Interactive Deep Convolutional Neural Network Visualization

Michael L. Iuzzolino* Shirly Montero Quesada† Athithyaa Panchapakesan Rajeswari‡
Santhanakrishnan Ramani§

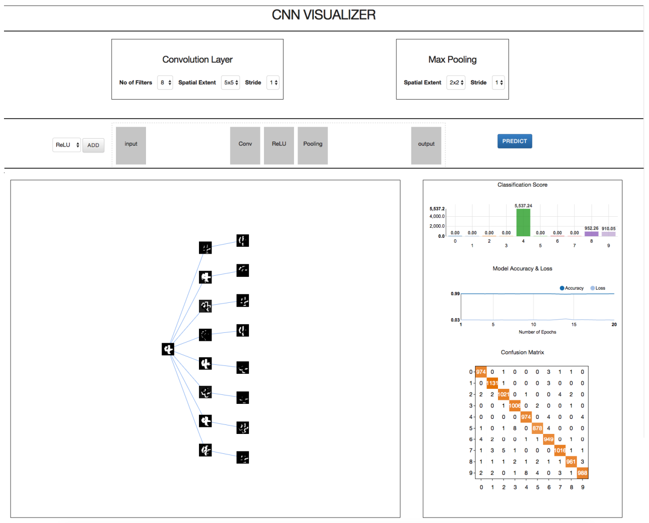
Abstract

Despite the burgeoning success of Deep Convolutional Neural Networks (CNNs) and their pervasion into myriad domains, ranging from art and music to science and medicine, their deep and complex architectures, innumerable parameters, and nonlinear activations render their inner workings nearly incomprehensible. This 'black box' nature of CNNs prohibits not only a robust intellectual understanding for academic endeavors and optimization engineering, but it also restricts these models and their exceptional pattern-recognition power to expert-users. Importantly, the lack of understanding how and why a CNN makes any particular decision also leads to significant legal liability for self-driving automobile companies and medical institutions. Interactive visualizations of CNNs may provide both expert and non-expert users the insight and intuition required to more optimally architect CNNs, as well as foster more confidence in their inner workings or reveal pitfalls to which solutions may be engineered. Based upon a review of current deep neural network visualization techniques, we have developed an interactive visualization of deep CNNs that enables users to interactively construct CNNs of various architectures and adjust the hyper-parameters of the CNN's layers. These models are trained on the MNIST dataset, and intermediate layer feature visualizations are provided, in addition to time-series training data of accuracy/loss per epoch, confusion matrices, and prediction probabilities for all potential classifications.

1 Introduction

Deep Convolutional Neural Networks (CNNs) are immensely powerful pattern recognition models that are utilized for an impressive variety of image recognition tasks. These tasks include image and video classification [3, 16, 27], semantic captioning and scene labeling [13, 35], facial and facial expression recognition [18, 22], electroencephalogram (EEG) classification [30], age and gender classification [20], sentence classification and document analysis [14, 26], in addition to other natural language processing (NLP), cyber security, and human action recognition tasks. CNNs are also used as function approximators in Deep Reinforcement Learning (RL) agents, such as AlphaGo [25], that are so successful that they are capable of beating world-class human Go players, a game considerably more complex than chess. Further, deep RL agents with CNN function approximators are currently being investigated for use in robotics to facilitate natural Human-Robotic Interactions (HRI) in medicine and therapy, production-line assembly, space exploration, and social home robotics [5, 9, 34].

Despite their growing ubiquity in increasingly complex, sensitive, and potentially life-threatening tasks, such as in medicine and self-driving automobiles, CNNs are largely "black-boxes" - their inner workings are difficult to visualize and understand due to their nonlinearity and astronomical dimensionalities. In other words, "the outcome of a Artificial Neural Network (ANN) cannot directly be



Convolution Layer: No of Filters 8, Spatial Extent 5x5, Stride 1
Max Pooling: Spatial Extent 2x2, Stride 1

	0	1	2	3	4	5	6	7	8	9
0	974	0	1	0	0	0	3	1	1	0
1	0	1131	1	0	0	0	3	0	0	0
2	2	2	1021	0	1	0	0	4	2	0
3	0	0	1	1006	0	2	0	0	1	0
4	0	0	0	0	974	0	4	0	0	4
5	1	0	1	8	0	878	4	0	0	0
6	4	2	0	0	1	1	949	0	1	0
7	1	3	5	1	0	0	0	1016	1	1
8	1	1	1	2	1	2	1	1	961	3
9	2	2	0	1	8	4	0	3	1	988

Figure 1: Interactive Deep Convolutional Neural Network Visualization interface. The image is from [12].

interpreted in terms of comprehensible rules" [4, 8]. This presents a significant issue for the domains of theoretic and engineering, but also for companies and medical institutions with respect to legal liability.

First, the uncertainty of the hows and whys of a neural network lead to a slew of time-consuming guesswork in building and refining model parameters. This results in CNN training times that generally span days or even weeks. Further, this esoteric nature of CNNs has relegated their power and range of possible applications to a small domain of expert users; importantly, even expert users are naive to why certain CNN architectures work so well nor is there a clear understanding of how to make improvements [37]. Secondly, two critical application domains for CNNs are transportation (self-driving automobiles) and medicine. However, even with an operable, efficient CNN system in hand, without a clear understanding of how these CNNs are operating and producing their results, a slew of legal liabilities encumber their implementation, which stagnates commercial progress and, ultimately, human and society progress and well-being.

As CNNs grow in complexity, depth, and ubiquity in all domains of society, the need for transparency and insight becomes paramount. For example, modern CNNs, such as GoogLeNet [31] and ResNet [10] have convolutional depths of 22 and 152 layers, respectively. GoogLeNet was the winner of the ILSVRC 2014, with a top 5 error rate of 6.7%; even more impressively, Microsoft's ResNet won the ILSVRC 2015 with an error rate of only 3.6%. Despite these monumental accomplishments, there remain only vague heuristic insights for architecting these models and even less of an understanding of how exactly they are operating. Further, Nguyen et al. have demonstrated that Deep neural networks can be easily fooled, leading to considerable security concerns, amongst others [24]. Consequently, there is dire need for intuitive visualization tools that cultivate a rigorous and reliable understanding of CNNs and how

*e-mail: michael.iuzzolino@colorado.edu
†e-mail: shirly.monteroquesada@colorado.edu
‡e-mail: athithyaa.panchapakesanrajeswari@colorado.edu
§e-mail: santhanakrishnan.ramani@colorado.edu

exactly they function and arrive at their results. What will follow from this insight are optimized solutions and training times, accessibility to a broad range of non-expert users, the mitigation of prohibitive liability issues, and ultimately, the driving forward of societal progress.

2 Related Work

Deep CNNs are difficult to visualize for a number of reasons. First, as their name suggests, they are deep. The term ’deep’ means that the architecture consists of many layers of nodes (hidden layers) between the input and output layers of the network. For example, a common toy CNN that is trained on CIFAR-10 [17], a dataset consisting of 60,000 32x32 color images in 10 mutually exclusive classes, typically has 6 convolution layers resulting in a total number of parameters on the order of 700,000. Put another way, the dimensionality of the datasets are extraordinary and not readily distilled into comprehensible visualizations. Secondly, once the information propagates beyond the first layer of the network and into the hidden layers with non-linear activation functions - typically RELU - a non-linear model is established along with the accompanying complexity and lack of interpretability. This non-linearity and high dimensionality of these models, along with massive data sets and computing power, is what imbues them with such staggering power as approximators and pattern detectors.

2.1 Heat Maps

2.1.1 Layer-wise Relevance Propagation

A number of innovative approaches have been taken to visualize deep neural networks (DNNs) with some modicum of success. One recent approach is a method called Layer-wise Relevance Propagation (LRP) [1]. LRP, stemming from a method called Deep Taylor Decomposition [23], computes scores for an image’s pixels and groupings of pixels that most strongly contribute to the classification and reconstructs the input image as a heat map based upon the scores. This yields a visualization in terms of the original image and the features that are being captured by the network and utilized most heavily for classification. Although this sheds some transparency on the network’s functionality and it being quickly picked up by the cognitive science community for applications to EEG analysis, it does not reveal the full extent to how the network is making decisions and at what level of the information processing it is occurring. In addition, the method’s results are highly sensitive to the algorithm’s parameters, ε and β, and an optimal tradeoff between numerical stability of the decomposition and meaning of the heatmap has yet to be achieved.

2.1.2 Hinton Diagrams

Similar to the LRP heatmaps, the Hinton diagram visualizes the parameters of a neural network layer as a grid of squares in various sizes and colors [2, 11]. The parameters of a neural network are the weights of the links (also known as edges or synapses) connecting the nodes of the layers; these weights take on real-valued numbers and can consequently be encoded into a visualization that takes advantage of both color, size, and spatial channels. A Hinton diagram maps the magnitude of the parameter onto the size channel, the sign of the parameter onto the color channel - white for positive values and black for negative, and spatially arranges the marks such that the rows correspond to the nodes of the sending layer and the columns correspond to the nodes of the receiving layer of the connection matrix. Although this is a first attempt at visualizing the internal layers of the network, better approaches have been developed that we will talk about in the next section.

2.2 Intermediate Layer Visualization

An alternative approach to heat mapping is a tool that dynamically visualizes the activations produced at each layer of a trained CNN as it processes an image or video [36]. Accompanying this visualization is a second visualization tool that visualizes features at each layer of the DNN. These interactive tools have shed additional transparency on CNNs, revealing that deeper CNN layers tend towards detection of ’local’, specific features of an image; for example, the wheel of a car of the face of a person. Although the details are beyond the scope of this paper, these locality revelations shed additional insight onto two other opaque characteristics of neural networks - transfer learning and the ’hacking’ of discriminative networks via generative models that produce structured noise [6, 7, 32, 33].

Other approaches have visualized the intermediate feature layers and the operation of the classifier, but were used with slightly different intentions. For example, Zeiler et al. used their intermediate feature layer visualization tools in a diagnostic role to make comparisons between models, as well as ablation studies to obtain performance measures from different model layers [37]. The visual comparison of models may prove highly effective in fostering another aspect of CNN intuition and understanding via intrinsically strong human learning mechanisms that utilize comparisons and contrasts as a way of cultivating knowledge about the world and it’s myriad objects. Yet another group approached deep network visualizations from another angle: hierarchical rectangle packing algorithms and a matrix reordering algorithm to show the derived features of a neuron cluster [21]. As alleged by the authors, this approach facilitates a better understanding of CNNs and therefore, it supports the machine learning experts during the refinement of the CNN architecture towards the improvement of performance – as opposed to only the trial-and-error strategy.

2.3 Interactive Neural Network Visualization

Lastly, Daniel Smilkov and Shan Carter, inspired by Andrew Karpathy’s ConvnetJS demo [15], have approached the visualization of deep neural networks pedagogically with their TensorFlow (TF) Playground [28]. TF Playground enables a user to adjust the hyper parameters of customizable DNN in-browser, loading various datasets, apply noise, set activation functions and learning rates, and a number of other features, all whilst enabling the user to then train their customized network and view the results in real-time. This approach provides users, expert and non-expert alike, with valuable insight into the various architectures and parameters and how the functionality of the model is effected. Using TF Playground as conceptual inspiration, we have developed a tool that enables a user to explore CNNs in a similar manner to how TF Playground explores DNNs, which we will describe in detail below.

3 Approach

Using TF Playground as an inspiration, we have developed an interactive visualization of convolutional neural networks using Python-Keras, Flask, and D3.js (Fig. 1). Based on the literature in the topic and our discussion with an expert in the field of deep learning and neural networks (Mozer, M. (2017, May 5). Personal interview.), we identified our first requirement that our visualization should render information on the learned features. In this way, our visualization could facilitate the mental representation of the neuron’s learning process. Our tool enables users to adjust the CNN’s convolutional and max-pooling layer hyper-parameters; this includes the number of filters per convolutional layer, spatial extent (i.e., the filter’s dimensionality), stride, and zero padding. Additionally, the user is provided with the ability to add layer units - convolutional, RELU, and MaxPool layers - and arrange them in a desired ordering via drag-drop interactivity. The user may then train their customized CNN on the MNIST dataset and obtain their CNN’s accuracy and loss scores, confusion matrix, and, importantly, visualizations at each of the intermediate layers that provide insight into what feature each filter of the layers have learned to detect. For example, the filters in the first convolutional layer may learn to detect edges whereas filters in deep layers may learn to detect more complex features, such as specific curvatures and complex patterns. Each of these aspects will be explained in greater detail in the proceeding sections.

3.1 CNN Models

Our visualization was dependent upon six CNN models. The base CNN architecture was predicated on the LeNet-5 model, and each model was trained for 20 epochs with a batch size of 128 on the MNIST dataset [19]. The architecture, dataset, and implementation details will be detailed below.

Architecture

A CNN consists of the following primary layers types: Input, Convolutional, RELU, MaxPooling, Dropout, Fully Connected (FC), and SoftMax. In the LeNet-5 architecture paradigm, Convolutional layers are followed by RELU and then a MaxPooling layer. Multiple convolutional layers may succeed one another, and the MaxPooling layer must be preceded by RELU. The convolutional layer is comprised of multiple filters - these filters are comprised of the parameters that are learned during training via back-propagation. The convolution layer is governed by four hyper-parameters: number of filters, spatial extent, stride, and zero padding, as alluded to in the section above.

Next, the RELU layer is the activation layer that enables non-linear modeling, and the MaxPooling layer may be viewed as an information compression layer. The MaxPooling layer is governed by two hyper-parameters: spatial extent and stride. These hyper-parameters dictate the extent of the information compression. The Dropout layer carries out a form of regularization, introduced by Srivastava et al., that prevents the model from overfitting and has an analogous function to ensemble learning [29]. The FC layer flattens the 2-dimensional layers into a 1-dimensional vector, allowing an onto mapping to the SoftMax layer, which contains as many nodes as there are classifications of the input data. The SoftMax layer calculates the probability for each possible classification of a given input, and renders a histogram containing those probabilities for all the classifications. The reason we chose not to visualize the prediction as an image on the tree is two-folded. First, when a prediction is obtained, the resulting images will be binary and not very informative. For example, the prediction could be black for the resulting ArgMax(SoftMax) and white for all the rest of the classifications. The second reason was that the image will be too cluttered, showing the connectivity of the nodes of the last layer with all the classifications.

Dataset and Training

Our models were trained on the MNIST dataset (Fig. 2), originally selected and experimented with by Chris Burges and Corinna Cortes, and further developed and popularized by Yann LeCun with LeNet-5. The database consists of a training set of 60,000 hand-written digit images, followed by an additional 10,000 for use as a test set, all with a dimensionality of 28x28x1. Keras' dataset library was utilized to obtain MNIST. The models were trained on the MNIST dataset for 20 epochs each, withholding 10% of the data for validation, and with a batch size of 128. For 20 training epochs, typical training times for a CNN on the MNIST dataset were on the order of an hour, depending upon the depth of the CNN architecture, with each epoch taking approximately 300 seconds to train and validate.

3.2 Model Visualizations

Design and Implementation

A key aspect of the interactive CNN visualization tool is the rapid generation of various user-decided CNN architectures and immediate visualizations, similar to what TF Playground affords. Given the extensive training times of CNN models, especially for those with deep

Figure 2: MNIST Dataset

architectures, two important design choices were required. First, we needed to consider how many hyper-parameters and architecture configurations we could allow the user to control. Each additional hyper-parameter exponentially increases the time-complexity, along with the space-complexity, given that each model's weight file for each epoch takes up approximately 20MB of space. A quick calculation of the training time required for six architecture configurations with all six hyper-parameter options, each with only two levels, resulted in a training time greater than 10 days, and more than a Terabyte of required storage space for model and weight files. These calculations were not inclusive of the additional image files that were generated at each epoch for every filter within all convolutional layers and the images at the RELU and MaxPooling layers, nor was it inclusive of file write-times.

Second, this required a design decision pertaining to whether optimization techniques during model training should be used to decrease the training times, such as callback checking for early stopping when models achieve acceptable loss or accuracy scores, or whether we needed to pre-train and store the models server-side for rapid retrieval. The space-complexity issue could be avoided in the former case to some degree, but provides a significant challenge for the latter approach.

To address these issues, we decided to pre-train and store the models for an optimal solution. This required that we constrain the user's choice of hyper-parameter settings to the convolutional layer's kernel size: 3x3 or 5x5. The architecture configurations were confined in the following way. The CNN model layering was formatted as *Input, X, RELU, Dropout, SoftMax*. Here, *X* is the architecture space in which the user has design control. The user may add three types of layers in this space, *X*: Convolutional, RELU, and MaxPool. Additionally, a Dropout layer was implicitly appended to each of the user's MaxPool inclusions. The user is limited to three configurations in *X*:

- Convolution, RELU, MaxPool
- Convolution, RELU, Convolution, RELU, MaxPool
- Convolution, RELU, MaxPool, Convolution, RELU, MaxPool

Consequently, this results in the training of six distinct CNN models: each of the above three architecture configurations coupled with each level of the kernel size hyper-parameter selection.

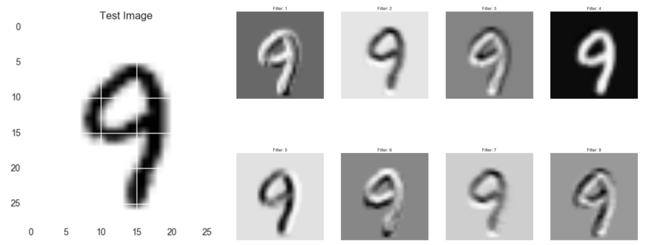


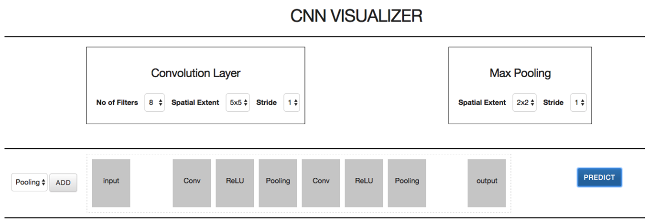
Figure 3: Left: The test image fed to a trained CNN. Right: The eight filters extracted from an intermediate convolutional layer of the network. Each filter has learned to detect a different feature of the input data; edges, hues, shadings, textures, etc.

Intermediate Layer Visualization

Intermediate layer visualization, such as depicted in Fig. 3, was accomplished via segmentation of the models from the input feed up to the intermediate layer of interest. Images were randomly drawn from the test set, and the sub-models were used to predict the output of the test image. The resultant outputs were two dimensional weight matrices that, when imaged, visualize the respective features that were learned during training. This is a crucial step in providing insight into the inner-workings of a CNN, in a similar fashion to TF Playground's output visualizations where an immediate classification is presented pending the variation of hyper-parameters. That is, by tweaking one parameter, an immediate effect can be observed on the outputs of the model.

Interactive Visualization Interface

The interactive visualization in Fig. 1 is comprised of two primary panes. The top pane contains the hyper-parameter selection controls for the convolution and maxpooling layers overlaying the network architecture configuration area. In this area, the user may explore various CNN architecture configurations by adding new layers (Conv, ReLU, or Pooling), followed by drag-dropping them into the desired configuration Fig. 4. Once a CNN is constructed, the user may then explore the properties via clicking the predict button, which will recover the trained model files, in addition to randomly choosing an image from the MNIST test set.



CNN VISUALIZER

Convolution Layer

No of Filters 8 Spatial Extent 5x5 Stride 1

Max Pooling

Spatial Extent 2x2 Stride 1

Figure 4: Interactive Control Panel

The generated network will then be displayed in the bottom pane, which is further divided into two sections. The left-hand side contains the trained CNN, with the input layer leading into the output layer from left to right. The randomly chosen MNIST test image is visualized at the input node, whereas the nodes in the successive layers represent the learned filters of the particular intermediate layer (Fig. 5).

The right-hand section of the bottom pane offers visualizations of the various aspects of the trained CNN. From top to bottom, the classification scores for the given random test set image are displayed. This is followed by a time-series visualization of the network's training loss and accuracy over a period of 20 training epochs. Lastly, a confusion matrix of the final network training state is visualized, enabling a sense of model accuracy (Fig. 6). For example, a diagonal matrix as seen in Fig. 6 provides an immediate insight into the model's high accuracy.

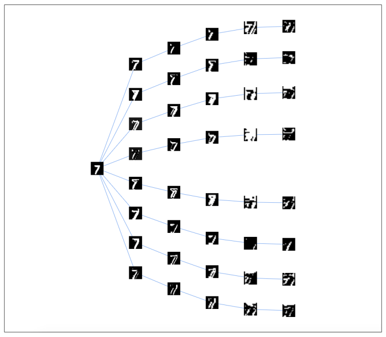
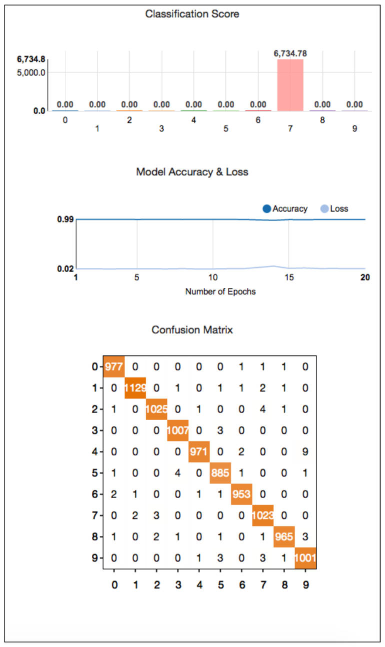


Figure 5: Network Visualization - Intermediate Layer Filter Imaging



	0	1	2	3	4	5	6	7	8	9
0	977	0	0	0	0	0	1	1	1	0
1	0	1129	0	1	0	1	1	2	1	0
2	1	0	1025	0	1	0	0	4	1	0
3	0	0	0	1007	0	3	0	0	0	0
4	0	0	0	0	971	0	2	0	0	9
5	1	0	0	4	0	885	1	0	0	1
6	2	1	0	0	1	1	953	0	0	0
7	0	2	3	0	0	0	0	1023	0	0
8	1	0	2	1	0	1	0	1	965	3
9	0	0	0	0	1	3	0	3	1	1001

Figure 6: CNN Statistics and Accuracy Visualizations

3.3 Summary

All six variations of the CNN models were pre-trained locally and stored to the server for rapid access and visualization. To accomplish this, the architecture and weight files of each model's final training epoch were saved with an additional loading and generation script. After the user creates an architecture, sets the hyper-parameters, and clicks generate, the respective model architecture and weight files are retrieved and loaded by the generation script. The model is segmented to generate the intermediate-layer feature detections of a test image, and the images are visualized on demand. Further, during the pre-training, each model's accuracy-loss scores are tracked and recorded for every epoch. This enables the time series data to be presented of the model's training dynamics. Lastly, confusion matrices are generated and stored at the end of the pre-training sessions for rapid retrieval and visualization of the matrix, providing the user with another metric for their constructed model's accuracy.

4 Future Work

As presented here, our visualization encompasses six possible models: three architecture configurations coupled with each level of the kernel size. However, due to the flexibility of the platform, these limitations could be pushed further to include more filters for the pre-existing configurations and eventually, more configurations, and more kernel sizes. A follow up of this work would be the validation of the design. Ideally, we would test the design with subjects that are interested in machine learning, specifically CNNs, but with little knowledge of the techniques. Our visualization does not include an introduction of any kind, excluding subjects with zero knowledge of CNNs. However, the flexibility of the design allows for an extension, for example, a clickable button on the top right with a link to a learning bite; E.g. a video clip of maximum 2 minutes in which the big picture is introduced with common or needed terminology used in the rest of the vis.

Acknowledgments

The authors wish to thank Dr. Danielle Szafir and Dr. Michael Mozer.

References

[1] A. Binder, S. Bach, G. Montavon, K.-R. Müller, and W. Samek. Layer-wise relevance propagation for deep neural network architectures. In *Information Science and Applications (ICISA) 2016*, pp. 913–922. Springer, 2016.

[2] F. J. Bremner, S. J. Gotts, and D. L. Denham. Hinton diagrams: Viewing connection strengths in neural networks. *Behavior Research Methods*, 26(2):215–218, 1994.

[3] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *Icml*, vol. 32, pp. 647–655, 2014.

[4] R. Dybowski and V. Gant. *Clinical applications of artificial neural networks*. Cambridge University Press, 2001.

[5] M. Gombolay, X. J. Yang, B. Hayes, N. Seo, Z. Liu, S. Wadhwania, T. Yu, N. Shah, T. Golen, and J. Shah. Robotic assistance in coordination of patient care. In *Proceedings of Robotics: Science and Systems (RSS)*, 2016.

[6] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

[7] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[8] A. Hart and J. Wyatt. Evaluating black-boxes as medical decision aids: issues arising from a study of neural networks. *Medical informatics*, 15(3):229–236, 1990.

[9] B. Hayes and B. Scassellati. Autonomously constructing hierarchical task networks for planning and human-robot collaboration. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pp. 5469–5476. IEEE, 2016.

[10] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

[11] G. E. Hinton and T. Shallice. Lesioning an attractor network: Investigations of acquired dyslexia. *Psychological review*, 98(1):74, 1991.

[12] M. Iuzzolino, S. Montero Quesada, A. Panchapakesan Rajeswari, and S. Ramani. Interactive cnn visualization.

[13] J. Johnson, A. Karpathy, and L. Fei-Fei. Densecap: Fully convolutional localization networks for dense captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4565–4574, 2016.

[14] N. Kalchbrenner, E. Grefenstette, and P. Blunsom. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*, 2014.

[15] A. Karpathy. Convnetjs, 2012.

[16] A. Karpathy and L. Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3128–3137, 2015.

[17] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images, 2009.

[18] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back. Face recognition: A convolutional neural-network approach. *IEEE transactions on neural networks*, 8(1):98–113, 1997.

[19] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[20] G. Levi and T. Hassner. Age and gender classification using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 34–42, 2015.

[21] 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*, 23(1):91–100, 2017.

[22] M. Matsugu, K. Mori, Y. Mitari, and Y. Kaneda. Subject independent facial expression recognition with robust face detection using a convolutional neural network. *Neural Networks*, 16(5):555–559, 2003.

[23] G. Montavon, S. Lapuschkin, A. Binder, W. Samek, and K.-R. Müller. Explaining nonlinear classification decisions with deep taylor decomposition. *Pattern Recognition*, 65:211–222, 2017.

[24] A. Nguyen, J. Yosinski, and J. Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 427–436, 2015.

[25] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[26] P. Y. Simard, D. Steinkraus, J. C. Platt, et al. Best practices for convolutional neural networks applied to visual document analysis. In *ICDAR*, vol. 3, pp. 958–962. Citeseer, 2003.

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

[28] D. Smilkov and S. Carter. Tensorflow playground, 2016.

[29] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[30] I. Sturm, S. Lapuschkin, W. Samek, and K.-R. Müller. Interpretable deep neural networks for single-trial eeg classification. *Journal of Neuroscience Methods*, 274:141–145, 2016.

[31] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–9, 2015.

[32] C. Szegedy, A. Toshev, and D. Erhan. Deep neural networks for object detection. In *Advances in Neural Information Processing Systems*, pp. 2553–2561, 2013.

[33] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[34] M. Vázquez, A. Steinfeld, and S. E. Hudson. Maintaining awareness of the focus of attention of a conversation: A robot-centric reinforcement learning approach. In *Robot and Human Interactive Communication (RO-MAN), 2016 25th IEEE International Symposium on*, pp. 36–43. IEEE, 2016.

[35] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pp. 2048–2057, 2015.

[36] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579*, 2015.

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