# Data-driven Stochastic Anomaly Detection on Smart-Grid communications using Mixture Poisson Distributions

Daniel L. Marino [1], Chathurika S. Wickramasinghe [1], Craig Rieger [2], Milos Manic [1]
[1] *Virginia Commonwealth University*, Richmond, VA, USA
[2] *Idaho National Laboratory*, Idaho Falls, Idaho, USA
marinodl@vcu.edu, brahmanacsw@vcu.edu, craig.rieger@inl.gov, misko@ieee.org

***Abstract*—Characterizing communications in smart-grid distributed control systems is fundamental for understanding the expected behavior and identify abnormal scenarios. In this paper, we present a stochastic data-driven approach to model the the communication network in smart-grid systems. Our approach uses Mixture Poisson distributions to model the packet communication between the network devices. The network is modeled using a directed graph, where each edge represents a Poisson distribution of the packets being transmitted. Parameters are learned using mini-batch Expectation Maximization in order to scale to large datasets. The advantages of the presented approach are 1) unsupervised data-driven discovery of representative communication patterns, 2) intuitive visualization of the expected behavior, 3) scalability to large datasets, 4) coherent and interpretable model. Tests were conducted in a simulated SCADA microgrid distributed control system environment.**

***Index Terms*—Anomaly Detection, Poisson processes, Cyber-physical systems.**

## I. Introduction

The smart-grid revolution fueled by the need for high-penetration clean energy sources and cost-effective electricity generation and distribution has produced a highly interconnected electrical network [1]. Information and communication technologies (ICTs) play a central role in this revolution as they support communication and control functions in cyber-physical systems. However, the inclusion of ICTs technologies has open numerous vector attacks which are increasingly targeting critical infrastructure [2]. Machine-Learning-based Anomaly Detection provides an important tool to detect such attacks and improve situational awareness [3].

Understanding communication patterns on Cyber-Physical systems is essential to perform anomaly detection and protect infrastructure that relies on ICTs. Machine-Learning black-box models provide an appealing approach for anomaly detection as they require little-to-no prior-knowledge. However these models are often hard to interpret [4], with complicated decision boundaries that can open the door to unexpected new vector attacks [5].

Instead of using a black-box model, we present an approach based on the use of Poisson mixture distribution to model the communication network. Following this approach, we work towards improving coherence and interpretability over black-box models. The Poisson model serves as a coherent and natural distribution to model the packet rate between different devices of the network.

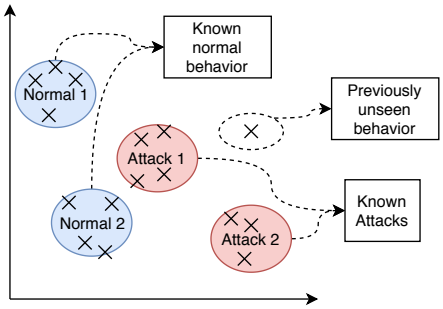


Fig. 1: Illustration of the communication modes learned by the Mixture-Poisson model.

We present a data-driven stochastic anomaly detection system called PM-ADS (Poisson Mixture Anomaly Detection System). PM-ADS models the communication network using Poisson distributions while using data to learn the model parameters. The presented approach measures the communication on a network using a packet sniffer and learns the network behavior profiles using the collected data. The learned behavior profiles are then used to identify abnormal communication that do not match the expected behavior.

Figure 2 offers a visual representation of the presented approach. The objective is to learn a multi-modal distribution that represents the different behaviors of the system. Each mode represents a particular behavior (state) of the system. After learning these communication modes, we can identify anomalous behavior (previously unseen) and known attacks by comparing the current state of the system w.r.t. the learned communication modes.

For learning the parameters, we present a mini-batch Expectation Maximization algorithm that was specifically designed to scale well to large datasets. This is an important requirement for real world systems. Data has a large volume as several packets are transmitted each second. Learning is also performed completely unsupervised. Labeled data is only used to evaluate the performance of the algorithm but it is not used for training. We demonstrate that anomaly detection can

be performed using only normal data.

The presented approach is especially useful in the following situations: 1) Identify anomalous behaviors which may correspond to possible attacks; 2) Understand the expected modes of communication of the system; 3) Understand the behavior of different cyber-attacks.

The rest of the paper is organized as follows: Section II presents the related work; Section III presents the PM-ADS model for network profiling and anomaly detection; Section IV presents the experiments performed to validate the approach.

## II. Related work

Cyber-Physical Systems(CPSs) have become core components in modern critical infrastructures [6]. This dependency of critical infrastructures on CPSs has made them vulnerable to various attacks [6]. Further, attacks on a single component of CPS can lead to catastrophic cascading failures [7]. Therefore, it is important to build resilient CPSs with the capability to detect any abnormal behaviors which lead to system failures.

Anomaly detection is the process of identifying patterns in data that do not represent the expected behavior of a system [8]. Many researchers have successfully used machine learning to detect anomalies in CPSs [8] [6] [9]. In [9], an unsupervised learning algorithm based on Recurrent Neural Networks was successfully used for anomaly detection in a water treatment plant. In [8], an unsupervised learning algorithm has proposed to learn a signal temporal logic formula (STL) which describes the normal behavior of the system. In [10], researchers have evaluated various machine learning models to identify power system disturbances. They have presented an evaluation which includes classifiers from various categories such as Probabilistic classification (Nave Bayes), Rule induction (OneR, NNge, JRipper), Decision tree learning (Random Forests), Non-probabilistic binary classification (SVM), and Boosting (AdaBoost). They have found that even simple models such as OneR and Nave Bayes have high performance in detecting attacks. Further, they have proposed an ensemble approach which combines JRipper and Adaboost models to detect power system disturbances.

One Class Support Vector Machines (OCSVMs) are widely used for anomaly detection where the models are trained using normal behavior of the system, and any unseen behavior is identified as an anomaly or an attacks [7]. OCSVMs are extensions of Support Vector Machines (SVMs) [11]. They can learn a decision boundary of a single class [12]. Any behavior which is different from the learned behavior will be detected as an outlier. Therefore,

Random Forest models has been widely used network intrusion detection [13], traffic flow predictions [14] and smart grid remote sensing [15]. Random Forest is a supervised machine learning algorithms which can be used to solve both regression and classification problems [14]. The basic idea of the Random Forest is to build multiple tree models together, resulting in an ensemble of decision trees [16]. Generally, these models show increased overall accuracy and stability compared to the models which use only a single decision tree model. Other advantages of these models include less sensitivity to outliers, less sensitivity to over-fitting, and ability to handle high dimensional data [14].

## III. Network Profiling and Anomaly Detection using Mixture-Poisson distributions

An overview of the presented PM-ADS is summarized in Figure 2. The PM-ADS measures the behavior of a CPS communication network by analyzing the packet data measured from the CPS network. The following are the main components of the PM-ADS:

- A packet sniffer measures the traffic on the network and obtains the data for training, testing and analysis.
- A time-fixed queue stores the packets from the last $\Delta t$ seconds in order to be analyzed.
- Graph analysis processes the packet data in the queue. The output is an adjacency matrix that represents the number of transmitted packets between devices for the last $\Delta t$ seconds.
- A Poison-Mixture model of the communication network which is used to characterize behaviors (profile the communication) and identify anomalies.

The following sections describe in detail: A) the graph representation of the communication network, B) the Poisson mixture model, C) the mini-batch Expectation Maximization algorithm used to learn the parameters of the Poisson mixture model, D) the anomaly detection approach.

### A. Network Graph

We model the communication network using a directed graph. Each node corresponds to a device in the network. The edges represent the communication between the devices. The communication between devices is characterized using a set of features. In this way, the network can be characterized at a given point in time using the corresponding adjacency matrix of the graph. Each entry of the adjacency matrix $x[u, v]$ corresponds to the value of feature between device $u$ and $v$(see Figure 3).

The adjacency matrices are derived from data obtained from packet capture data. We use the adjacency matrix to characterize the behavior of the system at a particular instant in time. Each node is represented using the IP address of the device. The edges between nodes represent the packet rate between devices.

### B. Poisson Mixture Model

Having defined a way to numerically characterize the network, we would like to learn from data a model that summarizes the behavior of the network. In this paper, given that the features have an integer domain, we choose Poisson distributions to model the system. The adjacency matrix $X$ is modeled using a mixture model:

$$p\left(\boldsymbol{X} \mid \theta\right) = \sum_{k=1}^{K} w_k p\left(\boldsymbol{X} \mid \lambda^{(k)}\right) \tag{1}$$

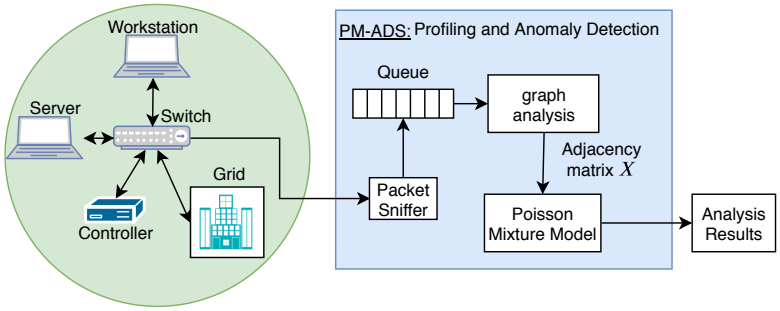


Fig. 2: PM-ADS: Behavior profiling using Mixture-Poisson Distributions

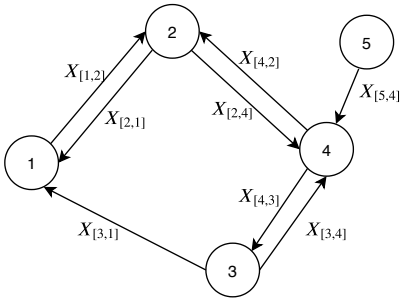


Fig. 3: Graph diagram. Nodes represent devices in the network with a unique IP address. Edges represent the communication between devices.

the mixture is composed of $K$ components, where each component models a particular communication behavior (mode). The matrix $X$ represents the features that characterize the communication (e.g. packet rate). $\theta = \left\{w_k, \boldsymbol{\lambda}^{(k)}\right\}_{k=1}^{K}$ are the parameters of our model learned from data. Each edge $(u, v)$ is modeled as a Poisson distribution:

$$p\left(\boldsymbol{X} \mid \boldsymbol{\lambda}^{(k)}\right) = \prod_{u=1}^{|V|}\prod_{v=1}^{|V|} \text{Poi}\left(X_{[u,v]} \mid \lambda_{[u,v]}^{(k)}\right) \tag{2}$$

where $\text{Poi}(x|\lambda)$ is a Poisson distribution with density function:

$$\text{Poi}\left(x = n \mid \lambda\right) = \frac{\lambda^n \exp\left(-\lambda\right)}{n!}$$

### C. Learning parameters

Given a dataset of samples $\mathcal{D} = \left\{\boldsymbol{X}^{(i)}\right\}_{i=1}^{|\mathcal{D}|}$, the parameters $\theta$ of the Poisson model described in 1 can be learned by maximizing the log-likelihood:

$$\theta^* = \arg\max_{\theta} \sum_i \ln p\left(\boldsymbol{X}^{(i)} \mid \theta\right) \tag{3}$$

In order to scale to very large datasets, we want to perform the maximization in 3 using mini-batch stochastic gradient ascend (Algorithm 1). This approach requires the computation of the gradients of the log-likelihood $\ln p\left(\boldsymbol{X}^{(i)} \mid \theta\right)$. However, directly computing the negative likelihood in 3 may lead to poor numerical performance when computing the gradients, specially for networks with a large number of nodes. The reason for the poor numerical stability is the product of probabilities in Eq. 2.

**Algorithm 1** Mini-batch Stochastic Gradient Ascend

**Input:** Number of iterations *max-iter*, Dataset $\mathcal{D}$, learning-rate $\eta$

**Output:** Learned parameters $\theta$

1: **for** $t = 1$ to *max-iter* **do**
2: $\quad \boldsymbol{X} \leftarrow$ Extract mini-batch$(\mathcal{D})$
3: $\quad$ **step-update:**

$$\theta_{[t+1]} \leftarrow \theta_{[t]} + \frac{\eta}{|\boldsymbol{X}|}\frac{\partial}{\partial\theta}\sum_i \ln p\left(\boldsymbol{X}^{(i)} \mid \theta\right)$$

4: **end for**
5: **return** Learned parameters $\theta_{[max\text{-}iter]}$

A more convenient representation is obtained by looking at the gradients of the log-likelihood w.r.t. the model parameters $\theta$:

$$\begin{aligned}\frac{\partial}{\partial\theta^{(k)}}\left[\ln p\left(\boldsymbol{X} \mid \theta\right)\right] =& \frac{1}{p\left(\boldsymbol{X} \mid \theta\right)}\frac{\partial}{\partial\theta^{(k)}}\left[w_k p\left(\boldsymbol{X} \mid \boldsymbol{\lambda}^{(k)}\right)\right] \\ =& p\left(k \mid \boldsymbol{X}, \theta\right)\frac{\partial}{\partial\theta^{(k)}}\left[\ln p\left(\boldsymbol{X}, k \mid \theta\right)\right]\end{aligned}$$

where:

$$\begin{aligned}p\left(\boldsymbol{X}, k \mid \theta\right) =& w_k p\left(\boldsymbol{X} \mid \boldsymbol{\lambda}^{(k)}\right) \\ p\left(k \mid \boldsymbol{X}, \theta\right) =& \frac{p\left(\boldsymbol{X}, k \mid \theta\right)}{p\left(\boldsymbol{X} \mid \theta\right)} = \frac{w_k p\left(\boldsymbol{X} \mid \boldsymbol{\lambda}^{(k)}\right)}{\sum_k w_k p\left(\boldsymbol{X} \mid \boldsymbol{\lambda}^{(k)}\right)}\end{aligned}$$

We can observe that the gradient is proportional to $p\left(k \mid \boldsymbol{X}, \theta\right)$. The term $p\left(k \mid \boldsymbol{X}, \theta\right)$ is the conditional probability of having $\boldsymbol{X}$ sampled from the mixture component $k$. This term serves as a weighting value that determines the amount that sample $X$ contributes to the update of the parameters for

**Algorithm 2** Mini-batch EM

**Input:** Number of iterations *max-iter*, Dataset $\mathcal{D}$, learning-rate $\eta$

**Output:** Learned parameters $\theta$

1: **for** $j = 1$ to *max-iter* **do**

2: $\boldsymbol{X} \leftarrow$ Extract mini-batch($\mathcal{D}$)

3: **Expectation:**

$$\pi^{(k,i)} = \text{Softmax}_k \left( \ln p \left( \boldsymbol{X}^{(i)}, k \mid \theta \right) \right)$$

4: **Maximization:** execute a mini-batch step-update

$$\theta_{[t+1]} \leftarrow \theta_{[t]} + \frac{\eta}{|\boldsymbol{X}|} \sum_{i,k}^{|\boldsymbol{X}|,K} \pi^{(k,i)} \frac{\partial}{\partial \theta} \ln p \left( \boldsymbol{X}^{(i)}, k \mid \theta \right)$$

5: **end for**

6: **return** Learned parameters $\theta_{[max\text{-}iter]}$

component $k$. We can express the update equations of gradient ascend as follows:

$$\theta_{[t+1]} = \theta_{[t]} + \frac{\eta}{|\boldsymbol{X}|} \sum_{i,k}^{|\boldsymbol{X}|,K} p \left( k \mid \boldsymbol{X}, \theta_{[t]} \right) \frac{\partial}{\partial \theta} \ln p \left( \boldsymbol{X}^{(i)}, k \mid \theta \right) \tag{4}$$

The update rule in 4 leads to the stochastic mini-batch Expectation Maximization (EM) algorithm described in Algorithm 2. The presented Mini-batch EM is a modification of the standard EM algorithm. In the Mini-batch EM, we replace the expensive maximization step of standard EM with a mini-batch single-step stochastic parameter update (step-4 in Algorithm 2). This allows us to scale the learning of parameters to large datasets.

For simplicity and readability, Algorithm 2 uses vanilla stochastic gradient descent to describe the maximization update-step. However, in the implementation of the algorithm, we used instead the Adam optimizer [17] to perform the maximization. The Adam optimizer provides faster convergence rates than vanilla gradient decent by using adaptive gradient moments in the step-update.

Algorithm 1 and 2 are equivalent, however, Algorithm 2 has the advantage that allows us to work directly with the log-likelihood of individual components $\ln p \left( \boldsymbol{X}^{(i)}, k \mid \theta \right)$, which allows us to improve numerical stability and simplify the implementation.

### *D. Anomaly detection*

Having characterized the network communication using the Mixture-Poisson distribution, we can use the cumulative density function (c.d.f.) of the model to detect anomalies. We define an anomaly as a point that is not covered by any of the mixture components. To quantify if a sample is covered by a component $k$, we check if the c.d.f. value is inside $(\alpha, 1-\alpha)$, where $\alpha \in (0, 1)$:

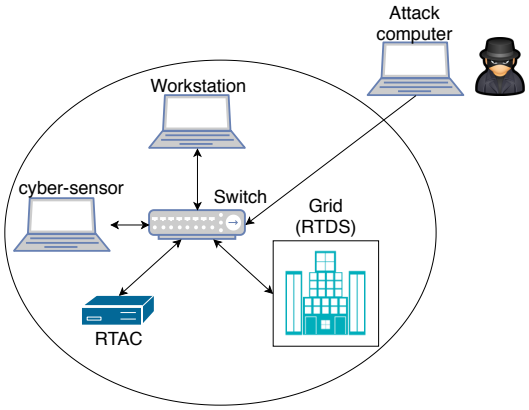


Fig. 4: Test-bed diagram.

$$\text{Ind}(\boldsymbol{X}, k) = \begin{cases} 1, & \text{if } \alpha \leq \text{cdf} \left( X_{[i,j]} | \lambda^{(k)}_{[i,j]} \right) \leq 1 - \alpha \\ & \text{for all } i, j \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

$$\text{Anomaly}(\boldsymbol{X}) = \begin{cases} 0, & \text{if Ind}(\boldsymbol{X}, k) = 1 \text{ for any } k \\ 1, & \text{otherwise} \end{cases} \tag{6}$$

Eq. 5 checks if a sample $\boldsymbol{X}$ is covered by component $k$. Eq. 6 defines an anomaly as a sample that is not covered by any component.

## IV. Experiments

**Dataset:** Figure 4 shows the diagram of the testbed used to collect the data used in the experiments. The testbed consist of a simulated SCADA microgrid that communicates with a set of RTAC controllers and a data historian using DNP3 protocol. An attack computer performs a series of scheduled cyber-attacks while recording the time stamps when the attacks are executed. The time-stamps are used to label the data by segmenting the packets between normal communication and attack (abnormal) communication. Label data was only used for evaluation and benchmarking purposes. Training of the PM-ADS model is completely unsupervised. All devices are connected using a switch. The PM-ADS cyber-sensor is also connected to the switch to sniff the communication packets and collect the data for analysis.

The dataset includes packets following cyber-attacks: 1) IP scan, 2) Port scan, 3) Replay attack, 4) DOS attack.

**Baseline:** We compared our approach with standard machine learning algorithms. For comparative analysis, we extracted a set of cyber features which are used as input to different types of machine learning algorithms. The feature extraction was carried out on TCP packet stream using windowing technique [7]. The TCP packet stream is considered as a time series. A set of statistical features were extracted using a set of neighboring packets within a one second time window. The extracted set of features are presented in Table II.

TABLE I: Window based TCP packet stream features

| Feature Name | Feature Description |
|---|---|
| Packet_rate | Number of packets within a window |
| Num_src_IP | Number of different source IP addresses in a window |
| Num_dst_IP | Number of different destination IP addresses in a window |
| Num_src_port | Number of different source ports in a window |
| Num_dst_port | Number of different destination ports in a window |
| Min_data_length | The minimum data length of packets in a window |
| Max_data_length | The maximum data length of packets in a window |
| Avg_data_length | The average data length of packets in a window |
| Min_win | The minimum window size of packets in a window |
| Max_win | The maximum window size of packets in a window |
| Avg_win | The average window size of packets in a window |
| Min_time_intv | The minimum time gap between packets in a window |
| Max_time_intv | The maximum time gap between packets in a window |
| Avg_time_intv | The average time gap between packets in a window |
| Min_pkt_src | The minimum number of packets per single source IP in a window |
| Max_pkt_src | The maximum number of packets per single source IP in a window |
| Avg_pkt_src | The average number of packets per single source IP in a window |
| Min_pkt_dst | The minimum number of packets per single destination IP in a window |
| Max_pkt_dst | The maximum number of packets per single destination IP in a window |
| Min_ttl | The minimum time to live value of packets in a window |
| Max_ttl | The maximum time to live value of packets in a window |
| Avg_ttl | The average time to live value of packets in a window |
| Num_byt | Number of bytes transmitted by packets in a window |
| Same_src_dst | Number of packets with src IP== dst IP |
| Same_ports | Number of packets with src port== dst port |
| Same_src_src_port | Number of packets with src IP== src port |
| Same_src_dst_port | Number of packets with src IP== dst port |
| Same_dst_src_port | Number of packets with dst IP== src port |
| Same_dst_dst_port | Number of packets with dst IP== dst port |
| Same_IP_port | Number of packets with src IP== dst IP and src port== dst port |
| Num_urg | Number of urgent packets in a window |

The extracted features were used as input to a machine learning model. In this experiment, three machine learning models were used for comparative analysis: 1) One Class Support Vector Machines (OCSVM), 2) Decision tree, 3) Random Forest. Decision trees and Random Forest were used as a classification method to classify data records into two categories: normal and attack.

**Anomaly detection:** In this experiment we evaluated the performance of the PM-ADS to detect anomalies. For the experiment, an anomaly corresponds to any of the cyber attacks executed during the experiment. Normal behavior corresponds to sections where no attack was being executed. We used a $\Delta t$ of one second for the queue. For this experiment, we only used normal data to train the PM-ADS. The objective is to characterize normal behavior and use the model to identify abnormal behavior in the communication (attacks).

Table II shows the accuracy, precision, recall and f1 scores obtained using the presented PM-ADS method and standard machine learning algorithms. The table shows that PM-ADS has comparable performance w.r.t. Decision trees and Random Forest approaches. OCSVM trades precision in order to improve recall and f1 score. An important take away point from these results is that PM-ADS provides comparable performance to standard machine learning methods, even when PM-ADS only uses packet rate information. However, the results suggest that there is room for improving the recall score by including more features into PM-ADS such as packet size, source and destination ports, etc.

TABLE II: Anomaly Detection Performance

| model | accuracy | precision | recall | f1 |
|---|---|---|---|---|
| PM-ADS | 0.991 | 1.000 | 0.888 | 0.941 |
| OCSVM | 0.988 | 0.987 | 0.999 | 0.993 |
| Decision Tree | 0.990 | 1.000 | 0.863 | 0.926 |
| Random Forest | 0.990 | 1.000 | 0.869 | 0.930 |

**Behavior profiling:** One of the main advantages of the presented approach is the inherent interpretability of results of the PM-ADS model. Figure 5 shows the communication behaviors (modes) learned from all data collected during the experiment. For this experiment, we used both normal and attack data to train the PM-ADS in order to profile the communication and discover the salient communication behaviors (modes). Training was performed completely unsupervised, labels are only used to verify what behavior is being represented by the corresponding Poisson component.

Figure 5 shows that the proposed PM-ADS model is able to find distinctive communication patterns for normal and attack scenarios. Results of PM-ADS are easy to visualize. The figure shows a plot of the expected packet rate ($\lambda^{(k)}$) in the form of and adjacency matrix for each communication mode $k$. We can observe that all normal communication falls in mode 11, which is characterized by low rate communication ( 2 packets per second). On the other hand, DOS communication is characterized by very high packet rates (mode 7). IP scan and port scan show a more active communication pattern that involves several devices. The results shown in Figure 5 conform with the expected behavior of normal communication and attacks. The figure also demonstrates the interpretability of the PM-ADS model.

## V. Conclusion

In this paper, we presented the PM-SD model for stochastic anomaly detection. The model provides a data-driven, unsupervised and interpretable approach for behavior profiling of communication networks. Experiments were performed using data measured from a SCADA micro-grid simulation. The

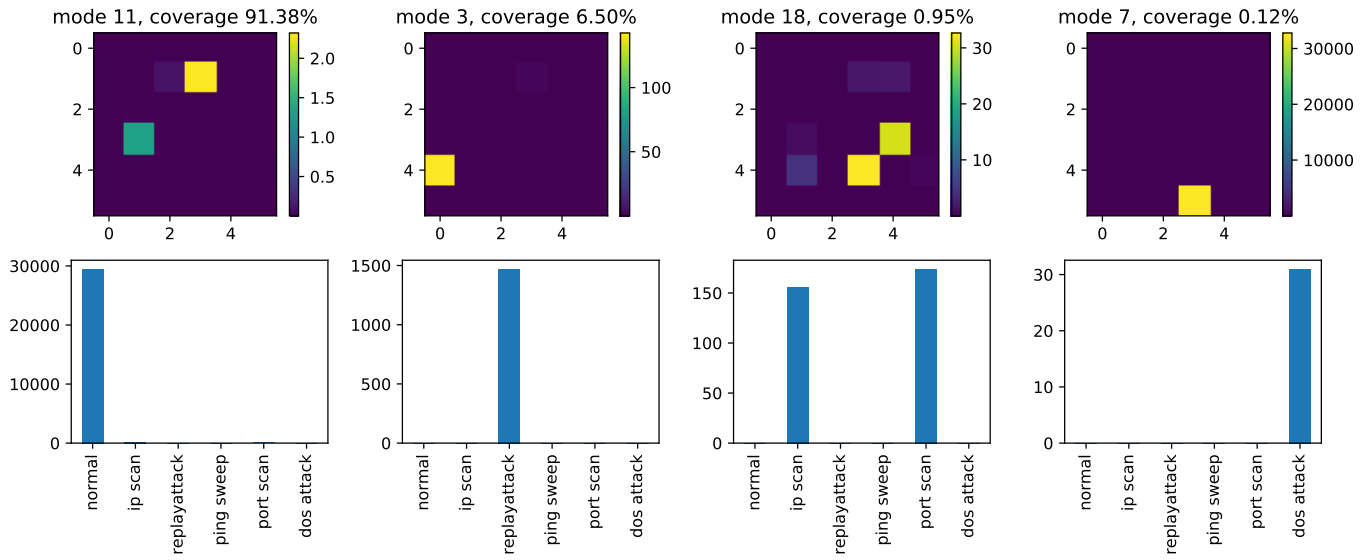


Fig. 5: Communication modes found using Poisson Mixture Model. The figure shows the expected packet rate in the form of an adjacency matrix (top). A bar plot (bottom) shows the type of samples that each particular mode represents.

presented approach is able to find salient communication modes that characterize the system. PM-SD trained with normal behavior data is able to recognize anomalous behavior from common cyber-attacks. Future work will be conducted on expanding the feature set used by the PM-SD model in order to increase accuracy and recall.

## REFERENCES


[1] L. Mookiah, C. Dean, and W. Eberle, "Graph-based anomaly detection on smart grid data," in *The Thirtieth International Flairs Conference*, 2017.

[2] A. Hahn, A. Ashok, S. Sridhar, and M. Govindarasu, "Cyber-physical security testbeds: Architecture, application, and evaluation for smart grid," *IEEE Transactions on Smart Grid*, vol. 4, no. 2, pp. 847–855, 2013.

[3] X. Clotet, J. Moyano, and G. León, "A real-time anomaly-based ids for cyber-attack detection at the industrial process level of critical infrastructures," *International Journal of Critical Infrastructure Protection*, vol. 23, pp. 11–20, 2018.

[4] D. Gunning, "Explainable artificial intelligence (xai)," *Defense Advanced Research Projects Agency (DARPA), nd Web*, vol. 2, 2017.

[5] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proceedings of the 2017 ACM on Asia conference on computer and communications security*. ACM, 2017, pp. 506–519.

[6] C. S. Wickramasinghe, D. L. Marino, K. Amarasinghe, and M. Manic, "Generalization of deep learning for cyber-physical system security: A survey," in *IECON 2018 - 44th Annual Conference of the IEEE Industrial Electronics Society*, 2018, pp. 745–751.

[7] K. Amarasinghe, C. Wickramasinghe, D. Marino, C. Rieger, and M. Manicl, "Framework for data driven health monitoring of cyber-physical systems," in *2018 Resilience Week (RWS)*, 2018, pp. 25–30.

[8] A. Jones, Z. Kong, and C. Belta, "Anomaly detection in cyber-physical systems: A formal methods approach," in *53rd IEEE Conference on Decision and Control*, 2014, pp. 848–853.

[9] J. Goh, S. Adepu, M. Tan, and Z. S. Lee, "Anomaly detection in cyber physical systems using recurrent neural networks," in *2017 IEEE 18th International Symposium on High Assurance Systems Engineering (HASE)*, 2017, pp. 140–145.

[10] R. C. Borges Hink, J. M. Beaver, M. A. Buckner, T. Morris, U. Adhikari, and S. Pan, "Machine learning for power system disturbance and cyber-attack discrimination," in *2014 7th International Symposium on Resilient Control Systems (ISRCS)*, 2014, pp. 1–8.

[11] C.-C. Chang and C.-J. Lin, "Libsvm: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 27:1–27:27, 2011.

[12] "One-class support vector machinesan application in machine fault detection and classification," *Computers and Industrial Engineering*, vol. 48, no. 2, pp. 395 – 408, 2005.

[13] J. Zhang, M. Zulkernine, and A. 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.

[14] C. S. Wickramasinghe, D. Marino, F. Yucel, E. Bulut, and M. Manic, "Data driven hourly taxi drop-offs prediction using tlc trip record data," in *2019 12th International Conference on Human System Interaction (HSI)*, 2019.

[15] L. Wang, X. Zhou, X. Zhu, Z. Dong, and W. Guo, "Estimation of biomass in wheat using random forest regression algorithm and remote sensing data," *The Crop Journal*, vol. 4, no. 3, pp. 212 – 219, 2016.

[16] V. Svetnik, A. Liaw, C. Tong, J. C. Culberson, R. P. Sheridan, and B. P. Feuston, "Random forest: a classification and regression tool for compound classification and qsar modeling," *Journal of Chemical Information and Computer Sciences*, vol. 43, no. 6, pp. 1947–1958, 2003.

[17] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.