\Delta FilterWeights} \tag{1}$$

### 3.2.2 Visualization of Gradients

The gradient values obtained were tensors hence could be visualized as a 3D volume. For analysis of the gradient values, using the CGV 3D viewer, the gradients were rendered volumetrically. In order to visualize the variation of gradients for input images belonging to different classes, three types of classes were chosen:

* A reference class
* Classification under inspection
* A similar Class
* A dissimilar Class

.

From each class a correctly classified and misclassified image were chosen and the gradient variations analyzed.

Reference Class

Similar Class

Dissimilar Class

Next, the gradients for sample images from classes mentioned above were visualized as a time-series in the cgv-viewer. The time-series function displays the 3D volume of gradients of those classes one after the other at a particular input time step. This helps us to the the similarities and differences in the gradients of images from different classes. For dissimilar class the values of gradients were less as compared to correctly classified class image as shown in few screenshots below:

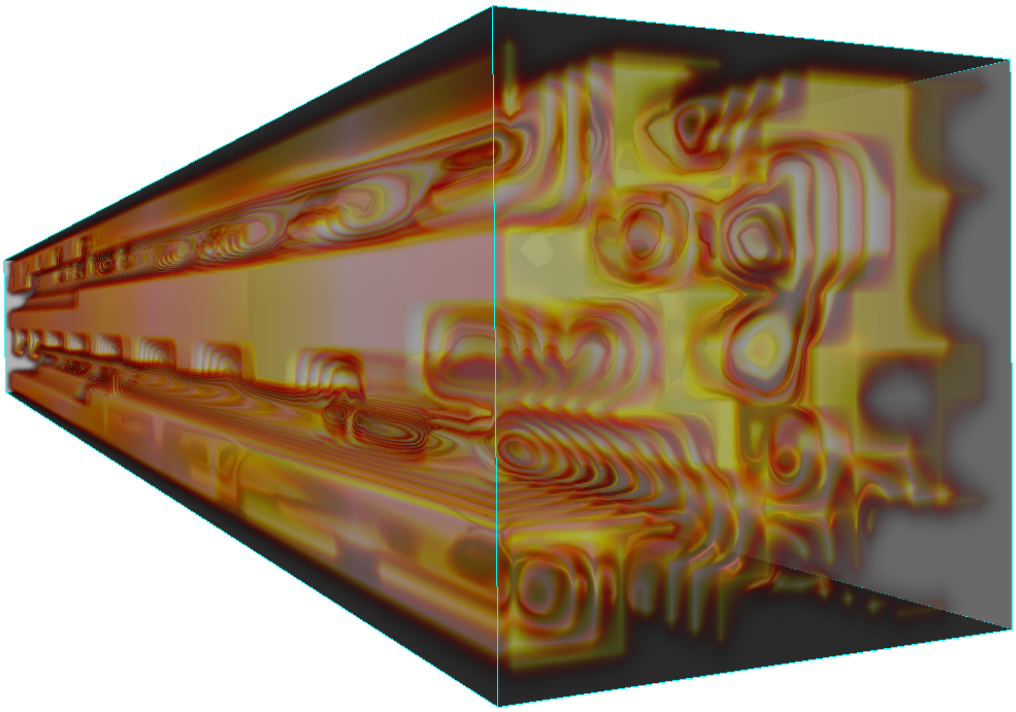


Figure 4: Chihuahua correctly classified

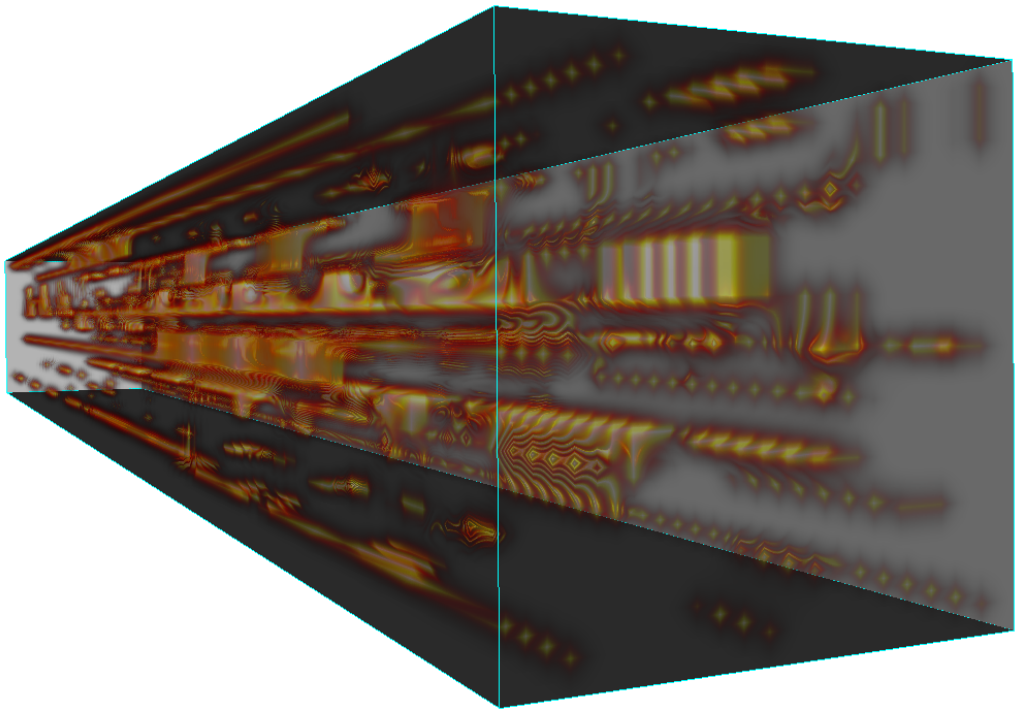


Figure 5: Chihuahua miss classified

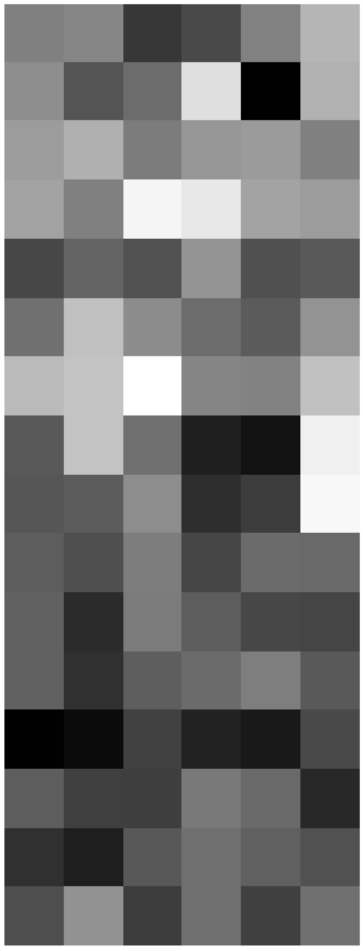
### 3.2.3 Importance measure

A score was obtained for each filter in the network by taking the norm of the gradient corresponding to the filter. A higher score implied a larger gradient and hence, higher relevance.

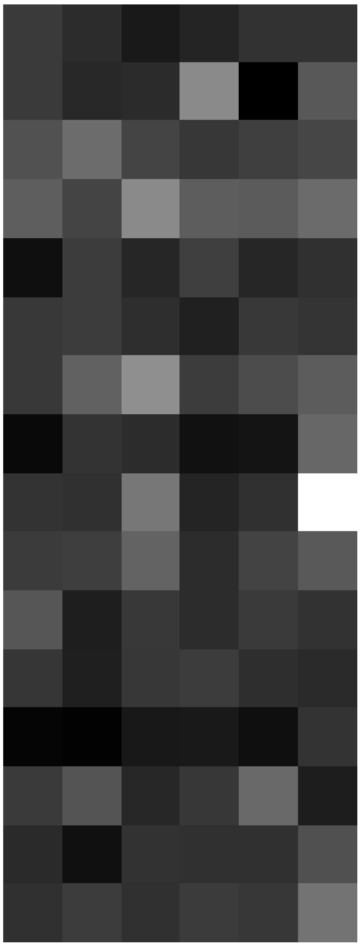
**Visualization**

In order to see the variation of importance values , visualization as a image was done.

For example, in the first convolutional layer in CaffeNet, there are 96 filters.The importance value for each filter was visualized in a 16x6 grid as shown below:



Importance w.r.t class 281 given input belonging to same class



Importance w.r.t class 281 given input belonging to different class

Since the gradients are input-dependent, they were averaged over a class to obtain a class-wise relevance metric in the classification to the class of reference neuron. The goal of computing these scores was to find **Unused and highly used parts of the network.**

Attempts at this involved defining a threshold below which the neurons, even if discarded, would not affect the performance of the network.This was performed using two methods:

- **Direct Analysis:** Removing nodes by directly using histogram of gradients

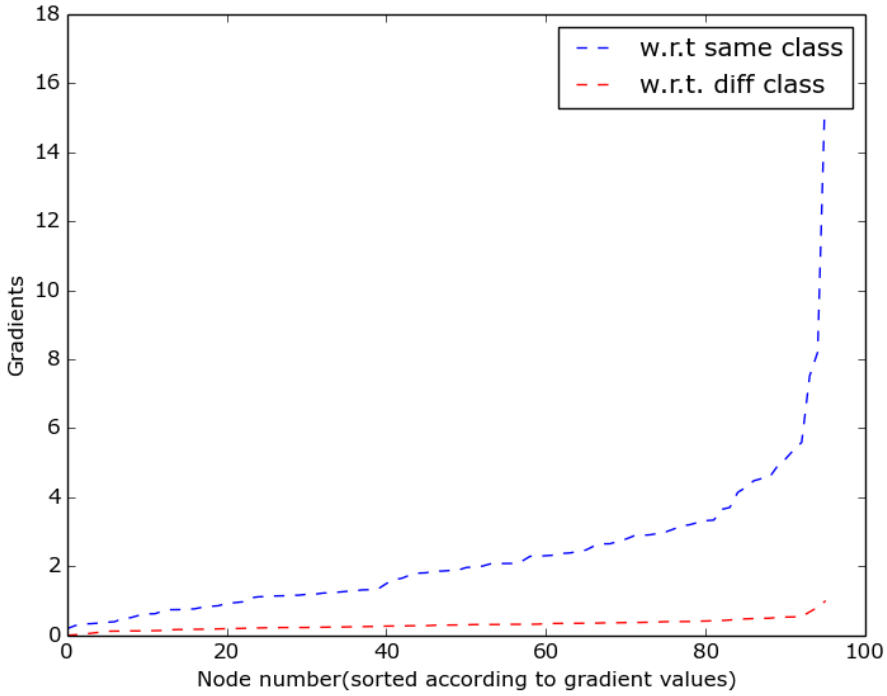



Figure 6: 3-D Visualization of PCA Reordered Conv1 layer Output

| Class | Initial error rate(%) | Error rate (least important node removed)(%) | Error rate (most important node removed)(%) |
|---|---|---|---|
| Persian cat | 10.5 | 10.5 | 14 |
| Pembroke | 9.39 | 8.83 | 15.46 |

$$Error\,rate = \frac{Number\,of\,misclassified\,images\,belonging\,to\,given\,class}{Total\,Number\,of\,images\,in\,given\,class} \quad (2)$$

- **Varying Threshold Analysis:** Varying the threshold and computing the error rate by making the weights of the neurons below that threshold to zero

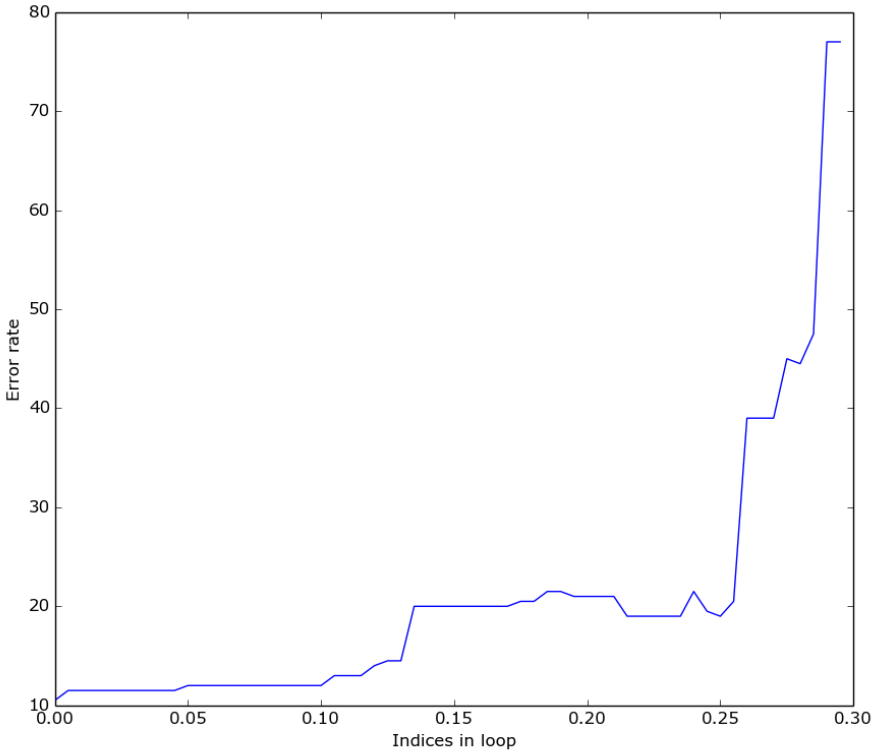



Figure 7: Analysis by varying threshold on Persian cat class(neuron num:283)

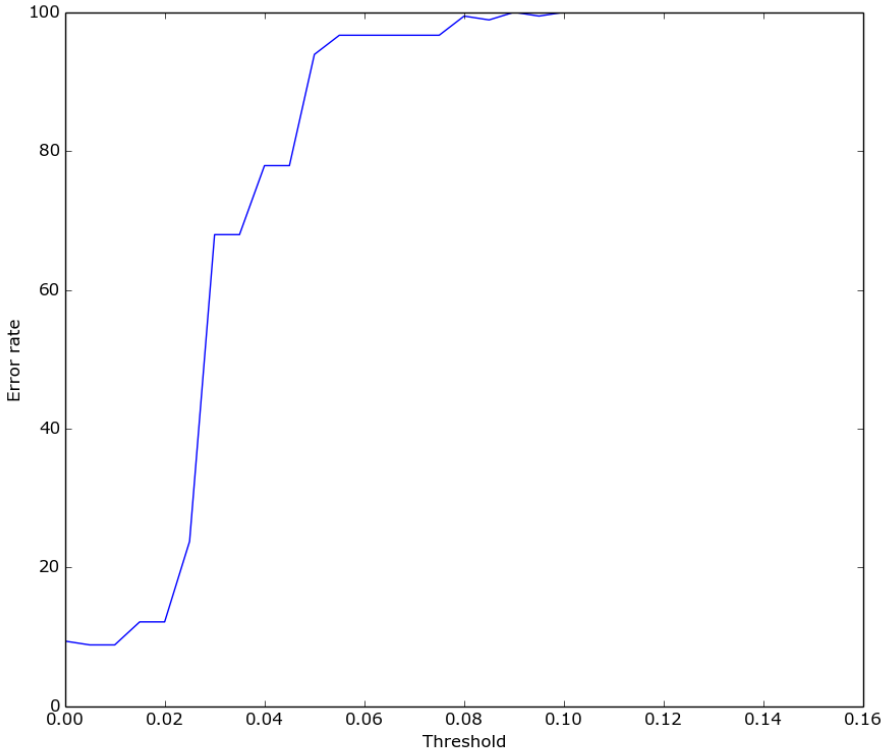



Figure 8: Importance w.r.t class 281 given input belonging to different class

# 4 Conclusion

Apart from giving insight into the operation of CNN's, visualization of the activation can help in building better architectures as well such as when [1] made changes to [3] CNN architecture such that new architecture retained much more information in the 1$^{st}$ and 2$^{nd}$ layer features and as a result, achieved better performance. 3D Visualization, in particular, is helpful in visualizing the higher level gradients and features which cannot be analyzed perfectly as images. The metric of importance developed during our research could help in pruning a network in addition to help identify the highly used parts of the network.

# 5 Structuring

The initial phase of our project involved an extensive study of the existing literature and code on visualizing and understanding CNN's and deep architecture in which everyone contributed equally. The division of work was more or less like the follow:

- **Dhruv Jain:**
  - Initial 2-D visualization of 5 Convolutional Layers.
  - Created binaries of output of 5 Convolutional layers to be fed into CGV viewer tool.
  - Explored the various output visualizations using the cgv-viewer for different inputs and parameter settings.
  - Developed code for PCA of rows from slices of filter output volume.
  - Developed code for PCA of filter output images and visualized them in cgv-viewer.
  - Worked on Finding the Gradients and plotting them to find importance measure.
  - Developed code to find gradients of similar, dissimilar classes and running them on cluster.
- **Abhinav Sharma:**
  - Initial 2-D visualization of 5 Convolutional Layers.
  - Explored the various output visualizations using the cgv-viewer for different inputs and parameter settings.
  - Developed code for various k-means clustering of rows from slices of the volume of filter outputs.
  - Developed code for PCA of rows from slices of filter output volume.
  - Developed code for PCA of filter output images and visualized them in cgv-viewer.
  - Helped in plotting and analyzing importance measure using gradients.
  - Visualized gradients in 3D using cgv-viewer first individually and then as a time series.

- **Aparna Balagopalan:**
  - ∗ Worked on activation maximization of a neuron, study using DeepViz visualization toolbox.
  - ∗ Explored various 3D visualizations of filter activations using CGV Viewer for patterns in the visualization using different parameter settings.
  - ∗ Worked on finding ordering between filters after k-means by groupings using similarity measures between cluster centroids
  - ∗ Developed an approximation of the gradients for validation
  - ∗ Worked out various net operations on the basis of the importance measure, also involving class-wise accuracy calculations using Imagenet dataset.
  - ∗ Developed code to compute gradients of a neuron in the final layer of a network with respect to weights in the intermediate layers of the network
  - ∗ Developed code to get an importance measure for every filter in the network

The code for the project is available at: `https://github.com/dhruvjain/CNN-VIS`

# References

[1] Zeiler, Matthew D., and Rob Fergus. "Visualizing and understanding convolutional networks." European Conference on Computer Vision. Springer International Publishing, 2014.

[2] Binder, Alexander, et al. "Layer-Wise Relevance Propagation for Deep Neural Network Architectures." Information Science and Applications (ICISA) in 2016

[3] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012.