Interactive Demo: A Visual Analysis System for Analyzing Deep Convolutional Neural Networks

Mengchen Liu LIUMC13@MAILS.TSINGHUA.EDU.CN
Jiaxin Shi ISHIJIAXIN@126.COM
Zhen Li ZHEN-LI11@MAILS.TSINGHUA.EDU.CN
Chongxuan Li CHONGXUANLI1991@GMAIL.COM
Jun Zhu DCSZJ@MAIL.TSINGHUA.EDU.CN
Shixia Liu SHIXIA@TSINGHUA.EDU.CN
Tsinghua University, Beijing, 100084 China

Abstract

In this paper, we introduce a visual analysis system called CNNVis, which aims to facilitate experts in better understanding and refining deep convolutional neural networks (CNNs). Given a trained CNN, CNNVis first extracts the derived features of neurons and connections between neurons. In addition, CNNVis aggregates the network to effectively visualize a large CNN. To help experts analyze deep models, CNNVis employs a hybrid visualization to visualize multiple facets of neurons and the connections between them. To demonstrate the effectiveness and usefulness of our system, we have conducted two case studies on two benchmark datasets.

1. Introduction

Deep convolutional neural networks (CNNs) have brought major breakthroughs in processing images (Krizhevsky et al., 2012; Szegedy et al., 2015), video (Karpathy et al., 2014), and speech (Sainath et al., 2013). In spite of the encouraging success of deep CNNs, the understanding of their inner working mechanism of them is limited. As a result, constructing high quality deep models is often reduced to substantial trial-and-error (Bengio, 2009; Bengio et al., 2013; Yosinski et al., 2015).

Recently, researchers have worked towards a better understanding of deep CNNs. The effort has mainly focused on calculating and visualizing the learned features of neurons for the task of image classification (Mahendran & Vedaldi, 2015; Simonyan et al., 2013; Yosinski et al., 2015; Zeiler & Fergus, 2014). In these methods, the learned features are represented by image patches or synthesized images. These methods can reveal what a deep CNN learns at different stages of computation. However, they fail to disclose the inner working mechanisms of CNNs, including the role of each neuron for different classes of images and the interactions between neurons.

To solve this problem, we are introducing a visual analysis system, CNNVis (Mengchen et al., 2016), to reveal the multiple facets of neurons and the connections between neurons. A demo video is available at http://shixialiu.com/publications/cnnvis/CNNVis-icml.mp4. With CNNVis, experts can explore a deep CNN from different perspectives, such as the learned features or the activations of neurons. In addition, it enables experts to examine the interactions between different neurons. In this paper, we illustrate the major components of CNNVis. We also present two case studies that demonstrate the promise of our system to understand and refine deep CNNs.

2. System Overview

Here, we provide an overview of CNNVis, starting with its user interface, followed by its system architecture.

2.1. User Interface

Fig. 1 depicts the user interface of CNNVis. It contains two different interactive areas: the CNNVis visualization and the control panel. The visualization view consists of two parts:

- A visualization to reveal the multiple facets of neurons and the connections between neurons (Fig. 1 (a));
- A line chart that shows an overview of the debugging information such as the average gradient of each layer (Fig. 1 (b)).

Proceedings of the 33^{rd} International Conference on Machine Learning, New York, NY, USA, 2016. JMLR: W&CP volume 48. Copyright 2016 by the author(s).

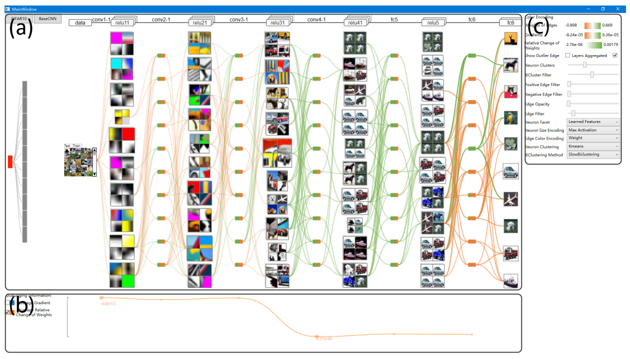

Figure 1. CNNVis user interface: (a) hybrid visualization; (b) line chart to show debugging information; (c) control panel.

The control panel (Fig. 1 (c)) contains a set of controls that enables experts to interactively customize the visualization such as changing the color coding of edges and filtering out unimportant edges.

2.2. System Architecture

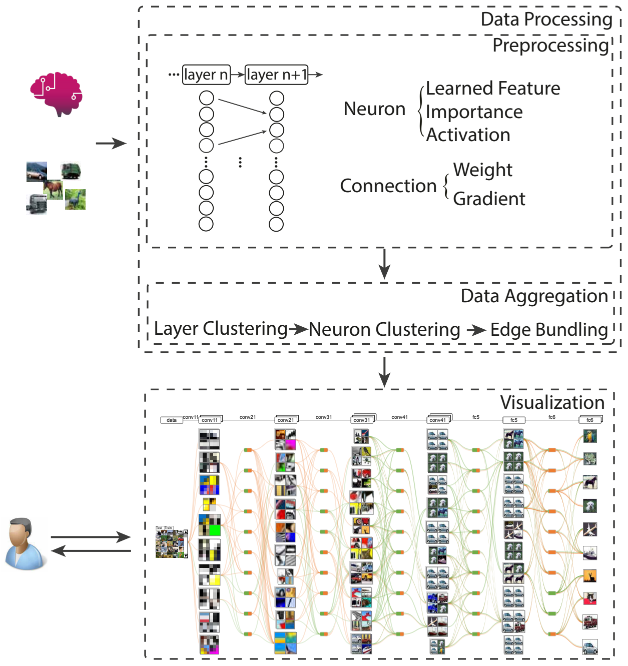

Figure 2. CNNVis system pipeline.

As shown in Fig. 2, CNNVis contains the following components:

- A preprocessing module that converts a CNN into a directed acyclic graph (DAG) and computes the derived features of neurons and connections between neurons;
- A data aggregation module that aggregates the layers, neurons, and the connections between neurons;
- A visualization module that shows multiple facets of neurons and the connections between neurons.

To use CNNVis, experts first need to load a trained CNN and the corresponding training set. As a CNN has no feedback loop, it can be formulated as a DAG. Based on this formulation, CNNVis then computes the derived features of neurons and the connections between neurons. The DAG and the derived features are then fed into the data aggregation module. Based on the derived features, the data aggregation module aggregates the DAG to effectively visualize a large CNN. It aggregates the layers, neurons, and connection between neurons. Experts can interactively modify the aggregation results according to their domain knowledge. The visualization module takes the derived features and aggregation results as input and visually illustrates them in a hybrid visualization. Experts can interact with the generated visualization for further analysis.

3. Data Processing

In this section, we introduce the data processing of CNNVis, which includes preprocessing and data aggregation.

3.1. Preprocessing

The processing module first converts the input CNN into a DAG, where each neuron is represented by a node and the connection between neurons are formulated as an edge. Then this module computes the derived features of neurons and the connections between them.

Neurons. To compute the derived features of neurons, we first run each image in the training set through the network. In this way, we compute the activations of neurons. According to these activations, we compute the learned features of the neurons by the method used in (Girshick et al., 2014). We also compute the importance of each neuron. In CNNVis, the importance can be defined in several ways, such as the average or maximal activation on a set of classes of images and the contribution to the final result.

Connections between Neurons. The connections between neurons can reveal how low-level features are aggregated into high-level features. Their strengths are often learned by minimizing the value of the loss function, which is solved by stochastic gradient descent (Bottou, 1991). For a given snapshot, we can get the strengths and gradients by parsing the model file (e.g, ".caffemodel" file for Caffe framework).

3.2. Data Aggregation

To effectively visualize a large CNN, we perform a series of data aggregation operations. We first cluster the layers and select a representative layer from each layer cluster. We then cluster the neurons in the representative layers. Several representative neurons are selected from each neuron cluster. To reduce the visual clutter caused by a large number of connections between neurons, we bundle the connections via biclustering (Sun et al., 2016). A bicluster is a subset of input neuron clusters and a subset of output neuron clusters.

4. Visualization

In this section, we introduce the visualization module of CNNVis.

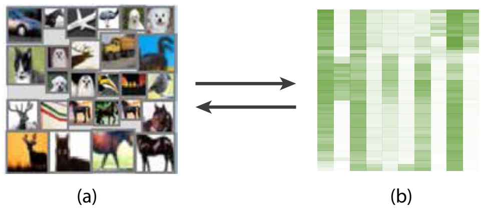

Figure 3. Switching between two facets of a neuron cluster.

4.1. Overview of the Visual Design

In CNNVis, each neuron cluster is represented by a large rectangle. Experts can analyze neuron clusters from multiple facets, such as the learned features, activations, and contributions to the final result. The learned features of neurons are placed using a rectangle packing algorithm (Fig. 3 (a)). The activations of neurons in a cluster are shown as a matrix visualization (Fig. 3 (b)). Each cell in the matrix represents the average activation of a neuron on a class of images. The importance of a neuron is encoded by the size of the rectangle. The connections between neurons are represented by curves connecting two neurons. Please refer to (Mengchen et al., 2016) for more details on how to generate such a visualization

4.2. Interaction

To enable experts to further analyze their models, we provide a set of interactions.

Interactive Data Aggregation. As the data aggregation method can be imperfect and different users may have different needs, we allow experts to interactively modify the aggregation results. For example, experts can drag a neuron to another neuron cluster to modify the neuron clustering results. In addition, experts can manually merge two biclusters.

Exploring Multiple Facets of Neurons. Previous research has focused on visualizing the learned features of neurons. However, other facets of neurons can also help experts analyze the roles of neurons from different perspectives. For example, the activations of neurons can help experts find important neurons for classifying a specific class of images. Thus, we allow experts to switch among different facets. For example, experts can switch to view the activations or the learned features of neurons.

Analyzing a Part of the Neurons. A large CNN may have thousands of neurons in each layer. Thus, it is desirable to allow experts to select a portion of the neurons to analyze. In particular, we allow experts to expand a layer cluster to analyze the neurons in these layers. In addition, we also allow experts to select a set of classes and show the neurons that are strongly activated by the images in those classes. Other neurons are deemphasized by setting them to be translucent.

Examining the Debugging Information. Analyzing the debugging information is a common way for experts to diagnose a failed training process. As there is often a huge amount of debugging information, we allow experts to explore the debugging information at different levels of granularity. For example, experts can change the color coding of edges to examine the gradients of weights. In addition, CNNVis shows the aggregated gradients of each layer as a line chart below. Other derived values such as the relative change of weights can be illustrated in the same way.

5. Case Studies

In this section, we demonstrate the effectiveness and usefulness of CNNVis using two real-world case studies. These two case studies are in conjunction with two domain experts (E_1 and E_2).

5.1. Analyzing Influence of Network Architecture

This case study demonstrates how CNNVis helps a deep learning expert (E_1) better understand the influence of network architecture, including the network depth and width.

The expert employed LeNet (Lecun et al., 1998) as the baseline model. He used the MNIST (Lecun et al., 1998) dataset to train and test the network. He adopted the implementation of LeNet in Caffe (Jia et al., 2014), where the network has 2 convolutional layers and 2 fully connected layers. The model achieved 0.87% error on the test set. In this case study, E_1 evaluated a set of variants of LeNet (with different depths and widths) qualitatively based on his experience.

The expert first examined how the depth of the model influenced the performance and checked the possibility of CNNVis to help him decide an appropriate network depth.

To investigate the performance of deeper models, he replaced each convolutional layer with a stack of 4 convolutional layers. He adopted ReLU as the activation function. The new model was named with DeepLeNet. The comparison between DeepLeNet and LeNet is summarized in Table 1. To investigate why DeepLeNet has a worse performance, E_1 first examined the weights of the first and second group of the convolutional layers. He instantly found that most of the weights in the first group of convolutional layers were positive (Fig. 4 (a)). This was different from LeNet (Fig. 4 (b)).

The expert commented that consecutive convolutional layers whose weights are mostly positive can be approximated with a close-to-linear function. This phenomenon indicated redundancy in the layers. He concluded that such redundancy may make the training process less effective and hurt the performance.

Table 1. Comparison between LeNet and DeepLeNet. "#ConvLayers" is the number of convolutional layers.

	Error	#ConvLayers
LeNet	0.87%	2
DeepLeNet	1.00%	8

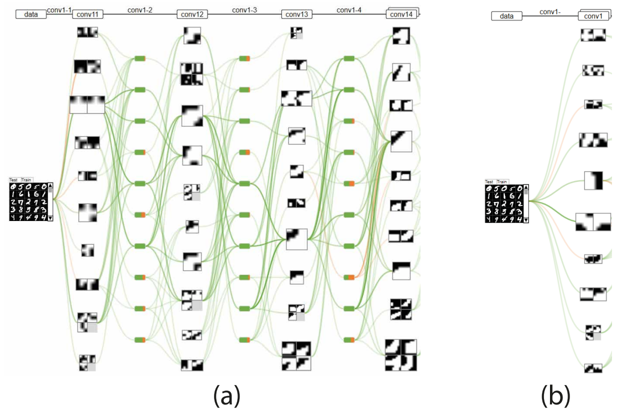

Figure 4. The comparison between LeNet and DeepLeNet: (a) the weights of the first group of convolutional layers in DeepLeNet; (b) the weights of the first convolutional layer in LeNet.

The width of a CNN is another important factor that influences the performance. To evaluate the influence of width comprehensively, the expert compared several variants of LeNet with different widths. The variants are named by LeNet×w, where w is the ratio of the number of neurons in the convolutional layers compared to that of LeNet. For example, LeNet×4 has four times the neurons of LeNet. In this case study, w was selected from $\{4, 2, 0.5, 0.25\}$. The architecture and performance of these variants and LeNet are summarized in Table 2.

Table 2. Comparison between CNNs with different widths.

	Error	Training loss	Testing loss
LeNet×4	0.87%	5.93×10^{-3}	2.55×10^{-2}
LeNet×2	0.86%	6.17×10^{-3}	2.59×10^{-2}
LeNet	0.87%	6.25×10^{-3}	2.78×10^{-2}
LeNet×0.5	1.16%	9.80×10^{-3}	3.41×10^{-2}
LeNet×0.25	1.36%	1.89×10^{-2}	4.60×10^{-2}

Expert E_1 observed that the performance of a wider model (LeNet×4) is comparable with that of LeNet. From the training loss and testing loss, he could not conclude that the wider model is more overfitting than the baseline model (LeNet), quantitatively. Thus, he wanted to qualitatively

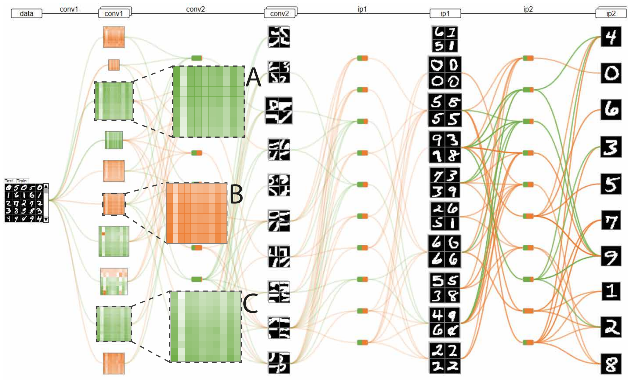

Figure 5. Neurons in a model that is too wide. Many neurons in the first convolutional layer have very similar activations.

examine LeNet×4. From the activations of the neurons in the first convolutional layer, he instantly observed that many neurons have very similar activations (A, B, and C in Fig. 5). The expert inferred that there may be redundant neurons in a model that is too wide. The expert further commented, "CNNVis can help me qualitatively evaluate the quality of a model. It provides a good complement for the quantitative criterion that we often use."

5.2. Interactive Model Refinement

Table 3. Performance comparison between ReLU (BaseCNN) and leaky ReLU with different parameters. "\." indicates that the parameter leads to a failed training process. "err" is short for error and the column that the value of the parameter is zero is the baseline.

	0	0.01	0.05	0.1	0.15	0.2	0.24	0.25	0.3
err(%)	11.45	11.12	10.78	10.35	10.07	10.18	10.11	\	\
std(%)	0.33	0.19	0.23	0.12	0.09	0.16	0.10	\	\

This case study aims to demonstrate how CNNVis facilitates machine learning experts in refining a CNN. Recently, expert E_2 wanted to construct a model to classify the images in CIFAR-10 dataset. Inspired by VGG nets (Simonyan & Zisserman, 2014), he designed a model with 10 convolutional layers and 2 fully connected layers (BaseCNN). The convolutional layers can be organized into 4 groups, containing 2, 2, 3, and 3 convolutional layers, respectively. The number of the neurons of the convolutional layer in each group is 96, 128, 256, and 512, respectively. Each group of convolutional layers ends with a max-pooling layer. This model achieved 11.32% error rate on the test set.

He was not satisfied with the model accuracy and wanted to improve the model. To find a potential direction for improvement, he examined the average relative change of the weights from the first layer to the last layer (Fig. 6A). In the corresponding line chart, the expert immediately identified a large change in the third group of convolutional layers (Fig. 6B). He pointed to this drop and said, "In a high

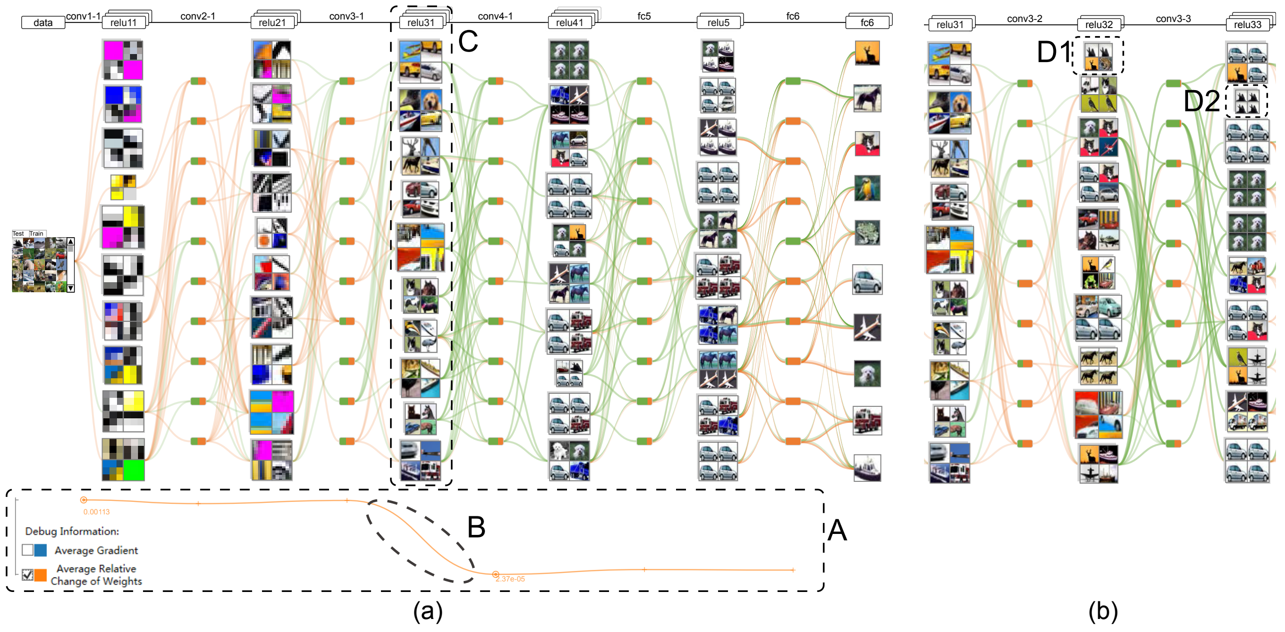

Figure 6. Network refinement. (a) the overview of BaseCNN; (b) Expanding the third group of convolutional layers.

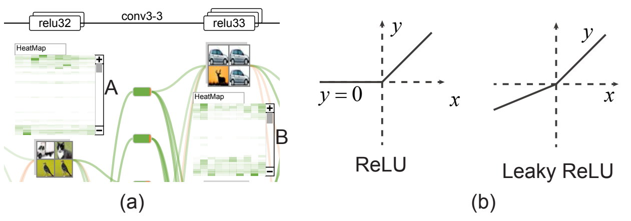

Figure 7. Illustration of activations: (a) the neurons with zero activations in layers relu3-2 and relu3-3 (output of corresponding convolutional layers); (b) illustration of ReLU and leaky ReLU.

quality CNN, the average relative change typically does not behave this way."

To figure out the potential reason for this change, he zoomed into the third group of convolutional layers (Fig. 6C) to examine it in more detail, focusing on exploring the learned features in each layer. His examination revealed that the learned features in these layers (Fig. 6(b)) were mostly parts of an object and had a clear meaning (e.g., an animal's face or a horse's head). He found two neuron clusters had no connections with other neuron clusters (D1, and D2 in Fig. 6). Thus, he switched to examine the activations of these neuron clusters (Fig. 7(a)). Some neurons with zero activations (inactive neurons) on all classes attracted his attention (A, and B in Fig. 7(a)). He further examined the inputs of the ReLUs in these neurons by expanding the layer cluster. He found that the inputs of these ReLUs were always less than zero. The expert pointed out that if the input of an ReLU is less than zero, it generates a zero activation (Fig. 7(b)). The neurons with zero activations make no contributions to the training process. Thus, it makes it more difficult to converge the model.

To solve this problem, E_2 proposed using leaky ReLU (Maas et al., 2013) to replace ReLU as the activation function. E_2 further commented that they usually start from ReLU for a task at hand due to its acceptable performance in most cases. Although Leaky ReLU allows small non-zero gradients when the neuron activity is less than zero (Fig. 7(b)), it is parameter sensitive (Maas et al., 2013). Experts have to try different parameters to find an acceptable performance, which is very time consuming. As a result, leaky ReLU is seldom used in the trial stage. If there are inactive neurons for all the data, it is preferable to try leaky ReLU.

To verify his assumption, E_2 tried leaky ReLU with 8 different parameters ranging from 0.01 to 0.3. To eliminate the randomness in the initialization, we trained each model five times with different random seeds and reported the mean and standard deviation. The performance comparison with BaseCNN is summarized in Table 3. The model with leaky ReLU performed better with most parameters. However, several parameters lead to failed training process (0.25, 0.3).

This case study demonstrates the ability of CNNVis to help experts find the potential limitations of a CNN model, which are hard to find without an interactive visualization toolkit. The finding can further inspire possible directions for improving the network architecture of CNNs. For example, expert E_2 commented, "Unlike an aimless trial-and-error process, this toolkit helps me easily find a potential direction for improvement. This definitely accelerates the model construction process in my research."

6. Implementation Notes

CNNVis was implemented with C++ and C #. The preprocessing module was implemented in C++ based on the Caffe framework (Jia et al., 2014). We leveraged the Caffe framework to run the data through the network and get the activations of neurons. The data aggregation module, user interface, and visualizations are implemented in C#. The images are processed by EmguCV (Korn, 2015), which is a cross platform .Net wrapper to the OpenCV image processing library. We adopted WPF framework to implement the user interface and visualizations.

Currently, we are trying to convert it to an online version with Javascript. Our online version is based on D3 (Bostock et al., 2011), which is a commonly used JavaScript visualization library.

7. Conclusions

We have introduced CNNVis, a visual analysis system that can help experts better understand and refine deep CNNs. The major feature of CNNVis is that it provides experts an interactive visualization to analyze the model from different perspectives. Two real-world case studies have illustrated the effectiveness and usefulness of our system.

References

Bengio, Y., Courville, A., and Vincent, P. Representation learning: A review and new perspectives. *IEEE PAMI*, 35(8):1798–1828, Aug 2013.

Bengio, Yoshua. Learning deep architectures for ai. *Foundations and Trends in Machine Learning*, 2(1):1–127, 2009. ISSN 1935-8237.

Bostock, Michael, Ogievetsky, Vadim, and Heer, Jeffrey. D3 data-driven documents. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2301–2309, 2011.

Bottou, Léon. Stochastic gradient learning in neural networks. *Neuro-Nımes*, 91(8), 1991.

Girshick, Ross, Donahue, Jeff, Darrell, Trevor, and Malik, Jitendra. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, pp. 580–587, 2014.

Jia, Yangqing, Shelhamer, Evan, Donahue, Jeff, Karayev, Sergey, Long, Jonathan, Girshick, Ross, Guadarrama, Sergio, and Darrell, Trevor. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

Karpathy, Andrej, Toderici, George, Shetty, Sanketh, Leung, Thomas, Sukthankar, Rahul, and Fei-Fei, Li. Large-scale video classification with convolutional neural networks. In *CVPR*, pp. 1725–1732, 2014.

Korn, Von Der. Emgucv. `http://www.emgu.com/wiki/index.php/Main_Page`, 2015.

Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. In *NIPS*, pp. 1097–1105, 2012.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *IEEE*, 86(11):2278–2324, 1998.

Maas, Andrew L, Hannun, Awni Y, and Ng, Andrew Y. Rectifier nonlinearities improve neural network acoustic models. In *ICML*, volume 30, pp. 1, 2013.

Mahendran, A. and Vedaldi, A. Understanding deep image representations by inverting them. In *CVPR*, pp. 5188–5196, 2015.

Mengchen, Liu, Jiaxin, Shi, Zhen, Li, Chongxuan, Li, Jun, Zhu, and Shixia, Liu. Towards better analysis of deep convolutional neural networks. *arXiv preprint arXiv:1604.07043*, 2016.

Sainath, T. N., r. Mohamed, A., Kingsbury, B., and Ramabhadran, B. Deep convolutional neural networks for lvcsr. In *ICASSP*, pp. 8614–8618, 2013.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

Simonyan, Karen, Vedaldi, Andrea, and Zisserman, Andrew. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *ICLR Workshop*, 2013.

Sun, M., Mi, P., North, C., and Ramakrishnan, N. Biset: Semantic edge bundling with biclusters for sensemaking. *IEEE TVCG*, 22(1):310–319, 2016.

Szegedy, Christian, Liu, Wei, Jia, Yangqing, Sermanet, Pierre, Reed, Scott, Anguelov, Dragomir, Erhan, Dumitru, Vanhoucke, Vincent, and Rabinovich, Andrew. Going deeper with convolutions. In *CVPR*, pp. 1–9, 2015.

Yosinski, Jason, Clune, Jeff, Nguyen, Anh, Fuchs, Thomas, and Lipson, Hod. Understanding neural networks through deep visualization. In *ICML Workshop on Deep Learning*, 2015.

Zeiler, Matthew D and Fergus, Rob. Visualizing and understanding convolutional networks. In *ECCV*, pp. 818–833, 2014.