Few-Shot Learning for Intrusion Detection

Aawez Mansuri
Department of Computer Science
California State University, Northridge
Northridge, United States

Wonjun Lee
Department of Computer Science
California State University, Northridge
Northridge, United States

***Abstract*—This paper explores the application of few-shot learning, specifically Prototypical Networks, to intrusion detection systems (IDS) as a solution to challenges faced by traditional machine learning models with imbalanced and evolving datasets. The study focuses on a binary classification task using NSL-KDD dataset involving 'Normal' and 'U2R' classes, and, despite the use of less than 0.080% of normal data and only 52 U2R samples, the proposed model achieves a high overall accuracy of 95.6%, a recall rate of 96% for the normal class, and 81% for the U2R class. This performance surpasses comparative models utilizing traditional Neural Networks, Recurrent Neural Networks, and K-Nearest Neighbors, indicating the superiority of the prototypical network approach in this context.**

Keywords: **Prototypical Networks, Few-Shot Learning, Embeddings, Network Intrusion Detection, NSL-KDD**

I. INTRODUCTION

With the rise of the digital era, the landscape of cybersecurity threats continues to evolve at an alarming pace, making Intrusion Detection Systems (IDS) an essential component in identifying and neutralizing these threats. Traditional IDS, as well as machine learning-based IDS, however, grapple with several challenges such as high false-positive and false-negative rates, and a scarcity of training data. In this paper, we explore the application of Prototypical Networks, an innovative model that leverages Few-Shot Learning (FSL) to discern patterns from a limited number of examples, for intrusion detection.

With the rise of the digital era, the landscape of cybersecurity threats continues to evolve at an alarming pace, making Intrusion Detection Systems (IDS) an essential component in identifying and neutralizing these threats. Traditional IDS, as well as machine learning-based IDS, however, grapple with several challenges such as high false-positive and false-negative rates, and a scarcity of training data. In this paper, we explore the application of Prototypical Networks, an innovative model that leverages Few-Shot Learning (FSL) to discern patterns from a limited number of examples, for intrusion detection.

Historically, IDS have relied heavily on predefined rules and hard-coded signatures to identify known threats. These rules and signatures, however, are static and do not adapt to the rapidly changing threat landscape, making this approach increasingly deprecated. Machine learning (ML) models, while more adaptable, suffer from their own set of challenges. The performance of these models heavily depends on the availability of large volumes of labeled training data, which can be challenging to acquire. Moreover, these models risk overfitting to the training data, which limits their capacity to generalize to new, unseen threats. In spite of their learning capabilities, traditional ML algorithms can still yield high false positives, especially when faced with novel or sophisticated attacks.

Historically, IDS have relied heavily on predefined rules and hard-coded signatures to identify known threats. These rules and signatures, however, are static and do not adapt to the rapidly changing threat landscape, making this approach increasingly deprecated. Machine learning (ML) models, while more adaptable, suffer from their own set of challenges. The performance of these models heavily depends on the availability of large volumes of labeled training data, which can be challenging to acquire. Moreover, these models risk overfitting to the training data, which limits their capacity to generalize to new, unseen threats. In spite of their learning capabilities, traditional ML algorithms can still yield high false positives, especially when faced with novel or sophisticated attacks.

II. RELEVANT WORK

Deep learning has been extensively employed for intrusion detection in recent years. Yin et al., for instance, applied a recurrent neural network to the NSL-KDD dataset for intrusion detection [1], while Wu et al. introduced a novel approach that utilized convolutional neural networks for the same purpose [2].

The concept of the prototypical network, which employs few-shot learning principles for classification tasks, was first proposed by Snell et al. [3]. The model was tested on image classification problems, demonstrating its ability to classify with just a few samples. In a similar vein, Yu et al. implemented few-shot learning for intrusion detection using the NSL-KDD and UNSW-NB15 datasets [4]. They compared the efficacy of traditional machine learning models to few-shot learning, shedding light on the potential of this approach. However, their methodology potentially limits our understanding of the model's full capabilities.

Prototypical networks operate using support and query sets for training and optimization, respectively. While the support set trains the model to swiftly adapt to classes, the query set evaluates the model's performance. Model parameters are optimized for increased accuracy based on its performance during evaluation. It's common to create both the support and query sets from the training data. However,

Yu et al. deviated from this norm by generating the support set from testing data, which could potentially result in compromised performance and unreliable results on unseen data.

Notably, prior research conducted training and testing on all the classes within the NSL-KDD dataset. Contrary to this approach, our work focuses on binary classification using only two classes at a time - normal and a selected malicious class - leveraging few-shot learning for this purpose.

III. METHODOLOGY

A. Dataset

Our research utilizes the NSL-KDD dataset for training and testing our model. This dataset, created by a research team at the University of New Brunswick in Canada, is an enhanced version of the KDD Cup 1999 Dataset, which was popularly used for intrusion detection. The NSL-KDD dataset rectifies limitations of the original dataset, such as redundancy, irrelevance, and use of outdated attack types. It is divided into two sections: KDDTrain+ (for training) and KDDTest+ (for testing), encompassing 41 features related to aspects like the number of bytes, packets, the service used, and the type of attack. The training dataset comprises 125,973 samples, while the testing dataset includes 22,544 samples, both categorized into normal and malicious classes. The malicious classes encompass attack types DoS, Probe, R2L, and U2R, with data distribution across these types being highly imbalanced (Figure 1). Since our study is centered on binary classification using prototypical networks, we focus exclusively on data pertaining to U2R attack types.

B. Feature Selection

Our research specifically targets the U2R class due to its minimal sample size. We utilize all features except num_outbound_cmds and difficulty_level. The former has a standard deviation of 0 and offers no value during model training, while the latter, indicating the difficulty level of samples, is irrelevant to our objective.

C. Data Pre-Processing

The pre-processing pipeline commences with data loading, followed by mapping the attack class with respective samples (buffer_overflow, loadmodule, perl, and rootkit are mapped as U2R). Owing to the dataset's imbalanced nature, we perform undersampling to balance the data. This reduces the normal samples to match the number of U2R instances, preventing overfitting and enabling us to test the model's effectiveness on unseen examples. The next step involves one-hot encoding of categorical data, followed by scaling to ensure all data is on a similar scale.

Despite the NSL-KDD dataset being clean and structured, we apply certain pre-processing techniques, such as undersampling, to balance out the imbalanced data distribution. We use the RandomUnderSampler from the imblearn library to perform this task, creating balanced data with equal samples from each class.

Despite the NSL-KDD dataset being clean and structured, we apply certain pre-processing techniques, such as undersampling, to balance out the imbalanced data distribution. We use the RandomUnderSampler from the imblearn library to perform this task, creating balanced data with equal samples from each class.

The final step involves standard scaling of the train and test datasets using the StandardScaler, which scales the features to have a mean of zero and a standard deviation of one, thereby handling outliers, preventing bias, and enhancing performance.

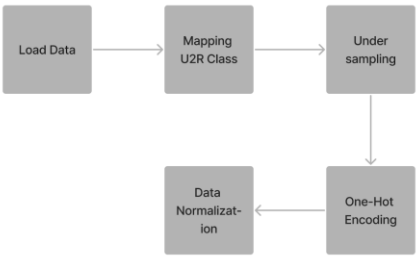

Figure 1. Data Pre-Processing Pipeline for the model.

D. Model Architecture

The model architecture of our few-shot learning system follows the design of a prototypical network, as illustrated in Figure 5. Our data is partitioned into a support set and a query set. The support set holds a small number of labeled examples from each class, while the query set contains the examples that need to be classified. We employ a simple feed-forward neural network to learn a meaningful embedding for the input data. The objective is to map input examples into a lower-dimensional space where similar samples have similar embeddings. The prototype for each class is then computed in the support set, with the prototype being the mean of the embeddings of all examples belonging to that class in the support set. For every sample in the query set, we calculate its embedding and the Euclidean distance between this embedding and each class prototype. Subsequently, we assign the query samples to the class with the nearest prototype. After each iteration, the loss is calculated, and the network's weights are updated via backpropagation, leading to revised prototype classes. Once the model has been trained, we evaluate its performance on the test data using the same strategy.

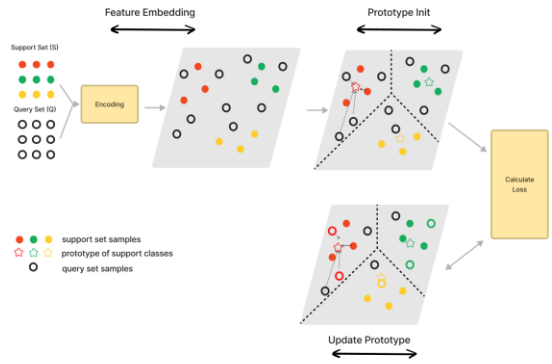

Figure 2. The prototypical network architecture of the model.

E. *Model Training*

The training of our model begins by setting the number of training epochs and the learning rate. The Adam optimizer is defined for updating the weights. We employ DataLoader to iterate through the training data in batches. For each iteration, we select the support set and query set from the batch. The neural network, as discussed in the previous section, is then used to calculate the embeddings of the support set. Subsequently, we compute class prototypes as the mean of these support set embeddings. The embeddings of the query set are calculated next, followed by the computation of the prototypical loss based on distances between these query set embeddings and class prototypes. The final step involves updating the model weights using both the optimizer and the computed loss. At the culmination of the training process, we are left with an accurate embedding representation that mirrors class similarity, with samples belonging to the same classes being much closer to each other.

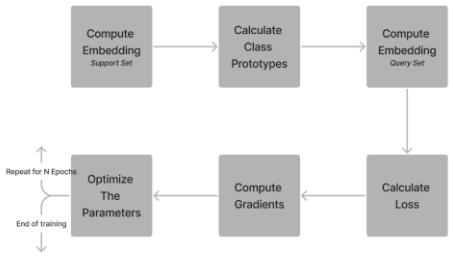

Figure 3. The training flow of the model.

F. *Model Testing*

To evaluate the performance of our model, we first switch the model to evaluation mode. We then compute embeddings for the complete training set and testing set utilizing the trained ProtoNet model. Subsequently, we determine the class prototypes based on the embeddings of the training data. Once this is complete, we ascertain the distances between the embeddings of the test data and the class prototypes. The test instances are then classified based on proximity to the nearest prototype.

G. *Model Evaluation*

In evaluating our machine learning model, we employ key metrics including accuracy, recall, precision, and the F1-score. Accuracy measures the overall correctness of the model, representing the proportion of true results in the total samples. Recall, also known as sensitivity or true positive rate, assesses the model's ability to correctly identify all relevant instances. Precision quantifies the model's capability to avoid false positive errors, essentially measuring the correctness of positive predictions. Lastly, the F1-score combines the model's precision and recall into a single score to balance both metrics. These measurements provide a comprehensive perspective on our model's performance, enabling a more precise interpretation of its predictive capabilities and limitations.

$$Accuracy = \frac{T_p + T_n}{T_p + T_n + F_p + F_n}$$

$$Precision = \frac{T_p}{T_p + F_p}$$

$$Recall = \frac{T_p}{T_p + T_n}$$

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

Figure 4. The evaluation metrics of the model.

IV. RESULTS

In our binary classification experiment using the NSL-KDD dataset, we focused specifically on Normal and U2R classes. The model, trained over 10 epochs with a learning rate of 0.001, was evaluated on a test set comprising 9711 Normal and 37 U2R samples. For each training batch of 40 samples, 10 from each class were designated as the support set, with the remainder forming the query set. The performance evaluation metrics included accuracy, precision, recall, and F1-score. Our model achieved an overall accuracy of 95.69%. As depicted in Figure 11's classification report, Label 0 represents the Normal class, and Label 1 denotes the U2R class. Of these, 9,295 normal samples and 416 U2R samples were correctly classified, resulting in high recall rates for both classes. However, 416 samples were incorrectly identified as U2R. Our model achieved a recall of 0.96 for Normal samples and 0.89 for U2R samples, indicating that it correctly classified 96% of Normal samples and 89% of U2R samples.

Our comparative analysis of the prototypical network involved training the NSL-KDD dataset using a Neural Network (NN), Recurrent Neural Network (RNN), and K-Nearest Neighbours (K-NN). As depicted in Figure 5, the NN model achieved an overall accuracy of 83.7%. Despite this, its recall rate was significantly lower compared to the

prototypical network, and only 30 out of 37 U2R samples were correctly classified.

On the other hand, the RNN model demonstrated an accuracy of 66.84%, correctly classifying all 37 U2R samples. However, it managed to correctly classify only 67% of normal samples, suggesting a model bias.

Lastly, we employed K-NN due to its efficient performance in classification tasks, particularly binary classification. The K-NN model achieved an accuracy of 80.54% and an average recall rate for both normal and U2R instances. Despite these metrics, the results were not optimal when compared to the prototypical networks.

In summary, the prototypical network outperformed the other models in terms of accuracy and recall rate for both classes, demonstrating its superiority in handling the NSL-KDD dataset for binary classification tasks.

Models	Accuracy	0 - Recall	1 - Recall	TP	TN	FP	FN
NN	83.70%	84%	81%	8129	30	7	1582
RNN	66.84%	67%	100%	6479	37	0	3232
KNN	80.54%	81%	76%	7823	28	9	1888
Prototypical Networks	95.69%	96%	89%	9295	33	4	416

Figure 5. Model Results.

V. CONCLUSION

In conclusion, our model outperformed NN, RNN, and KNN in both accuracy and recall rate, affirming the efficiency of Prototypical Networks in handling intrusion detection systems tasks. Our model, with its impressive 96% recall rate for normal samples, dramatically decreases the potential for falsely identifying normal network instances as malicious—an essential characteristic for real-world applications where normal network events far outnumber malicious ones.

Despite RNN's commendable recall rate for U2R samples, its lower recall rate for normal samples remains problematic, as it implies a 33% likelihood of incorrectly flagging normal network events as malicious. Our Prototypical Network model, on the other hand, displayed exceptional performance even with limited training data. It achieved the highest recall rate for normal samples using just 0.077% of the normal training data, underscoring its suitability for scenarios with data constraints, such as Intrusion Detection Systems.

While the superior accuracy, better handling of limited data, low false-positive rate, and practicality of our model for data-scarce tasks are evident, the real-world implementation of Prototypical Networks still faces hurdles due to the novelty of the algorithm and limited experimentation.

Future work should aim to explore the applicability of few-shot learning beyond the realm of Computer Vision, examining its potential in areas such as Natural Language Processing, Cognitive Computing, and other classification tasks. The quest to refine and broaden the implementation of Prototypical Networks continues, promising exciting advancements in the field of machine learning.

REFERENCES

[1] C. Yin, Y. Zhu, J. Fei and X. He, "A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks," in IEEE Access, vol. 5, pp. 21954-21961, 2017, doi: 10.1109/ACCESS.2017.2762418.

[2] K. Wu, Z. Chen and W. Li, "A Novel Intrusion Detection Model for a Massive Network Using Convolutional Neural Networks," in IEEE Access, vol. 6, pp. 50850-50859, 2018, doi: 10.1109/ACCESS.2018.2868993.

[3] J. Snell, K. Swersky, and R. S. Zemel, "Prototypical Networks for Few-shot Learning," in Advances in Neural Information Processing Systems 30, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 4077-4087.

[4] Y. Yu and N. Bian, "An Intrusion Detection Method Using Few-Shot Learning," in IEEE Access, vol. 8, pp. 49730-49740, 2020, doi: 10.1109/ACCESS.2020.2980136.

[5] Mahfouz, Ahmed M., Deepak Venugopal and Sajjan G. Shiva. “Comparative Analysis of ML Classifiers for Network Intrusion Detection.” International Congress on Information and Communication Technology (2019).

[6] M. Tavallaee, E. Bagheri, W. Lu and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, Ottawa, ON, Canada, 2009, pp. 1-6, doi: 10.1109/CISDA.2009.5356528.

[7] Wang, Yaqing & Yao, Quanming & Kwok, James & Ni, Lionel. (2020). Generalizing from a Few Examples: A Survey on Few-shot Learning. ACM Computing Surveys. 53. 1-34. 10.1145/3386252.

[8] J. Zhao, S. Shetty and J. W. Pan, "Feature-based transfer learning for network security," MILCOM 2017 - 2017 IEEE Military Communications Conference (MILCOM), Baltimore, MD, USA, 2017, pp. 17-22, doi: 10.1109/MILCOM.2017.8170749.

[9] N. Moustafa and J. Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," 2015 Military Communications and Information Systems Conference (MilCIS), Canberra, ACT, Australia, 2015, pp. 1-6, doi: 10.1109/MilCIS.2015.7348942.

[10] L. Liu, P. Wang, J. Lin and L. Liu, "Intrusion Detection of Imbalanced Network Traffic Based on Machine Learning and Deep Learning," in IEEE Access, vol. 9, pp. 7550-7563, 2021, doi: 10.1109/ACCESS.2020.3048198.

[11] Archit Parnami, and Minwoo Lee, "Learning from Few Examples: A Summary of Approaches to Few-Shot Learning,", 2022, doi: 10.48550/arXiv.2203.04291.