Interactive Visualization of Neural Network Activities

Zijian Li
University of Washington

Callin Switzer
University of Washington

Yue Zhao
University of Washington

Zhengde Zhao
University of Washington

ABSTRACT

This project builds an interactive visualization system for neural networks, with application to the flight modeling of a virtual moth. The system visualizes the weight connections, as well as the activation values of the nodes real-time in response to adjustments in the input values. Weights distribution in the network is shown, with the brushing scheme to select absolute values in the desired range. The graph is plotted using the force directed layout scheme to accommodate to networks with different sizes. Visualization results using different neural network structures are shown with various interaction. Extraction of useful information using the system is discussed with examples.

KEYWORDS

Neural networks, Data Visualization, Interactive

ACM Reference Format:
Zijian Li, Callin Switzer, Yue Zhao, and Zhengde Zhao. 2019. Interactive Visualization of Neural Network Activities. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Background

Insects like moths have very simple brains, yet they are capable of precise and subtle flying maneuvers, which involve complex fluid dynamics. They generate locomotor force by activating the flight muscles to move their wings, and the aerodynamic forces and torques it generates enable they to preform various flying behaviors including fast forward, odor plume tracking, hovering in front of flowers, decelerating upon approach and conpensating for enrionmental perturbations [**?**] [**?**] . To understand how moths control their flight is already by itself interesting, not to mention that bio-inspired flapping wings system has potential application to micro air vehicles according to Chen in [**?**].

Multiple recent researchers has targeted this question. In [**?**], the authors treated this question as an inverse problem, where the input of the system is the initial position, initial velocity, targeting position and targeting velocity while the output of the system is the wing motion. They developed an aerodynamic model illustrating how a particular motion in the wings of a moth could result in a kinetic motion. In [**?**], they found that not only the wings but also the shape of the body plays an important role in a moth's flight control. New variables such as the angle of the body of the moth were introduced into the model. Further study has also shown that structural deformation also affects flapping wing energetics, see [**?**]. Nevertheless, to solve the inverse problem of flight control is intrinsically difficult. It involves solving a highly nonlinear dynamically system.

Project Goal

In the related study by Dr. Switzer, a researcher in the Biology Department of University of Washington, several deep neural networks are designed to solve the moth flight control problem. Imagine a moth trying to fly, it knows its current position and state, and where it wants to go. To achieve the goal, it needs to figure out the force control over its body that takes it on the trajectory to the destination, ending up with some final velocity. The feedforward neural networks were designed going from "where I am" and "where I want to go" (inputs), to the corresponding controls and final derivatives (outputs). The neural networks are trained with artificial dataset from Monte Carlo simulation, and are pruned to regularize the weights.

The goal of this project is to visualize the architectures of the pruned neural networks, which is capable of eliminating weights close to zero. Moreover, this project is unique in that it also interactively show the values of the nodes, in response to real-time adjustment of the input values.

Related Work

Neural networks are a special type of graphs, which are highly organized in their structures. For a feedforward neural network used in this study, nodes in the graph are divided

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

into layers, and the group of nodes in each layer only connect with nodes in adjacent layers. [**?**] described the visualization of feedforward neural networks, which aligns the position of the nodes according to their layer number. Figure 1 shows the structure of a fully-connected feedforward neural network used in the virtual moth modeling, in a typical way of conceptually visualizing a neural network.

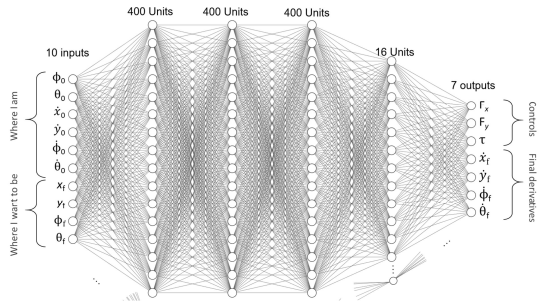

Figure 1: Feedforward Neural Network
Example of a fully-connected feedforward neural network used in the modeling of a virtual moth.

The above graph is clear in understanding the model structure conceptually, but has some drawbacks. First of all, this visualization does not show information such as activation values on the nodes and the weights of the connections. Secondly, it is bad under scaling. For example, there are 400 units in the middle layers, but the plot is incapable of showing the complete weight structure. Lastly, the layout is only suitable for simple structures like feedforward network. With the rapid development of new network structures nowadays such as the recurrent neural network [**?**], we need a more flexible framework to visualize neural networks.

2 METHODS

Computational Graph Construction

A fully-connected feedforward neural network has connections between each two adjacent layers in the form of a bipartite graph, which can be represented by a weight matrix W, where W_{ij} goes from the i-th node in the previous layer to the j-th node in the next layer. The weight matrices are also the typical data structure of storing a neural network. From the weight data, we can construct a directed graph

$$G = (V, E)$$

consisting of the set of nodes V and the set of edges E. The elements in the graph have the following properties:

$$v \in V: \text{ node ID } i, \text{ layer number } l, \text{ activation value } u,$$
$$e \in E: \text{ source } i, \text{ target } j, \text{ weight } w.$$

Note that the activation value u on each nodes is determined not only by the neural network itself, but also depends on the input values. Therefore, the network construction should be real-time in response to interactive input selection. To solve this problem, we construct the graph together with all the element properties. For the feedforward neural networks in our study, we implement the forward pass algorithm starting from the input layer values $u^{(0)}$., and iterates throughout the whole network to the output layer:

$$u_j^{(l+1)} = \tanh\left(\sum_i W_{i,j}^{(l+1)} u_i^{(l)} + b^{(l+1)_j}\right).$$

Since many neural networks are highly pruned in our study, a large portion of weights are zeros or close to zero. It is undesirable to show these edges, and thus we throw away them in the construction phase to reduce the amount of computation in the following visualization stage. Nodes that are not connected to any edge are also thrown away after edge trimming to keep the graph connected.

Force Directed Layout

In stead of aligning the nodes according to the layers into strict lines, we use the force directed layout scheme [**?**]. Each node v_i is regarded as a charged particle with charge q_i. Each two nodes v_i and v_j repel each other with the force

$$F = \frac{q_i q_j}{d_{ij}^2}.$$

Each edge e_{ij} is considered as a spring, applying the attracting force to the nodes it connects:

$$F = k(L - d_{ij}).$$

Besides, each node in motion is subject to air resistance

$$F = -bv_i.$$

At each time step, the forces acting on each node are calculated, which are then integrated to update its velocity and position. In our system, we implement the force directed layout scheme with d3-force, which uses a velocity Verlet numerical integrator for simulating physical forces on particles.

To show the layered structure of the neural network, we apply customized external force field. Similar to plotting nodes in different layers onto different X-positions, we apply attractive X-forces towards different X-position for nodes in different layers. A constant attractive force in the Y direction is also applied to ensure the vertical symmetry of the

network. The whole graph is also attracted to the center of the plot. The charges and forced are chosen according to the total number of nodes in the graph:

$$F_X \sim \frac{1}{|V|^2}, \quad q \sim \frac{1}{|V|}.$$

Visual Encoding

Once the graph of the neural network is constructed, we can encode the properties of the nodes and edges. The layer numbers of the nodes are encoded with the customized force directed layout. To represent the activation values of the nodes and highlight the nodes with the largest absolute values, we use the sizes of the nodes to encode the absolute values. The sign of the activation values are encoded with two different colors, representing positive and negative. Since the activation function in the neural network always scales the values of the nodes to $[-1, 1]$, we can control the size of the nodes in a certain range. For input and output layers where the activation function is not applied, additional scaling scheme (e.g. activation function) can be applied to adjust the size.

To highlight the most important weights and not to be distracted by other edges, we use both thickness and opacity to over-encode the absolute values of the weights. Since the network is fully-connected between layers with upto 800000 edges, such over-encoding can avoid small weights to cover the space and highlight the dominant weights. To make the system adjustable to different networks, the encoding is linearly scaled according to the maximal weight.

Interaction

The key feature of the interactive neural network visualization system is the automatic changing of node sizes in response to adjusting input values. For the 10 input values of the virtual moth model, we use slide bars for the user to adjust these values. The activation values of all the nodes in the network are computed in real-time and visualized on the plot. The values in the input layer are also visualized in the X-positions on the slide bar, with values shown in text.

Another important interaction is the brushing scheme to select weights with absolute values in a desirable range. For each edge, we plot its absolute weight on the brushing bar. The sign of the weight is encoded with the color and the Y-position on the brushing bar, forming two lines of dots in two distinct colors. From the brushing bar, the user can see the distribution of the weights in the neural network from the dot density. The "absolute value + double lines" design can exclude the middle part of weights around zeros in a simple way.

In addition to the above two major interactions, the visualization system also allows for other interactions. We use tooltips for detailed information demand, for example the layer number l and activation value u of a node, and the connection information and weight value of an edge. The drag-and-move interaction provided by the force directed layout is also useful. As all the nodes in a single layer (except the input and output layer) are essentially equivalent to each other, the positions of them are arbitrary. User can drag nodes to avoid overlapping of the main edges. Besides, there is a drop-down menu for users to choose the neural network models to visualize. For the virtual moth study, we provide 6 feedforward networks with different structures.

3 RESULTS

Visualization Interface

Figure 2 shows the interface of the web-based interactive neural network visualization system. Users can use the drop-down menu to load the neural network model to visualize. The force directed scheme then configures the layout of the graph plot. By adjusting the slide bars on the left, users can see the activation value change of all the nodes instantly, encoded by the size of the nodes. Users can use the brushing bar at the bottom to choose the absolute value range. The brushing bar also shows the weight distribution, as well as how many weights are selected in the target range.

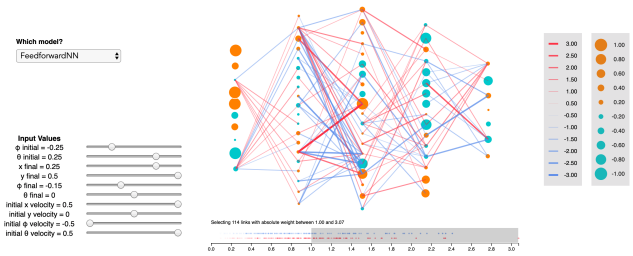

Figure 2: Visualization Interface
The web-based interface of the interactive neural network visualization system.

As a base example of our visualization, we use the above pruned neural network with 3 hidden layers of sizes: (20, 20, 16). Input values are chosen using the slide bars to mimic a jumping up of the virtual moth: $x_f = 0.25, y_f = 0.5, \dot{x}_0 = 0.5, \dot{y}_0 = 0, \phi_0 = -0.25, \theta_0 = 0.25, \phi_f = -0.15, \theta_f = 0, \dot{\phi}_0 = -0.5, \dot{\theta}_0 = 0.5$. The brushing bar selects only weights with absolute values over 1. We can see that the network is sparse under weight filtering, and there are several dominant edges with large weight values. For the nodes, the visualization shows that only a subset of nodes have significant activation values, and many of them are zeros or close to zero. The weight distribution is shown in the brushing bar, indicating the network is pruned to exclude weights close to zero.

Scaling to Large Networks

The visualization system is capable to accommodate large neural networks as well. Figure 3 shows the visualization of a neural network with hidden layer sizes: (512, 512, 512, 512). Comparing to the previous example, the network is larger in both depth and width, with over 2000 nodes and nearly 800000 edges. We can see that the force directed layout fits this huge network in the plot space nicely. The layers are represented as the disk-shaped clusters. Computational wise, the largest neural network in our study only takes seconds to load and configure the layout, which is tolerable.

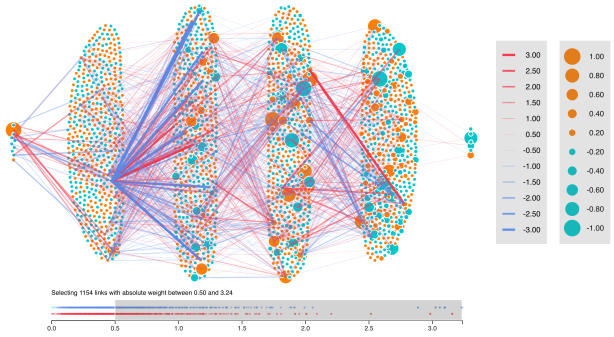

Figure 3: Large Neural Networks
The visualization system is capable of accommodating to large neural networks automatically.

4 DISCUSSION

The interactive neural network visualization system provides useful information to users in both training and analysis stages. By using the brushing bar to exclude weights close to zeros, the user can get an idea of how the network can be trimmed. For example, in Figure 2, we can see that the network is sparse under weight filtering, and there are several dominant edges with large weight values. The weight distribution shows that the absolute values of the weight distribution is dense around 1, indicating that the network has been pruned to exclude weights close to zero. For the large network example in Figure 3, we can see there are areas where the most of the largest weights are going out, indicating the hot spots in the neural network.

The interaction of adjusting input values is also useful in understanding the neural network, and is helpful for a explainable model. In Figure 4, we show two scenarios with the same inputs as in Figure 2. The only difference is that in the bottom plot y_f is adjusted from 0.5 to −0.5. We can see that a different group of nodes are in active now, and some nodes have a sign change. Researchers can use this feature to identify the active area in the neural network while doing certain movement, and which nodes are interconnected.

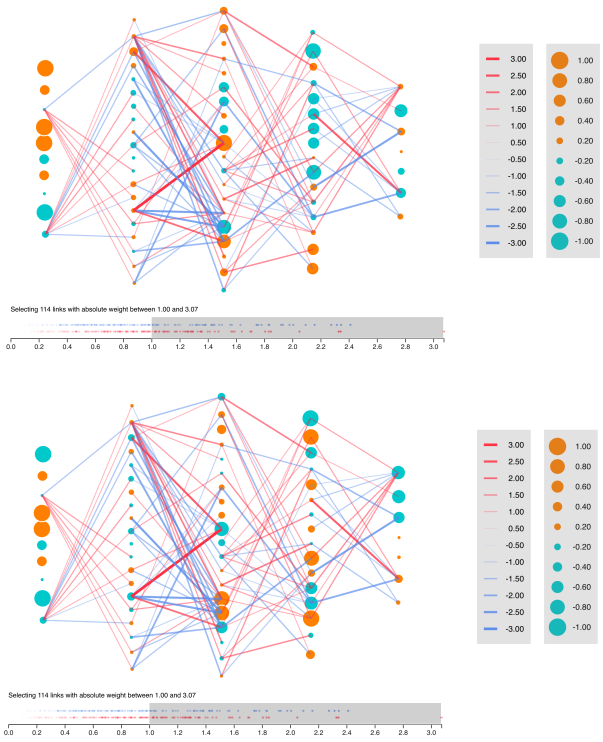

Figure 4: Input Adjustment
Comparing the node values under adjustment of a single input value.

5 FUTURE WORK

Besides the current features, there can be additional improvements in future work. The in addition to changing the node values in real-time, we can also change the network structure at the same time. Nodes that are of zeros activation values are of no use in propagating information, thus the edges going out of it are actually useless, regardless of their large weights. Therefore, we may define the activity level of each edge by the activation level of the outgoing node. The visualization of the activity level can be encoded such that inactive edges are invisible (e.g. use thickness or opacity).

Additionally, we may show the strongest path that is connected to each node upon tooltip. Also, it is interesting to visualize the configuration of the virtual moth in real-time (imaging a moth that moves its body), which helps the audience to understand the meaning of the inputs and outputs better. Lastly, we can visualize the distribution of the node values in real-time, as an estimation of "neuron activity level".