Interactive Network Visualization and Anomaly Detection Based on Deep Learning

Hayden Chen
hchen222@ucsc.edu

CSE261 Project

Deep neural networks have been used extensively to automatically detect network anomalies, assisting network administrators to deal with network failure and attack rapidly. LSTM is one of the most accurate DNN models for network anomaly detection [9]. However, human intervention is needed after an anomaly emerges, so a interactive interface that incorporates network traffic data and anomaly records can support human to investigate the network according to the information give by the IDS. This paper propose an interactive system that visualize the network data along with the detected anomalies, providing an overview of the network and also drill-down views of network traffic information for human to locate, target and resolve problems more easily.

1 Motivation

As people's lives rely increasingly on the internet, it is essential for network administrators to provide reliable services and secure personal data protection. Services that reach to thousands of millions of users suffer from the delicacy inherited from its own complexity and they are also easily targeted by attackers. One minute out-of-service can cause a large amount of financial loss. Therefore, Cyber-security becomes one of the most popular field in the industry.

The upkeep of a healthy network depends on exhaustive prevention of failures and rapid intervention after a failure happens. As a result, real-time detection of anomalies is an essential part for a Cyber-security system. However, even the best detection system can't resolve the problem. Human intervention is still a necessary part of the system. How we can clearly present data and anomalies to network administrators and how we can assist them in the process of resolving network problems is the motivation and topic for this paper.

2 Related Work

In the field of network security analysis, KDD'99 and NSL-KDD dataset are among the most commonly used ones. According to [12] KDD'99 has some inherit problems in which data redundancy, about 75% and 78% duplicated data in the training and test set respectively, is the most negative factors in the machine learning model training, which could lead to a biased model. NSL-KDD fixed some of the problems in KDD'99 but it doesn't yet completely simulate the real-life network. However, NSL-KDD is reliable for establishing comparative results between different models. 37% of the training data are attacks that are grouped into four categories: DOS, Probe, R2L and U2R while the other 63% are normal network traffics [10].

In their survey [5] of deep learning-based network detection system, Kwon D. and etc. introduced the common techniques that are used for network anomaly detection. These techniques cover the overall

process of network data mining, from the data reduction pipeline (dimensionality reduction, clustering and data sampling) to the different supervised and unsupervised models including restricted Boltzmann machine (RBM), deep Boltzmann machine (DBM), deep belief network (DBN), deep neural network (DNN) and recurrent neural network (RNN). Neural network models generally has better performance than other models due to the high level features serving as accurate representation of the original data [5]. RNN is an improved version of DNN that is able to memorize previous data when dealing with time or space sequential data so it is suitable for data mining tasks that analyze related sequence patterns. Network graph data is usually time related, especially when attacks are happening, the real-time network traffic evolve temporally indicating that a possibility high accurate classification model for network anomaly can be built with RNN. The survey lacks the actual performance comparison of the aforementioned models albeit provides a comprehensive introduction to all the aspects related to network anomaly detection.

As a matter of fact, deep neural network models are widely investigated in the field of network anomaly detection. However, they either lack a thorough performance evaluation using standard machine learning metrics on the NSL-KDD dataset [1, 2, 3, 4] or gives a sub-optimized result [13]. Naseer et al. gave a detailed analysis in standard machine learning metrics on several deep learning model trained with the NSL-KDD dataset. Naseer et al. mainly focused on how Auto-encoder, CNN and LSTM (Long short-term memory) perform with the NSL-KDD train set and NSL-KDD test set in metrics including Receiver Operating Characteristic (ROC), Area under Curve (AuC), Precision-Recall Curve, test set accuracy and mean Average Precision (mAP), all of which are derived from the confusion matrix. They came up with the result where LSTM gave the highest accuracy on both the NSL-KDD Test+ and the NSL-KDD Test21 dataset while CNN gave the highest AuC value of 0.955 and 0.916 on NSL-KDD Test+ and NSL-KDD Test21 dataset respectively. This indicates that even the best model to-day gives classification results with nearly 90% true positive rate at the cost of about 20% false positive rate, meaning in production environment, human interference is unavoidable for network anomaly detection and prevention. However, this paper focuses on smoothing the information transition experience between machine anomaly detection system and a human mind by introducing interactive interface that visualize knowledge from the automatic system for a human operator.

McGuffin demonstrate an algorithm to visualize network data in different layouts [8]. One of the result is suitable for visualizing network by clustering connected nodes and localize cross-over of edges using Force-Directed Layout algorithm. The open-source network visualization framework, Cytoscape Web [7] includes implementation of the aforementioned algorithm along with other layout algorithms.

3 Research Approach

3.1 Network Structure

We first visualize the network structure using the Cytoscape.js [7] library. There are three main algorithms we are focusing on: CoSE Bilkent, fCoSE and Cola layout. Cola layout is a force-directed physic simulation layout algorithm, whose disadvantage is its computational cost and non-deterministic result. The result of these algorithms are shown in Figure 1 and 2. Because we have a stable network structure and don't need to recalculate the network structure each time the users view the visualization, we choose fCoSE as our layout algorithm. fCoSE stands for fast CoSE. It is an improved version of the CoSE algorithm which can generate a layout in a shorter time by reducing the iteration. Cytoscape.js doesn't have native implementation of fCoSE, however, there exist an extension for fCoSE layout.

The Figure 1, 2, and **??** show networks that have hierarchy structure, which is how generally large

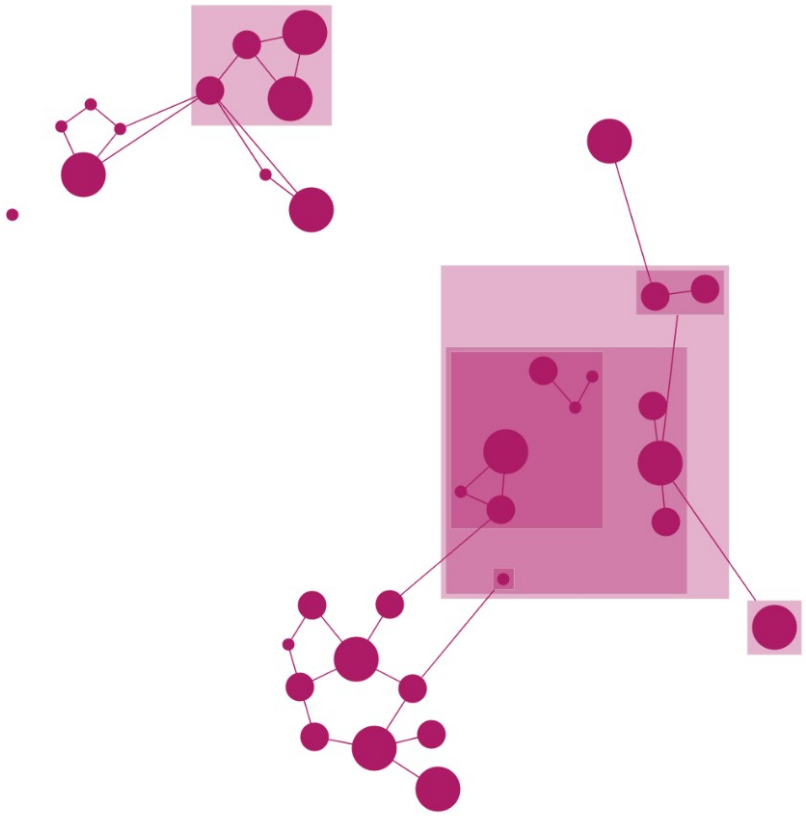

Figure 1: CoSE

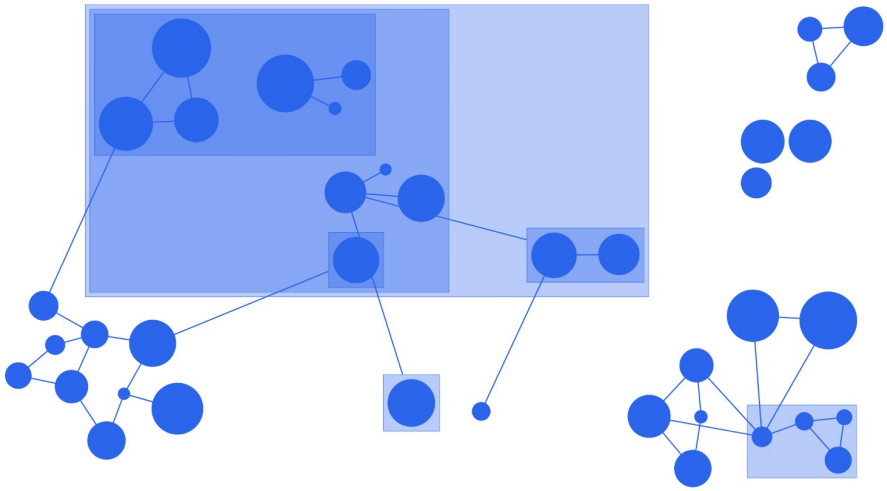

Figure 2: fCoSE

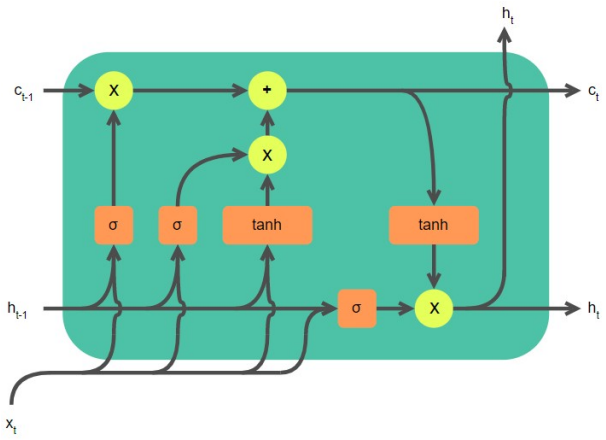

Figure 3: The data flow inside an LSTM cell

real-life network structure. However, the NSKDD dataset doesn't contain the topography information due to privacy concerns. So we use a pair-to-pair network dataset [6] [11] to simulate a simple network structure and assign traffic to the edges of the graph randomly.

3.2 Model - LSTM

Long Short Term Memory is a special kind of Recursive Neural Network, as it has feedback connection in addition to traditional feed-forward connection, which means it can process a sequence of data. It is widely used in network anomaly detection or intrusion detection systems. As shown in Figure 3, inside a LSTM unit, there are four gates that control the feedback information flow into the memory cell c. i is the input gate, which regulates the new values flowing into the cell, f is the forget gate, which controls how much a value remain in the cell and the output gate o controls the how the cell memory value should output as the hidden state. The LSTM unit intuitively imitate how memory works in the human brain and is very effective at recording the dependencies between the items in the input sequences. W_* are the weight of the memory gates It is the learnable parameters of the LSTM model that are optimized during the model training process. Finally, like all other RNN model, LSTM outputs the current cell memory as the hidden state h_t.

$$
\begin{aligned}
f_t &= \sigma_g(W_f x_t + U_f h_{t1} + b_f) \\
i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\
o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \\
\tilde{c}_t &= \sigma_c(W_c x_t + U_c h_{t1} + b_c) \\
c_t &= f_t \circ c_{t-1} + i_t \circ \tilde{c}_t \\
h_t &= o_t \circ \sigma_h(c_t)
\end{aligned}
$$

We can stack multiple LSTM layers and forming so called stacked LSTM: the output of the previous layer becomes the input of the next layer and the overall hidden state h_n will have a shape of $[l * d, m]$ where l is the number of layers, d is the number of directions and m is the hidden feature size. In our

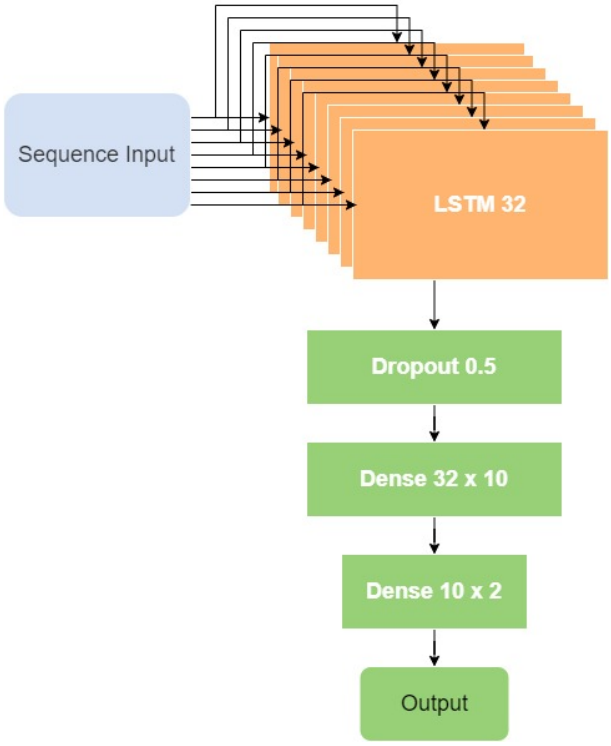

Figure 4: The model structure

model, we are using unidirectional LSTM because network traffic doesn't have a backward information feedback like sentences have.

After the LSTM layers are two fully connected dense layers which have 10 and 2 features output and ReLU and softmax activation function respectively. We also add a 0.5 dropout layer in between the dense layers and the LSTM to control the risk of overfitting. The model structure is illustrated in the Figure 4

3.3 Model - Training and evaluation

We train the model using the NSL-KDD training dataset to build a classification model based the LSTM model in 15 epochs, feeding all 41 features inside the NSL-KDD dataset. The training accuracy is shown in Figure 5. The training accuracy reaches to over 97.5% at epoch 2 and the validation accuracy fluctuate around 81% in the entire training process. One reason for that might be an over complicated model structure that results in overfitting of the data. However, the model gives high specificity and precision meaning it generally performs better at recognize normal traffic and the abnormal traffic it predicts is usually correct.

recall	specificity	accuracy	precision
0.756175	0.93461	0.833038	0.938582

Table 1: Evaluation of the LSTM model

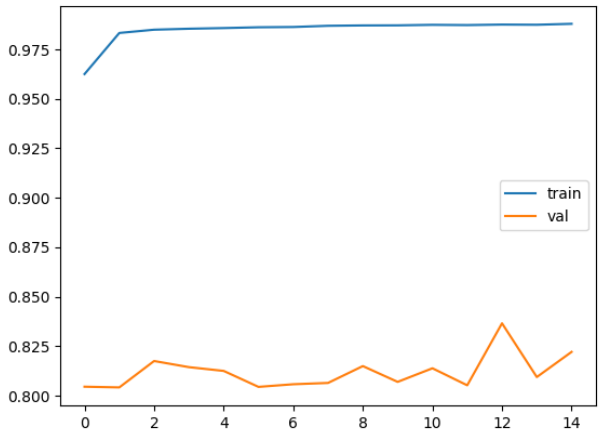

Figure 5: The accuracy trend for training dataset and validation dataset

3.4 Visualization

With the classification model available, we then implement a server-client environment which simulates a IDS running in real-time, feeding information to the visualization on the client side. We save the trained LSTM model into as file and instantiate it when the IDS server starts. When a client side starts, it will initiates a WebSocket connection with the server and once the connection is established, the server starts sending traffic data with prediction label from the classification model to the client side. Finally the client side draws visualization using the data collected from the WebSocket. We calculate a softmax value from the output of the model as the confidence factor of that classification, which is then mapped to a color interpolation factor in the visualization generating different color on the node edges to indicate the possibility of an attack detected.

We build the graph and edge visualization with Cytoscape.js, which provides a set of jQuery-like APIs that draws graph and binds event listeners to HTML canvas element. It is proven to be reliable and responsive. The library can draw network graph with tens of thousands of nodes and edges while remaining a responsively interactive. There are some limitation to the customization of the drawing elements. For example, it is difficult to add outline to the edges since the library defines a small set of styling rules that don't contain customizing outline of edges. However, the library is enough for the visualization of this project.

Users can interact with the visualization and drill into the edge to see the detail traffic information coming from the dataset features and the numeric confidence value give by the model. We also provide the functionality to pause the real-time inflow of traffic data that enables users to inspect the state of the network in a particular time frame. Lastly, we provide a modifiable buffer size for the traffic so that traffic in the past can stay in the visualization before disappearing.

Finally, to actually see the effectiveness of our LSTM model, we create a side-by-side visualization of the traffic with predicted label and one with the ground truth by utilizing the NSL-KDD test dataset. This comparison view is valuable for testing the prediction performance of a IDS classifier.

4 Result

As shown in Figures 6 7 8. We are able to visualize a small peer to peer network with traffic. The unidirectional arrow indicates the source and destination of the traffic and the color of the arrow represents the class of the traffic. For abnormal traffic, we use the confidence value, which is in the range of $[0, 1]$, to interpolate between yellow and red. Lastly, we show the comparison view comparing the LSTM model output and the ground truth.

5 Conclusion

In the project, we create a IDS using LSTM model and combine it with a visualization system. It provides a intuitive way for human to investigate the reports give by the IDS. By bridging the machine learning effectiveness and human intelligence, the system help increase the security of network infrastructures. There are future work that can be conducted on this system: 1. The NSL-KDD dataset doesn't contain the source and destination of network traffic, which could be used to trained a LSTM model that recognized the location of the traffic and the previous states of the source and destination, increasing the prediction accuracy. 2. The visualization can generate the network layout in the server beforehand because the network structure is usually stable and generating the layout in frontend leads to performance given sufficiently large amount of nodes and edges. 3. A differential view should be provided to better visualize the difference between the prediction and ground truth, which can also be extended to compare any two models.

References

[1] M. Z. Alom, V. Bontupalli & T. M. Taha: *Intrusion detection using deep belief networks*. In: *2015 National Aerospace and Electronics Conference (NAECON)*, pp. 339–344, doi:10.1109/NAECON.2015.7443094. ISSN: 2379-2027.

[2] R. C. Aygun & A. G. Yavuz: *Network Anomaly Detection with Stochastically Improved Autoencoder Based Models*. In: *2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud)*, pp. 193–198, doi:10.1109/CSCloud.2017.39.

[3] Loïc Bontemps, Van Loi Cao, James McDermott & Nhien-An Le-Khac: *Collective Anomaly Detection Based on Long Short-Term Memory Recurrent Neural Networks*. In Tran Khanh Dang, Roland Wagner, Josef Küng, Nam Thoai, Makoto Takizawa & Erich Neuhold, editors: *Future Data and Security Engineering*, Lecture Notes in Computer Science, Springer International Publishing, pp. 141–152, doi:10.1007/978-3-319-48057-2_9.

[4] Ahmad Javaid, Quamar Niyaz, Weiqing Sun & Mansoor Alam: *A Deep Learning Approach for Network Intrusion Detection System* "3"(9), pp. 21–26. Available at `https://eudl.eu/doi/10.4108/eai.3-12-2015.2262516`.

[5] Donghwoon Kwon, Hyunjoo Kim, Jinoh Kim, Sang C. Suh, Ikkyun Kim & Kuinam J. Kim: *A survey of deep learning-based network anomaly detection* 22(1), pp. 949–961. doi:10.1007/s10586-017-1117-8. Available at `https://doi.org/10.1007/s10586-017-1117-8`.

[6] Jure Leskovec, Jon Kleinberg & Christos Faloutsos: *Graph evolution: Densification and shrinking diameters* 1(1), pp. 2–es. doi:10.1145/1217299.1217301. Available at `https://doi.org/10.1145/1217299.1217301`.

[7] Christian T. Lopes, Max Franz, Farzana Kazi, Sylva L. Donaldson, Quaid Morris & Gary D. Bader: *Cytoscape Web: an interactive web-based network browser* 26(18), pp. 2347–2348.

doi:10.1093/bioinformatics/btq430. Available at `https://academic.oup.com/bioinformatics/article/26/18/2347/209295`. Publisher: Oxford Academic.

[8] M. J. McGuffin: *Simple algorithms for network visualization: A tutorial* 17(4), pp. 383–398. doi:10.1109/TST.2012.6297585. Conference Name: Tsinghua Science and Technology.

[9] Sheraz Naseer, Yasir Saleem, Shehzad Khalid, Muhammad Khawar Bashir, Jihun Han, Muhammad Munwar Iqbal & Kijun Han: *Enhanced Network Anomaly Detection Based on Deep Neural Networks* 6, pp. 48231–48246. doi:10.1109/ACCESS.2018.2863036. Conference Name: IEEE Access.

[10] S. Revathi & A. Malathi: *A detailed analysis on NSL-KDD dataset using various machine learning techniques for intrusion detection* 2(12), pp. 1848–1853. Publisher: Citeseer.

[11] Matei Ripeanu, Ian Foster & Adriana Iamnitchi: *Mapping the Gnutella Network: Properties of Large-Scale Peer-to-Peer Systems and Implications for System Design*. Available at `http://arxiv.org/abs/cs/0209028`.

[12] M. Tavallaee, E. Bagheri, W. Lu & A. A. Ghorbani: *A detailed analysis of the KDD CUP 99 data set*. In: *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, pp. 1–6, doi:10.1109/CISDA.2009.5356528. ISSN: 2329-6275.

[13] M. Yousefi-Azar, V. Varadharajan, L. Hamey & U. Tupakula: *Autoencoder-based feature learning for cyber security applications*. In: *2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 3854–3861, doi:10.1109/IJCNN.2017.7966342. ISSN: 2161-4407.

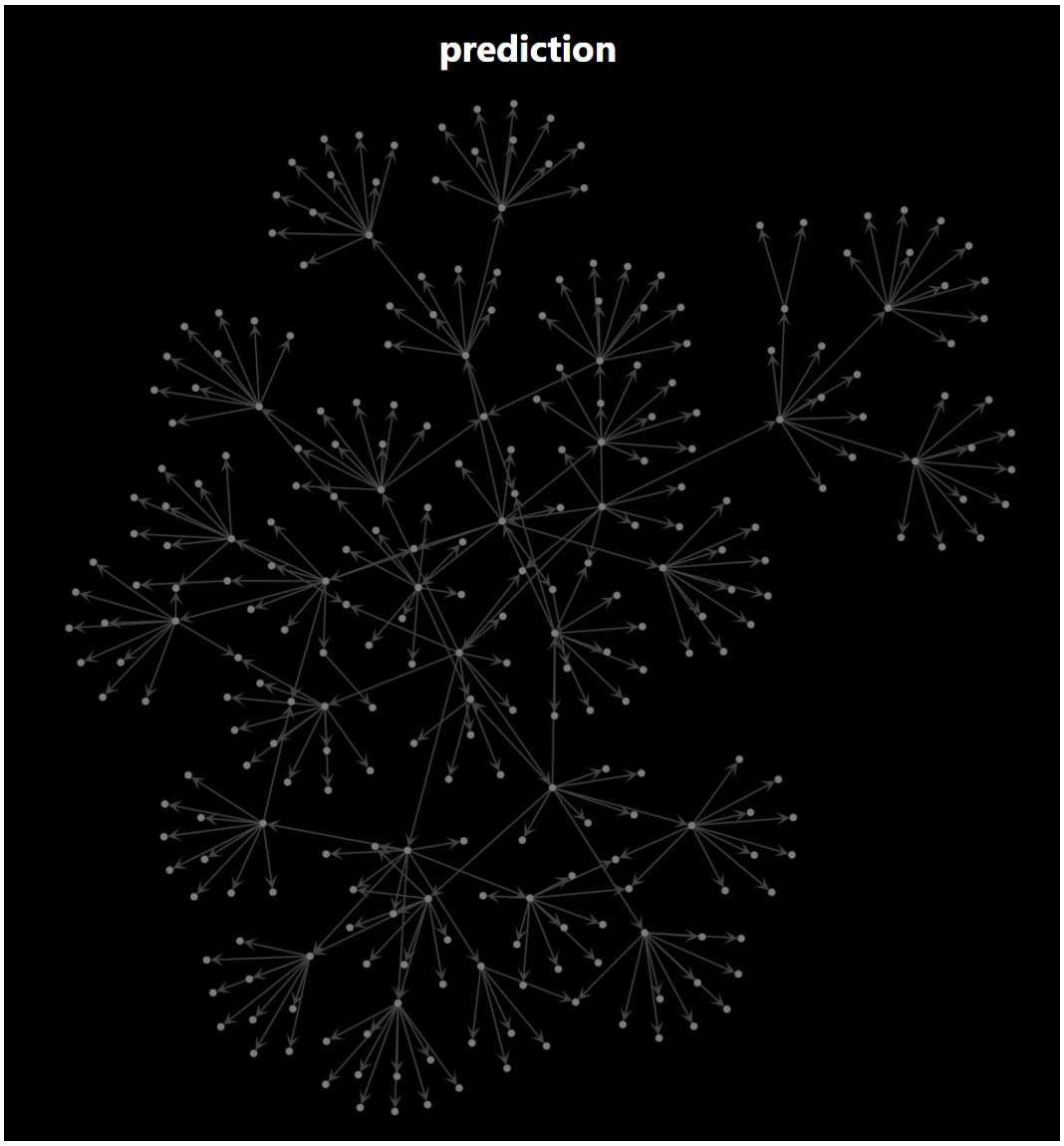

Figure 6: Network structure visualization with 300 connections

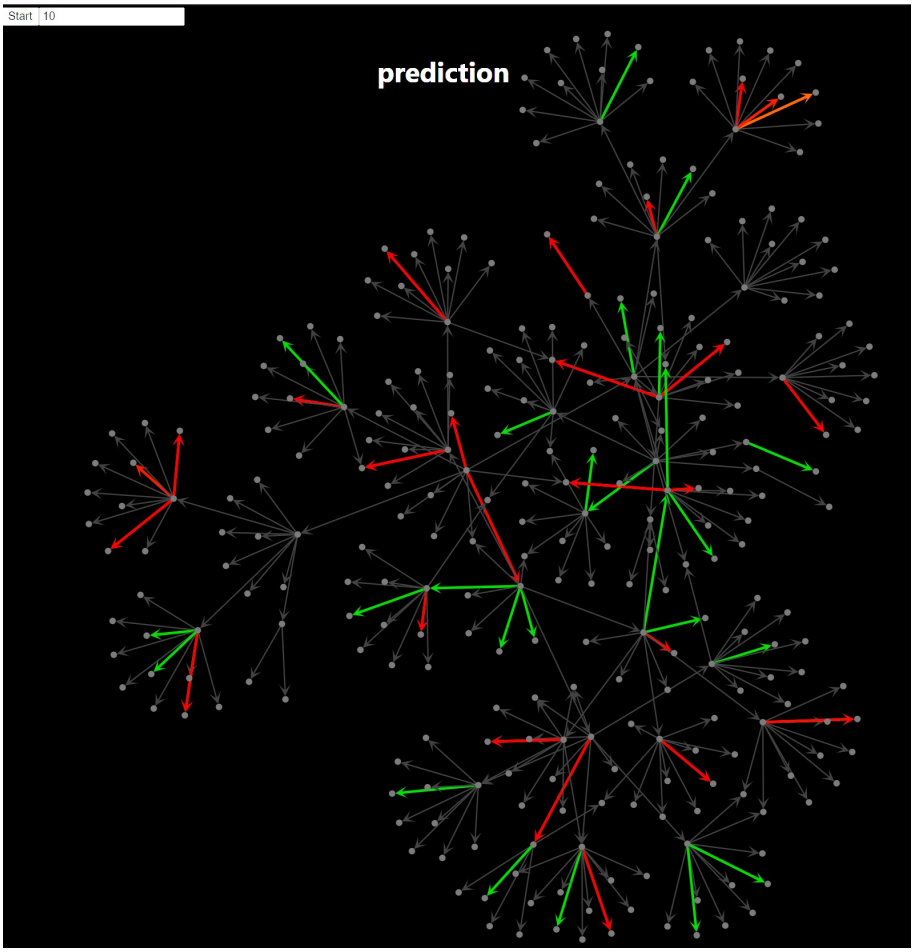

Figure 7: Traffic visualization: red edges indicate abnormal traffic and green edges indicate normal traffic. An interpolation between yellow and red represent the confidence of such prediction where yellow means lowest confidence and red indicates the model is almost certain of its prediction

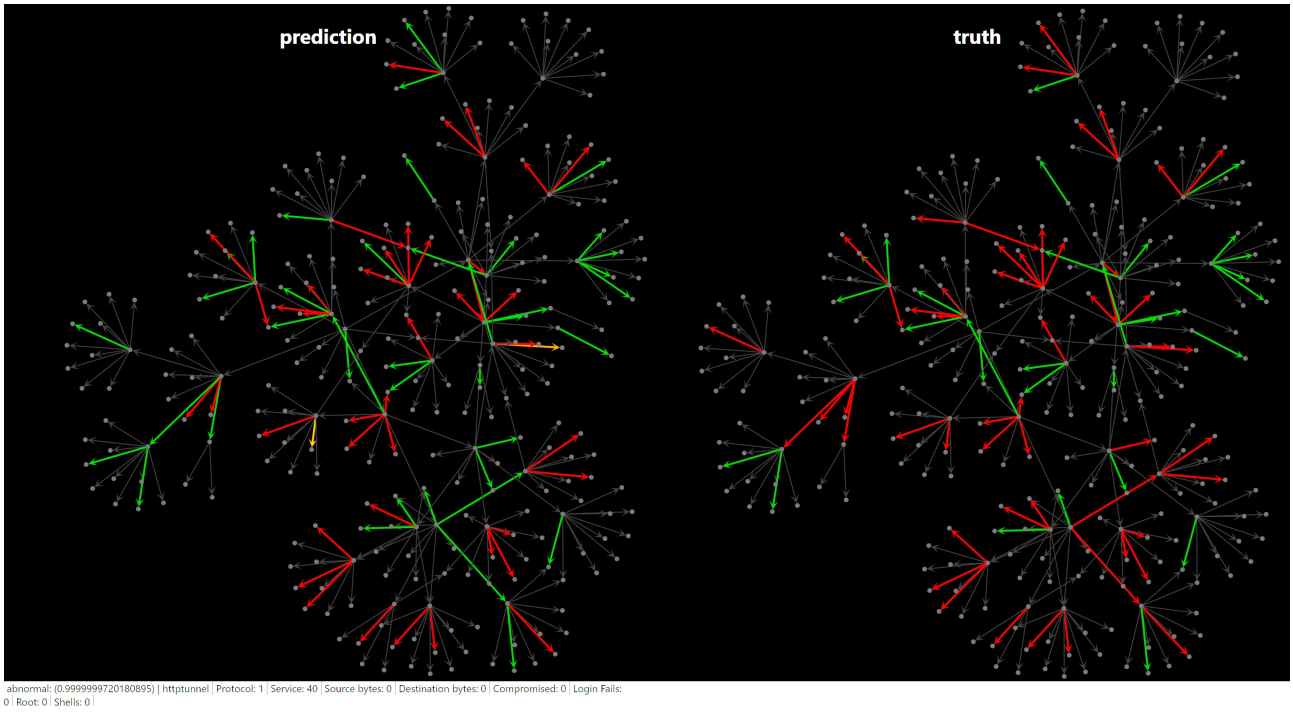

Figure 8: The side-by-side comparing view between the model prediction and the ground truth