Artificial Neural Networks: Structure and Behavior

Kendall Lowrey, Thanh Tam Nguyen

Abstract—Artificial Neural Networks have traditionally been treated as black boxes, both in their development and in their use. We present a method to discover the internal structure of neural networks by visualizing activation properties of the network with respect to input data: co-activations of multiple neurons. Our method combines statistical analysis techniques with a modified Sankey Diagram to show flow of data through neural networks unlike previous visualizations methods. Implications for this technique beyond behavioral and structural visualization include the optimization of an artificial neural network through parameter reduction and further understanding of their processing.

1 Introduction

Neural networks, both artificial and biological, are layers of individual nodes that are used to estimate or approximate functions operating on some input data. The function they approximate are trained or learned in similar ways: neurons that behave in the intended way are preserved while neurons that produce errors are adjusted. In biological networks, this training takes the form of forming stronger connections between neurons of different layers within a network such that when examined, the behavior or the network is directly related to its structure. Artificial networks, however, are trained on vast amounts of data, usually more data and faster than a human may comprehend. As a result, artificial neural networks (ANN) are often treated as a black box in their usage and study [2].

As ANNs are merely layers of nodes acting as matrix transformations, it may be intuitive to visualize them by laying out the nodes in vertical groupings with lines as their connecting arithmetic functions, as in figure 1. While this is indeed the 'structure' of the neural network, it says nothing to its behavior or function. Extensions to this idea may elucidate more complicated neural networks where layers have non-linear function behavior. Unfortunately, these representations still do little more than descriptive wording or pseudo-code to explain their behavior.

Historically, Hinton or Bond diagrams were used to abstractly represent neural networks and their neuron weights [7]. These methods have a higher learning curve and only work for small networks; modern neural networks are reaching a size that includes billions of parameters.

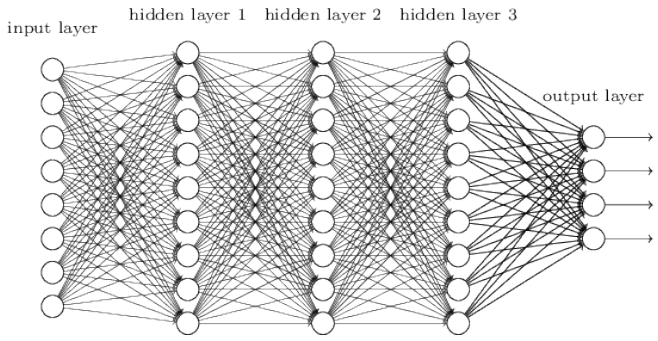

Fig. 1. Simple Neural Network Visualization

Recent developments in ANNs, specifically in using Convolutional Neural Networks [5], offer a different visualization method due to their inherent use of images as their data inputs. Successive layers of transformations in the neural network take the input image and modify it such that the final layer can make a safe decision as to what the input image contains. This requires the convolution of simple image primitive kernels over the input image. These kernels capture image features such as horizontal or vertical lines, points, and gradients, among others. While it is fascinating that these kernels are trained in the convolutional neural network from just training data alone, it is left to the user's imagination to integrate them among the entire network's behavior. Modern software tools may assist in this endeavor, but serves as validation of the networks trained behavior more than anything [3].

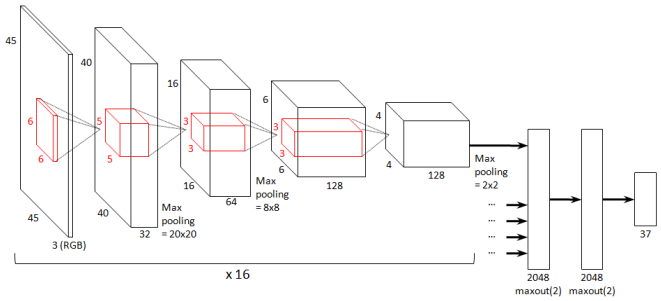

Fig. 2. Convolutional Neural Network

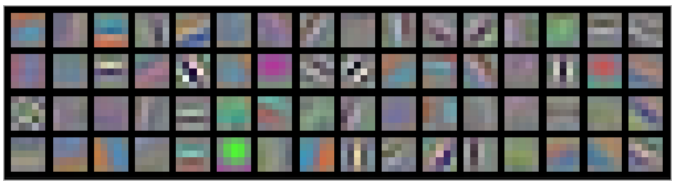

Fig. 3. Convolution primitives

Other than layers and their transforms, a neuron's connections and weights can be visualized in a number of ways. Other than by coloring the vertices's in a graph layout (which would quickly become crowded or unmanageable), a superimposed matrix can be used to show how a group of neurons are interconnected [1] [4]. This representation allows for a quick understanding for how the static properties of a network may be visualized, and is conveniently two dimensional.

In short, neural network visualizations frequently represent the definition of a particular network as nodes in a connected graph, specific layer's transformations on their inputs, or highlighting the connections of the graph and represent this data in a variety of ways.

2 Co-activation Visualization Overview

The approach we take to glimpse the the internals of the black box are to consider the behavior of the network's components in the presence

- *GitHub: klowrey E-mail: klowrey@cs.washington.edu*
- *GitHub: tam-nt E-Mail: nttam15@uw.edu*
- *University of Washington, CSE 512, Spring 2015*

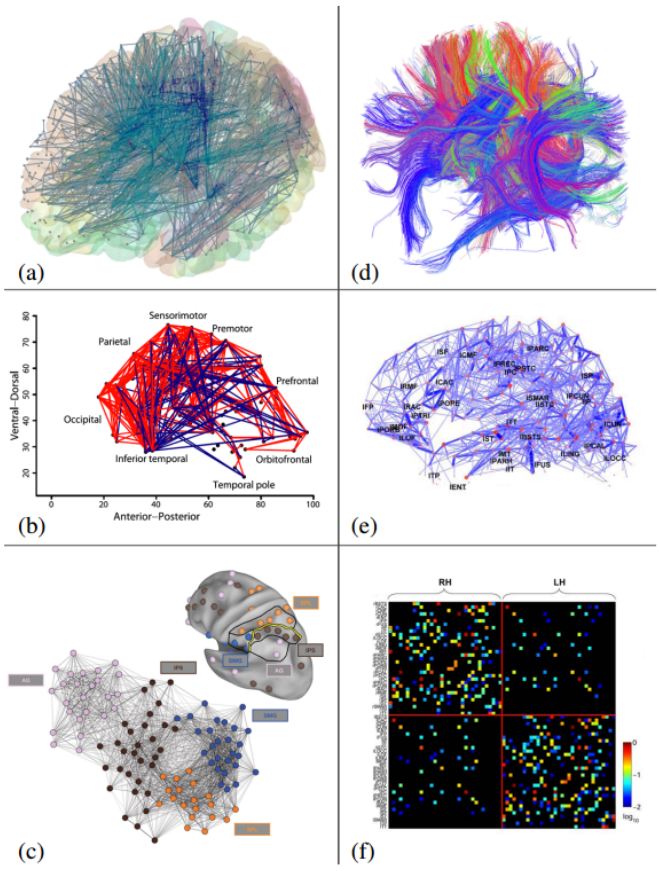

Fig. 4. Brain Connectivity Analysis. Alper et al's version, denoted by (f) presents the most uncluttered and simplest view of the same data.

of data input over time. Neurons in the network, when excited due to input, activate. Groups of neurons that activate at similar times are said to co-activate in response to an input. By measuring when the neurons co-activate and when they do not, we can study how the network is expected to behave.

The activations of the neurons in a network are considered to be non-linear – after all, an ANN should be able to approximate any function. As such, we need to consider dimensionality reduction techniques that can find a reduced space of the neuron's activation data. Activation data, in this case, is the neuron's output value in response to the network receiving some input.

The non-linear dimensionality reduction technique we choose to use is t-Distributed Stochastic Neighbor Embedding (t-SNE) [6]. t-SNE is able to reduce data to fewer dimensions without the constraint of linear variance like Principle Component Analysis.

2.1 Data Generation Methods

The crux of this method is the data generation and collection methods; without the proper data this visualization would be impossible. We first presume that a neural network structure has already been developed and is capable for use in the intended function approximation space, and data that can be used to test the network is available. This is critical because in the primary usage of this method, we need a general representation of the behavior of the network: this suggests using a networks test-set of data (as opposed to the training set used to properly configure the network). We expand on alternate methods in the Future Works section.

Presented here were three types of networks, two of which were trained and had their data collected with Javascript. The third was related to robotics control algorithm and trained using 30 Amazon Web Service's EC2 compute instances, with neural network running on a GPU for higher performance.

Data collection begins with normal test routine where each test input is passed through the network, resulting in every neuron computing its post-synaptic activation value based on its weights and inputs. We specify the post-synaptic value as some neural network implementations split a network's layers into pre- and post- synaptic transformations. Instead of examining the output of the network, we collect each neuron's activation value as our visualization data set. At the end of the test set of inputs, we have a vector of activation values for each neurons that correspond to the number of inputs.

Each neuron's activation weights are then treated as a highly dimensional feature vector. This feature vector represents the behavior of the individual neuron within the network – we now wish to understand this neuron's behavior within the framework of the entire network. We use tSNE as both a dimensionality reduction technique and visual grouping method. Without discussing the details of tSNE too much, we manually adjust the perplexity value for the tSNE algorithm to achieve a sufficient grouping of neurons. The default for the perplexity value was set at 30.

The tSNE algorithm for each network presented here was run in whichever platform was most convenient for the given network as creating new support software was avoided. All implementations of the algorithm performed the same, whether it was in Matlab, Python, or Javascript.

Some caveats to directly reducing the feature vector with tSNE include what kind of data was presented to the network, and which neurons are included in the optimization. First, to collect data representative of the network's behavior, we need to ensure a range of inputs representative of what it was trained to process is presented. Secondly, different networks have different types of input data. Image processing networks take each color channel of each pixel in an input image as an input neuron – this quickly overwhelms the tSNE optimization and prevents reduction of the features into co-activation groups. As such, whether input neurons and output neurons are included in the tSNE optimization is decided on a network by network basis.

2.2 Visualization Methods

After our data is collected, we begin our visualization procedure. We wanted to preserve the traditional graph form visualization of neural networks much like 1 as a homage and convenient frame of reference for those familiar with such visuals. As such, we visualize the network as a series of vertical columns of nodes where each column is a layer, and each node is a neuron within the layer. The inputs to the network exist on the left of the visual, with fully connected lines linking each layer to the next until the output neurons on the right of the visual.

Within each layer, we plot a neuron based on its output from the tSNE step, normalized between zero and one (representing the top and bottom of the rectangular visual area). This allows for a direct translation of the similarity data from tSNE to be visually represented. To represent the relationship between layers, the distance between neurons is encoded as a darker and wider link between the layers. For some networks with many neurons per layer, the links were cut off below a threshold for both aesthetic and performance reasons. As each neuron links to every neuron in the previous layer and every neuron in the next, the links can quickly blacken the space between layers and not convey information. Thresholding level is chosen empirically to convey only the strongest co-activations. This ended up being around 5,000 links as the upper limit of d3js's ability to load and render that quantity of data as well.

This visualization was also inspired by Sankey Diagrams and their method of representing flow between nodes through links, however we modified away from a naive Sankey due to our specific visualization needs.

3 Results

We collected data from networks trained to approximate the Exclusive OR (XOR) function, recognize hand-written numerical digits with the MNIST dataset, and control robots in interactive walking and running tasks. While each network's results are discussed in more detail

in their figure's captions, we can briefly overview the results. This method of neural network structure produces visualizations that suggests the underlying structure of a neural network indeed affects its function. These results lets us make decisions and judgements as to a network's ability to function well.

3.1 Conclusion and future work

We consider this a proof of concept into opening the black box that is a neural network and suggest more avenues of research. From what we understand, this kind of holistic network view has not been attempted before with success, and may prove useful in rendering the behavior of networks for researchers.

The most obvious next step would be to visualize the activations of the network after it has been structure with this method. As discussed in 8, certain areas of this visualization may activate when specific input conditions are met. Visualizing the effect of these input conditions would be an excellent addition to this method, but requires network specific support software to provide this functionality. Our robot control network and training software would have to be converted into Javascript, for example. For the MNIST network, we could see which neurons were activated in response to similar inputs, discovering which neurons are responsible for specific image features.

Another functional option for this research would be to visualize the network as it is being trained. We have discussed that an untrained network is indeed worth visualizing, but for very long training runs, we could use this method to give the researcher feedback beyond just a loss function metric. Visualizing the whole network during training could give insights into which parameters to tweak to sufficiently generalize the training network beyond what a singular loss value could provide.

References

[1] B. Alper, B. Bach, N. Henry Riche, T. Isenberg, and J.-D. Fekete. Weighted graph comparison techniques for brain connectivity analysis. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 483–492. ACM, 2013.

[2] J. M. Benítez, J. L. Castro, and I. Requena. Are artificial neural networks black boxes? *Neural Networks, IEEE Transactions on*, 8(5):1156–1164, 1997.

[3] D. Bruckner. Ml-o-scope: a diagnostic visualization system for deep machine learning pipelines. Technical report, DTIC Document, 2014.

[4] S. Hinze and U. Schmoch. Opening the black box. In *Handbook of quantitative science and technology research*, pages 215–235. Springer, 2005.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[6] L. Van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(2579-2605):85, 2008.

[7] J. Wejchert and G. Tesauro. Visualizing processes in neural networks. *IBM Journal of Research and Development*, 35(1.2):244–253, 1991.

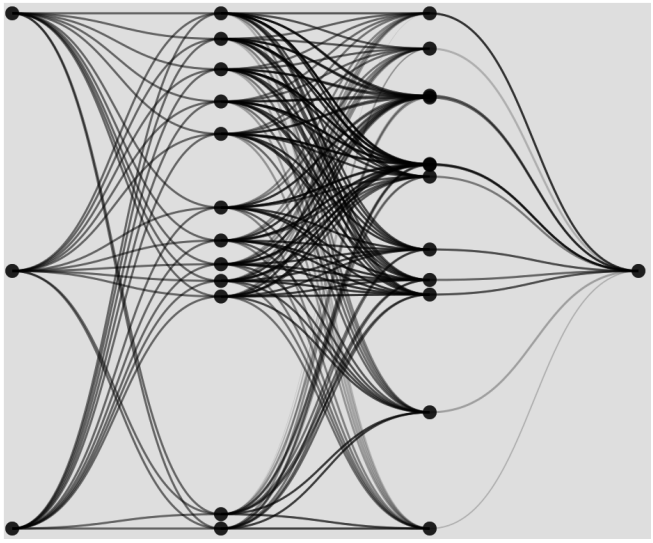

Fig. 5. The untrained XOR network. The network consists of two layers of 12 neurons each. As the network is untrained, we can see that there is very little visual clustering after the tSNE processing. Also, the links between the layers are almost uniformly dark. We take this to mean that the neurons are activating randomly due to the random weights they are initialized with: there is no structure to observe. This is excellent to know because we can begin network training knowing that the network has been initialized with a high degree of randomness – training will be more likely to be successful.

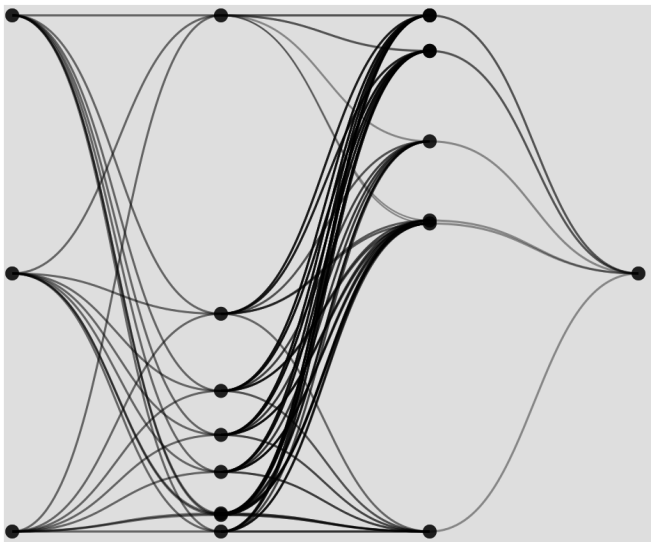

Fig. 6. The trained XOR network features distinct groupings both within a layer and between layers, suggesting that learning has taken place and thus affected the network's structure. While learning can be verified numerically, we can use the visualizations to experiment further with the network beyond numerical results. For instance, we can observe neuron groupings and reduce the number of neurons potentially making the network more efficient.

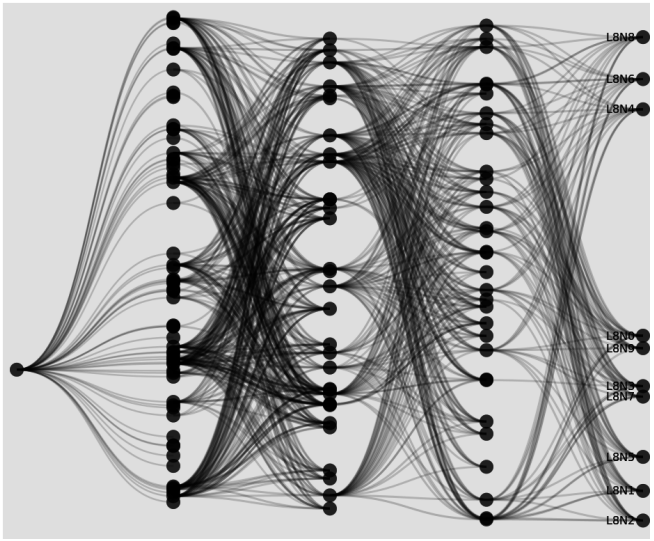

Fig. 7. The MNIST network of hand-written digits. There are many groupings throughout the network of 64, 32, 32 neurons, before a softmax layer to the 10 different classes. We can see groupings in the output layers that reflects some of the mis-predictions other networks may face with the MNIST dataset. We also observe that through the link threshold there are some neurons that do not significantly contribute to the next layer; this could be cause to remove them from the network much like the neuron group consolidation idea demonstrated by the simplified XOR model in 10

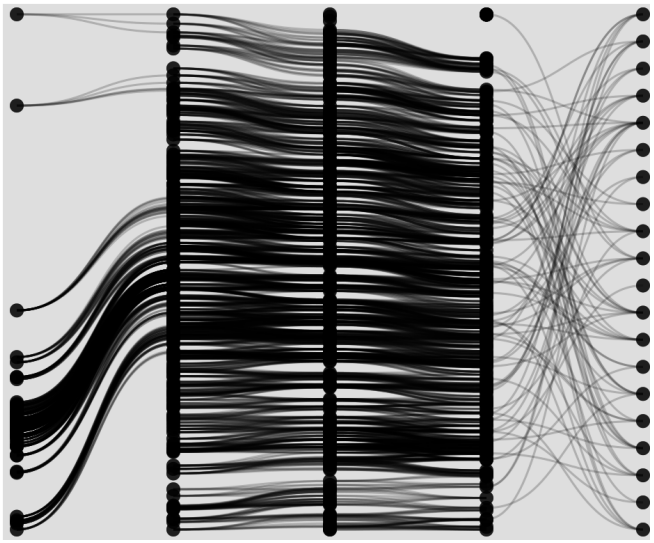

Fig. 9. The humanoid robot running control network. This network does not feature the same split as the walking robot control network, but the actions of the robot are more significantly varied – it does not hit a limit cycle in its brief dashing between locations. While we know training has happened (the links are not in random directions but focused), this particular visualization is not as useful as others.

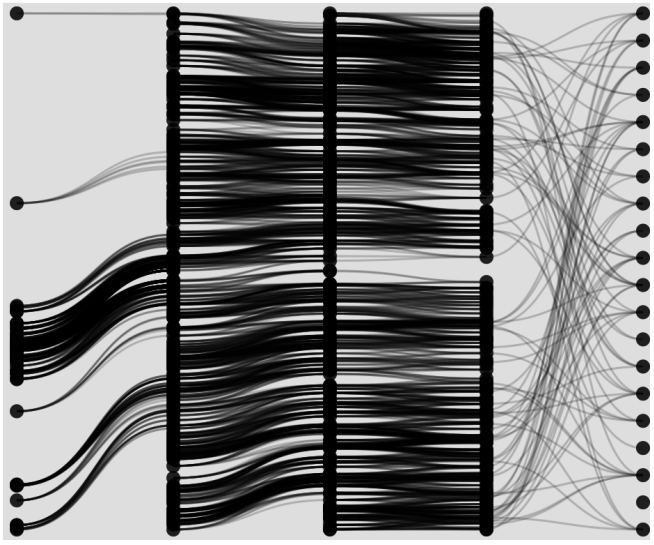

Fig. 8. The humanoid robot walking control network. An interesting feature of this network is the split at the third layer. As this is a bipedal robot that features a limit cycle walking gait (that was learned through trajectory optimization), we presume this split represents the symmetry encountered during walking, although more investigation would be required to confirm this.

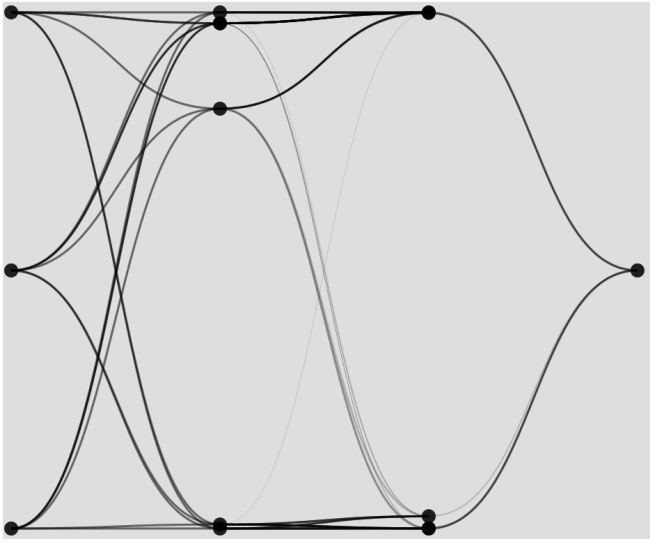

Fig. 10. We simplify the trained XOR network based on the visualized groupings and create a smaller network of 6 neurons and 5 neurons. The loss function for both networks trained on three-way XOR were the same (0.009) suggesting that the performance is comperable with less neurons. This network simplification is a potential application of this visualization technique.