CS230

Anomaly and Threat detection in network traffic using Deep Learning

Hiten Patel
hitenp@stanford.edu

Abstract

Network anomaly detection refers to the problem of detecting anomalies or attacks in the network traffic. With the ever growing network traffic, Network Anomaly and Threat Detection is a critical part in cybersecurity domain given new variety of attacks that arises frequently. In recent years, deep learning has been on the critical path of anomaly detection especially in the cybersecurity area. While traditional known rules or signatures based and machine learning algorithms have been used for anomaly detection, these methods are useful only for detecting point anomalies or novelty detection and are unable to detect or adapt to the changing patterns in the data. This project describes a deep learning model combining the distinct strengths of a Convolutional Neural Networks and Recurrent Neural Network; specifically a Bi-directional LSTM. The proposed model offers a high accuracy as well as high detection rate and comparatively lower False Positive Rate. Finally, this project compares against the state-of-art ML based models to evaluate the effectiveness of the proposed model as well as the deep learning techniques in the field of network anomaly detection.

1. Introduction

Traditional ML-based anomaly detection system mainly classifies and detects network traffic by analyzing the manually extracted features of network traffic. Such approaches still present a high false positive rate, which significantly limits the in-time detection efficiency, incurs large manual scrutiny workload, and cannot detect any unknown and new (0-day) attacks. On the other hand, Deep Learning based systems can not only analyze the manually extracted features but also automatically extract the features from the original traffic and have been proven to detect new features and attack patterns automatically to discover new attacks in this constantly evolving landscape.

In the proposed model, CNN is used to automatically extract and learn the spatial/high level features of a dataset and the Bi-LSTM layers to learn the long time-range temporal features of the data and combine these two networks into a hybrid model to predict attacks. The model is then evaluated on two network intrusion datasets, NSL- KDD and UNSW-NB15, and demonstrates that this offers a higher detection capability (better detection rate and validation accuracy) with lower false positive rate.

Below, I outline the datasets and the preprocessing steps required to train this model. I also describe the model architecture and related hyper parameters and optimization methods used for this model. Finally, I present the evaluation of this model against various metrics. The comparison with ML based approach is also performed to serve as the performance benchmark (baseline) for the effectiveness of the proposed CNN and LSTM based deep learning model.

2. Related work

Traditional machine learning techniques such as Support Vector Machine (SVM) , Random Forest [9] and Adaptive Boosting [10] have been often used by researchers for constructing Network Intrusion Detection classifiers. However, the downside of these approaches are they suffer from high false positive rate and lower detection rate. These approaches are not scalable to large dataset and also the validation accuracy rarely scales as the size of the data increases.

Deep learning techniques are increasingly getting popular to address these problems. Efficient Deep CNN-BiLSTM Model for Network Intrusion Detection [3] presents an effective approach which stacks these CNN and bi-directional LSTM layers to learn and detect attacks in the network dataset and this project derives motivation from this paper. DL-IDS: Extracting Features Using CNN-LSTM Hybrid Network for Intrusion Detection System [4] is the hybrid approach to build network intrusion detection system. A hierarchical convolutional and recurrent neural network, HAST-IDS [5] has been used to learn spatial and temporal features from the network traffic data. Lu-Net [6] is another deep learning approach where multiple CNN and RNN layer are stacked together starting with coarse-grained to fine-grained granularity in terms of number of CNN and RNN units in each layer. DANTE: Predicting Insider Threat using LSTM on system logs[7] is also an attempt for threat detection system predicting insider threats on the network data set.

3. Dataset and Features

Datasets in this domain are typically sourced from network traffic from various network devices or auditing events from applications. For this project, following publicly available datasets are used to evaluate the model:

- UNSW-NB15 : The raw network packets of the UNSW-NB 15 dataset is a more contemporary dataset that was created in the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS) for generating a hybrid of real modern normal activities and synthetic attack behaviors.

- NSL-KDD is a refined version of the original predecessor KDD99 dataset generated by Canadian Institute for Cyber Security (CICS) and consists both the benign and attack vectors. It has been one of the most popular dataset for Network Intrusion Detection Systems analysis.

Both datasets have time step column, essentially having time series data which makes the use of RNN a good choice to capture temporal behavior and patterns.

3.1 Threat Model

The UNSW-NB15 dataset contains attacks categorized into nine groups : DoS, Exploits, Generic, Reconnaissance, Worms, Shellcode, Analysis, Backdoor and Fuzzers. Thus, for multi-category attack prediction, in this dataset ten classes have been used including Normal category for benign events. For this dataset, the attack samples were collected from the three real-world websites: CVE (Common Vulnerabilities and Exposures), BID (Symantec Corporation), and MSD (Microsoft Security Bulletin)

The NSL-KDD dataset contains attacks categorized into four groups : Denial of Service (DoS), Probe (Probing Attacks), R2L (Root to Local Attacks) and U2R (User to Root Attack). Thus, for multi-category attack prediction, in this dataset five classes have been used including Normal category.

3.2 Data Preprocessing

Preprocessing of the datasets is performed by normalization of numeric features and one-hot encoding of categorical features.

1. One hot encoding : Categorical columns such as “protocol”, “state” and “service” are one hot encoded to improve model training.

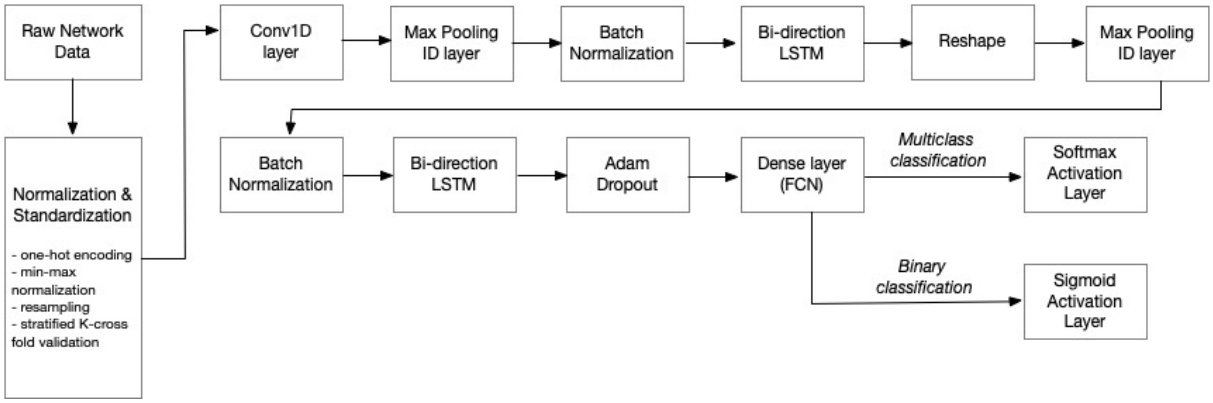
2. Normalization : Min-Max normalization is used as a feature scaling technique to rescale the input data in the range of (0,1). The motivation to apply this is for better convergence of gradient descent and improved regularization effect [12]

3. Resampling - UNSW-NB15 dataset has low number of records for categories like Worms, Fuzzers etc. To solve this issue, data in the training set is upsampled using imblearn.oversampling python library with 'minority' parameter to make sure every attack category has a comparable number of records.

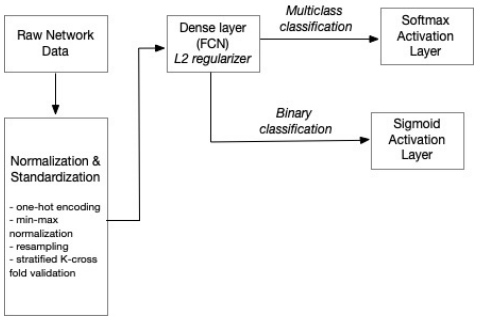
4. Stratified K-cross fold validation : Stratified K-cross fold validation technique is used to ensure the folds used for cross validation preserve the percentage of samples for each class. [13]

4. Methods

4.1 Proposed Multi-Class and Binary Classification Model

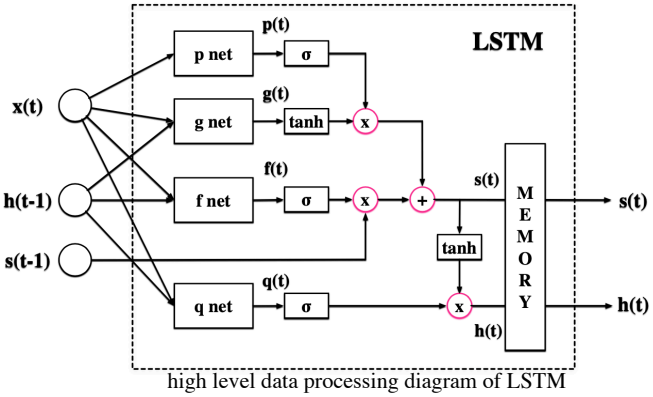


4.2 Benchmark / Baseline Model



The proposed model is implemented using TensorFlow backend, Keras and Scikit learn packages and were trained using GPUs on AWS as well as Google Colab notebooks.

The proposed model combines 1-Dimensional Convolutional Neural Network (1-D CNN) and multiple layers of Bi-directional LSTM (Bi-LSTM) network. Since the network packet is presented in a 1D format, we use 1D convolution. The main idea behind leveraging 1-D CNN layer is to allow the network to automatically extract features of raw data and also aiding in better extraction of coarse grained features at the start of the network. CNN and max pooling layers are used together for parameter sharing and spatial invariance



high level data processing diagram of LSTM

characteristics. Parameter sharing allows to constrain the number of parameters in the initial layers resulting in feature extraction using fewer amount of computation resources while spatial invariance enables to better recognize correlation between features. Rectified Linear Unit (ReLU) is used as the activation function for this layer for faster convergence.

Batch Normalization is applied in between the intermediate layers to reduce the effects of covariance shifts from layer to layer during training to prevent slower learning speed and to allow making the weights fluctuations more robust in the network. Reshape Layers are then applied to reshape the output from the previous layer to feed into the subsequent layers.

The feature map generated by CNN often manifests the spatial relations of data. CNN does not work very well for the data of long-range dependency. RNN, on the other hand, has an ability to extract the temporal features from the input data. The two Bi-LSTM layers (64 and 128 units) in the model are arranged in a manner which doubles it's kernel size in every iteration.

The reason for this choice is to mimic the use of coarse grain to fine grained learning to better understand the correlation of features learnt by the first 1-D CNN Layer and then long range time dependent features by the subsequent LSTM layers. Between each Bi-LSTM layer, there is a Max Pooling layer to dismiss the least relevant features and the Batch Normalization layers to normalize the output data of the previous intermediate layer in order to boost performance and decrease training times. Since the output granularity changes from one LSTM layer to another, a reshape layer is added to reshape the input data for the subsequent LSTM block. The final layer in the model is the fully connected layer serving as the output layer with *softmax* activation function for multi-class classification and *sigmoid* activation function for binary classification.

Adam dropout with the rate value of 0.6 is used as a regularization technique to better generalize during model inference. The Loss function used is *categorical_crossentropy* as the labels are one hot encoded [11] and *binary_crossentropy* for multi class and binary classification respectively. Batch size is set to 32.

5. Experiments/Results/Discussion

5.1 Experiments : I experimented with stacking more CNN and LSTM layers in the network and also increasing the number of units in CNN & LSTM layers but that did not significantly improve the accuracy and detection rates but resulted in increased training times and significantly more number of parameters in the network. (See Appendix for more details)

5.2 Model Evaluation

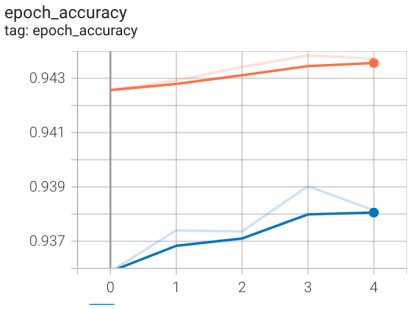
The model was trained on AWS (p2.xlarge and p3.2xlarge machines) as well as Google CoLab using GPUs and the evaluation was performed using TensorBoard and Scikit metrics python library.

The performance of the model is based on two scenarios: (1) **binary classification** to predict whether the data represents either as an attack or a normal traffic; (2) **multi-class classification** to predict whether the data represents either as normal or as one type of attacks given in the dataset attack model (namely, 10 classes for UNSW-NB15 and 5 classes for NSL-KDD).

Repeated k-fold cross-validation is performed to improve the estimated performance of a machine learning model. This involves repeating the cross-validation procedure multiple times and reporting the mean result across all folds from all runs. This mean result is expected to be a more accurate estimate of the true unknown underlying mean performance of the model on the dataset, as calculated using the standard error. The primary metrics used to evaluate the performance of models for both the datasets are Accuracy, Detection Rate and False Positive Rate (FPR) . Accuracy measures the model's ability to predict all classes. Detection Rate (TP / TP + FN). measures the model's ability to predict attacks correctly. FPR (FP / FP+TN) is the percentage of normal records classified as attacks. F-1 score provides a single score that balances both the concerns of precision and recall in one number.

5.3 UNSW-NB15 Results:

The results of the binary and multi class classification on this dataset are evaluated for k (# of cross folds) ranging from k=2 to k=10.



Binary Classification:
The mean accuracy is 93.16% with the best accuracy of 93.81% and mean detection rate is 93.16%. The best results are for k=10. The mean F1-score is 93.23% and the best F1-score is 93.86% for k=10. The mean false positive rate (FPR) is 0.7% which is remarkable.

For comparison purposes, the benchmark tests using the logistic regression model with L2 regularization on the similar dataset were obtained to test the effectiveness of the proposed DL model . The accuracy rate on logistic regression using Keras is 81% and using Scikit is 80%.

Multi-Class Classification:
The mean accuracy is 83.18% and mean detection rate is 82.04%. The best results are for k=10. The mean F1-score is 78.9% and the best F1-score is 80.26% for k=10. The mean false positive rate (FPR) is 2% which is noteworthy to evaluate the effectiveness of the model and the best FPR is 0.019 for k =10.

For comparison purposes, the benchmark tests using the logistic regression model with L1L2 as well as just L2 regularization on the similar dataset were obtained to test the effectiveness of the proposed DL model . The accuracy rate on logistic regression using Keras with L2 regularization is 68% and Scikit was 67%. L1L2 regularization did not perform well on this dataset relative to just using L2 regularization.

5.4 NSL-KDD Results:

The results of multi class classification on this dataset are evaluated for k (# of cross folds) ranging from k=2 to k=6. Binary classification was not evaluated on this dataset due to time constraints as the multi class results looked very promising anyway.

Multi-Class Classification:
The mean accuracy is 98.9% and mean detection rate is 98.97%. The best results are for k=6. The mean F1-score is 99.29%. The mean false positive rate (FPR) is 0.25% which is again noteworthy to evaluate the effectiveness of the model. For comparison purposes, the benchmark tests using the logistic regression model with L2 regularization on the similar dataset were obtained to test the effectiveness of the proposed DL model . The accuracy rate on logistic regression using Keras is 82.34% and detection rate is 78.48%.

6. Conclusion/Future Work

This project used the combination of CNN and Bi-Directional LSTM layers to automatically learn and extract the spatial and temporal features trained on multiple network datasets to help detect evolving threat and attack vectors. The benchmark tests on both the datasets shows that the hybrid CNN and LSTM based deep learning model significantly outperforms ML based approaches in terms of performance and also offers much better detection rates. Network Threat Detection is an evolving space, although, empirically stacking more layers or increasing the number of units in LSTM layers did not further improve the accuracy / detection rates, it is imperative to evaluate the model on more datasets to improve the threat model based on the evolving novel threat /attack patterns and behaviors. Furthermore, it would also be important to infer the causal relationship between features and detections to make the detections more human understandable as these detections would typically be triaged by the threat response analysts manually before raising security incident / alarms.

7. References

[1] The UNSW-NB15 data set - https://research.unsw.edu.au/projects/unsw-nb15-data-set

[2] NSL-KDD dataset - *https://www.unb.ca/cic/datasets/nsl.html*

[3] Jay Sinha and M. Manollas "Efficient Deep CNN-BiLSTM Model for Network Intrusion Detection" AIPR 2020: Proceedings of the 2020 3rd International Conference on Artificial Intelligence and Pattern Recognition

[4] Pengfei Sun, Pengju Liu, Qi Li, Chenxi Liu, Xiangling Lu, Ruochen Hao, and Jinpeng Chen, "DL-IDS: Extracting Features Using CNN-LSTM Hybrid Network for Intrusion Detection System", April 2020

[5] W. Wang, Y. Sheng, J. Wang, X. Zeng, X. Ye, Y. Huang, and M. Zhu, "Hast-ids: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection," *IEEE Access*, vol. 6, pp. 1792–1806, 2017.

[6] Peilun Wu and Hui Guo, "LuNet: A Deep Neural Network for Network Intrusion Detection" 10.1109/SSCI44817.2019.9003126, Dec 2019

[7] Qicheng Ma and Nidhi Rastogi, "DANTE: Predicting Insider Threat using LSTM on system logs"

[8] Donghwoon Kwon, Kathiravan Natarajan, Sang C. Suh, Hyunjoo Kim, Jinoh Kim "An Empirical Study on Network Anomaly Detection Using Convolutional Neural Networks"

[9] J.Zhang,M.Zulkernine,andA.Haque,"Random-forests-based network intrusion detection systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 38, no. 5, pp. 649– 659, 2008.

[10] W. Hu, J. Gao, Y. Wang, O. Wu, and S. Maybank, "Online adaboost- based parameterized methods for dynamic distributed network intrusion detection," *IEEE Transactions on Cybernetics*, vol. 44, no. 1, pp. 66–82, 2013.

[11] https://keras.io/api/losses/probabilistic_losses/#categoricalcrossentropy-class

[12] https://en.wikipedia.org/wiki/Feature_scaling

[13] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html

8. Appendix

8.1 Results Summary Table for UNSW-NB15 dataset

Binary Classification Results				
	Conv + LSTM	Logistic Regression (Keras)	Logistic Regression (Scikit)	Random Forest
Mean Accuracy	93.16%	78%	79.9%	86.13%
Mean Detection Rate	93.16%	86%	87%	87.8%
Mean F1 Score	0.93	0.791	0.795	0.85
False Positive Rate	0.7%			
MultiClass Classification Results				
	Conv + LSTM	Logistic Regression (Keras)	Logistic Regression (Scikit)	Random Forest
Mean Accuracy	83.18%	66.5%	66.7%	77.45%
Mean Detection Rate	82.04%	64%	66%	77.4%
Mean F1 Score	0.80	0.68	0.71	0.75
False Positive Rate	2%			

8.2 Results Summary Table for NSL-KDD dataset

MultiClass Classification Results				
	Conv + LSTM	Logistic Regression (Keras)	Logistic Regression (Scikit)	Random Forest
Mean Accuracy	98.9%	82.34%	84.45%	85.35%
Mean Detection	98.97%	78.48%	84.4%	85.3%
Mean F1 Score	0.97	0.80	0.83	0.83
False Positive Rate	0.25%			

8.3 Experiments

1) I experimented with the model (shown on right) which consists of multiple blocks of 1D-Convolutional and LSTM layers with batch norm and drop outs applied at each layer. The final layers consisted of 1D-Convolutional layer, global average pooling and a fully connected layer as referenced in [6]. As promising as it looks, however it did not perform well in terms of accuracy and detection rate. Specifically, the accuracy and detection rates were ~10% lower compared to the current implemented model.

```
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv1D(64, kernel_size=64, padding="same", activation="relu", input_shape=(196, 1)),
    tf.keras.layers.MaxPool1D(pool_size=2),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.LSTM(64, return_sequences=False),
    tf.keras.layers.Reshape((64, 1), input_shape=(64,)),
    tf.keras.layers.Dropout(0.5),

    tf.keras.layers.Conv1D(128, kernel_size=128, padding="same", activation="relu"),
    tf.keras.layers.MaxPool1D(pool_size=2),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.LSTM(128, return_sequences=False),
    tf.keras.layers.Reshape((128, 1), input_shape=(128,)),
    tf.keras.layers.Dropout(0.5),

    tf.keras.layers.Conv1D(256, kernel_size=256, padding="same", activation="relu"),
    tf.keras.layers.MaxPool1D(pool_size=2),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.LSTM(256, return_sequences=False),
    tf.keras.layers.Reshape((256, 1), input_shape=(256,)),
    tf.keras.layers.Dropout(0.5),

    tf.keras.layers.Conv1D(512, kernel_size=512, padding="same", activation="relu"),
    tf.keras.layers.GlobalAveragePooling1D(),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(10, activation='softmax')
])
```

2) I also experimented with doubling up the LSTM units in the current implemented model i.e 128 and 256 units Bi-Directional LSTMs instead of existing 64 and 128 Bi-Directional LSTMs. This did not improve the accuracy or the detection rate but remained approximately similar compared to the current implemented model, however resulted in significant increase in the number of parameters in the network as well as increased training time for the model.