General-Purpose Unsupervised Cyber Anomaly Detection via Non-Negative Tensor Factorization

MAKSIM E. EREN, Advanced Research in Cyber Systems, Los Alamos National Laboratory, USA
JUSTON S. MOORE, Advanced Research in Cyber Systems, Los Alamos National Laboratory, USA
ERIK SKAU, Information Sciences, Los Alamos National Laboratory, USA
ELISABETH MOORE, Information Sciences, Los Alamos National Laboratory, USA
MANISH BHATTARAI, Theoretical Division, Los Alamos National Laboratory, USA
GOPINATH CHENNUPATI, Alexa, Amazon, USA
BOIAN S. ALEXANDROV, Theoretical Division, Los Alamos National Laboratory, USA

Distinguishing malicious anomalous activities from unusual but benign activities is a fundamental challenge for cyber defenders. Prior studies have shown that statistical user behavior analysis yields accurate detections by learning behavior profiles from observed user activity. These unsupervised models are able to generalize to unseen types of attacks by detecting deviations from normal behavior, without knowledge of specific attack signatures. However, approaches proposed to date based on probabilistic matrix factorization are limited by the information conveyed in a two-dimensional space. Non-negative tensor factorization, on the other hand, is a powerful unsupervised machine learning method that naturally models multi-dimensional data, capturing complex and multi-faceted details of behavior profiles. Our new unsupervised statistical anomaly detection methodology matches or surpasses state-of-the-art supervised learning baselines across several challenging and diverse cyber application areas, including detection of compromised user credentials, botnets, spam e-mails, and fraudulent credit card transactions.

CCS Concepts: • **Computing methodologies → Anomaly detection**; **Factorization methods**; • **Security and privacy → Intrusion detection systems**.

Additional Key Words and Phrases: anomaly detection, Poisson tensor factorization, non-negative tensor factorization, unsupervised learning, cyber security, CPD, malware, data fusion, ensemble learning, GPU

ACM Reference Format:
Maksim E. Eren, Juston S. Moore, Erik Skau, Elisabeth Moore, Manish Bhattarai, Gopinath Chennupati, and Boian S. Alexandrov. 2022. General-Purpose Unsupervised Cyber Anomaly Detection via Non-Negative Tensor Factorization. *Digit. Threat. Res. Pract.* 3, 9, Article 111 (February 2022), 28 pages. https://doi.org/10.1145/3519602

1 PRIOR PUBLICATION NOTICE

This paper is an extension of work that was originally presented in proceedings of the 18th annual IEEE International Conference on Intelligence and Security Informatics (ISI) by Eren, Moore, and Alexandrov [26]. In this extended

This manuscript has been approved for unlimited release and has been assigned LA-UR-22-21176.

Authors' addresses: Maksim E. Eren, Advanced Research in Cyber Systems, Los Alamos National Laboratory, USA, maksim@lanl.gov; Juston S. Moore, Advanced Research in Cyber Systems, Los Alamos National Laboratory, USA, jmoore01@lanl.gov; Erik Skau, Information Sciences, Los Alamos National Laboratory, USA; Elisabeth Moore, Information Sciences, Los Alamos National Laboratory, USA; Manish Bhattarai, Theoretical Division, Los Alamos National Laboratory, USA; Gopinath Chennupati, Alexa, Amazon, USA; Boian S. Alexandrov, Theoretical Division, Los Alamos National Laboratory, USA.

2576-5337/2022/2-ART111 $15.00
https://doi.org/10.1145/3519602

paper, we improve our prior findings, demonstrate additional use cases, and present a more detailed description of our methods. In comparison to the original publication, we showcase the multipurpose capability of our methods by extending the anomalous authentication event detection results to include the identification of botnet traffic, spam e-mails, and fraudulent credit card transactions. Additionally, we boost the real-world operational value of our methodology by reducing false positive rates using an ensemble of tensors. We also provide a fast Python implementation of the CANDECOMP/PARAFAC Alternating Poisson Regression (CP-APR) algorithm, that was originally implemented in the MATLAB Tensor Toolbox [8]. Python CP-APR, named pyCP_APR[1], can be used via an API similar to Scikit-learn [48], and leverages a *PyTorch* [47] backend for faster computation of tensor decomposition on the GPU, which is essential for analysis of large sparse tensors. pyCP_APR also includes a *Numpy* backend for tensor decomposition of dense and sparse tensors on the CPU. Finally, we differentiate the performance for *link-prediction* and *event-prediction* in our results, and consider the cases where the model cannot score an event or a link due to failed mapping. With the new scoring methods, we present performance results for a wider range of use cases.

2 INTRODUCTION

Detection of cyber anomalies, such as compromised accounts, insider threats, malware traffic, and phishing continues to be a significant challenge for cyber defenders. In 2016, when Turcotte et al. introduced the Poisson matrix factorization model for cyber anomaly detection, 63% of confirmed data breaches involved stolen user credentials [57].This figure has climbed to 80% in 2020 [4], and the average number of yearly security breaches has increased by 67% within the past 5 years [13]. At the same time, the cost of malicious insider attacks increased by 15% in 2019, and still continues to be the type of threat that takes the longest to resolve [13]. Because an insider has fewer security barriers to overcome, a breach that takes minutes to accomplish can take months or years to discover [2]. At the other end of the spectrum, botnets were one of the costliest cybercrimes in 2019 [13], and spam e-mails persist as an extremely effective attack vector. Phishing was involved in 81% of the cyber espionage breaches in 2020 [3]. Meanwhile, over 250 million devices were compromised by a phishing system named *TrickBooster* in 2019, and the prominent Windows threat, *Emotet*, which uses e-mail as the entry point to organization networks was one of the top threats globally in 2019 [6]. When hunting for intruders or malicious insiders on their networks, incident response teams primarily rely on rule-based indicators such as hand-crafted signatures or open-source threat intelligence feeds. Although rule-based indicators perform well when detecting known attacks, they require immense manual work to tune for each enterprise network, and often fail to detect patient and persistent attackers. Currently, alerts are generated only for 9% of attacks [5], and the average cost of a security breach is $3.86 million [1]; therefore, there is an urgent need to improve statistical anomaly detection methods and their associated operational workflows in order to drive increased adoption.

Machine Learning (ML) and user behavior analytics aid in defense against threats by increasing detector effectiveness, reducing response and recovery times, and saving up to 38% in technology spending [12, 13]. However, it is difficult to understand the decisions made by the popular ML systems, such as neural networks, since they are black-boxes [31]. Alternatively, non-negative tensor factorization methods produce interpretable statistical results that can be understood by incident responders. At the same time, generating actionable alerts with anomaly detection systems requires identifying unusual events that correspond to malicious activity. New events happen continually on a network, and no labeled datasets exist with enough detail to build reliable detection systems using supervised learning alone. Because the number of daily events on a corporate network can easily reach into the millions or billions, deployable anomaly detectors must achieve extremely low false alarm rates. Eliminating rare, but benign, events from these alerts is challenging, since human activities are difficult to predict; for example, users authenticate to new network resources, visit new websites, and receive e-mails from new sources on a continual

[1]pyCP_APR is available at https://github.com/lanl/pyCP_APR

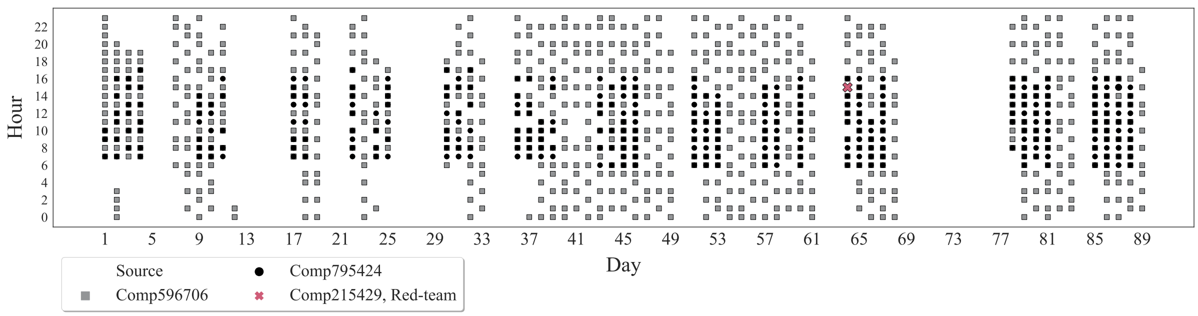


Fig. 1. Hourly authentication events from multiple source computers over 90 days for one compromised user, User087542, in the LANL Unified Host and Network Dataset. The user's activity reveals time- and device-based predictable patterns that deviate from the single anomalous log-on.

basis. Our work builds upon state-of-the-art algorithms for user behavior analysis to build more nuanced methods of normal user behavior over time. We show that our model can accurately detect actions that deviate from the norm by demonstrating its performance in detecting stolen user credentials during a penetration testing event on the Los Alamos National Laboratory (LANL) network, identifying botnet activities hidden among the background HTTP traffic collected from an Internet Service Provider (ISP), detecting real spam e-mails, and recognizing fraudulent credit card transactions, all with the same unsupervised model.

Previous work has shown that recommendation system models based on matrix factorization can identify "peer groups" of users and devices, which allow data-driven predictions of future user actions [21, 46, 57]. Users tend to exhibit a seasonal behavior in the network, where patterns of activities are correlated in time. For example, a simple predictable user behavior is an employee initiating a logon from a desktop computer every weekday, except Friday, at approximately 7:00am. Figure 1 shows activities from such a real, anonymized LANL user (User087542) over 90 days. In this work, we extend peer-based models to include multiple dimensions of an activity profile, such as users, source devices, destination devices, authentication status, IP addresses, and temporal information. We apply tensor factorization to extract complex and nuanced high-dimensional latent activity profiles that provide highly predictive models of user behavior. These models allow us to to improve the sensitivity and specificity of peer-based anomaly detection.

Extending tensor factorization methods in a principled statistical framework allows us to achieve state-of-the-art anomaly detection results on a wide range of cyber security applications, including detecting botnet activities, spam e-mails, fraudulent credit-card transactions, and compromised user credentials. Our unsupervised detection results are compelling because they compete with prior state-of-the-art supervised methods that require labelled malicious activity. As compared to prior supervised methods, our approach is much more general and allows us to detect previously-unseen types of anomalies. Our contributions include:

- Generalizing existing statistical models to *jointly* learn multi-dimensional activity profiles.
- Demonstrating that jointly-learned activity profiles improve the detection of *anomalous events*.
- Presenting state-of-the-art results for *anomalous entity* detection, for example, identifying a penetration testing team's source device within the top 3 most anomalous devices during the month-long test period.
- Performing botnet, spam e-mail, and credit card fraud detection using an unsupervised methodology that competes with prior supervised and semi-supervised solutions.
- Reducing false positive rates via p-value fusion methods over an ensemble of tensors.
- Developing a fast Python implementation of the CP-APR algorithm, named pyCP_APR, that can run on both a CPU and GPU, and provides a user-friendly API similar to the widely used Scikit-learn package.

- Expanding the utility of CP-APR by incorporating an anomaly detection interface in pyCP_APR.

3 BACKGROUND

Our work draws on prior advances in the areas of statistical anomaly detection and tensor factorization. Here, we present a brief summary of related work in both fields.

3.1 Anomaly Detection

Detecting fraud and network intrusions without specific knowledge of the attacker's methods has been a long-standing and vexing problem [45]. The broad range of prior approaches include classical ML-based methods, such as boosting and random forests [43, 49, 60], and deep learning-based methods, such as Generative Adversarial Networks (GANs) [64] and Autoencoder-based approaches [18, 23, 33]. A number of prior approaches have performed anomaly detection in reduced feature space using dimensionality reduction techniques [32, 36, 39, 45]. A variety of classical factorization-based techniques, such as Principal Component Analysis (PCA) [17] and Non-negative Matrix Factorization (NMF) [7] have been applied to detect anomalies using reconstruction error as a metric. PCA and NMF extract "normal" patterns hidden in the data by performing dimensionality reduction. However, reconstruction error-based models lack a direct statistical interpretation, and thus do not directly produce a p-value for anomalies. Statistical models have stronger mathematical guarantees and provide more direct methods for fusing analytic outputs.

Prior studies have explored statistical Poisson Matrix Factorization (PMF) methods, based on recommendation systems and Poisson p-values, to perform anomaly detection [46, 50, 57]. In statistics, the Poisson distribution models the frequency of event occurrences per unit time. The Poisson distribution with rate parameter $\lambda > 0$ models discrete variable as shown in Equation (1). The rate λ is both the mean and variance of the Poisson distribution.

$$X \in \{1, 2, \ldots\} : p(X = x) = \frac{\lambda^x e^{-\lambda}}{x!} \qquad (1)$$

Sanna Passino et al. used two bipartite graphs with the dimensions *User - Destination* and *User - Source*, applied PMF to detect anomalies, and extended their statistical model to incorporate covariates about users and computers [46]. Similarly, Volkovs et al. showed that a deep neural network can learn to augment user preference data and alleviate the problems of cold start [61]. Price-Williams et al. developed a model that detected anomalous users via their historic authentication times [50]. Turcotte et al. demonstrated that Fisher p-value fusion can combine independent p-value scores of user's logon and process start events for anomaly detection [57]. We build on these existing anomaly detection frameworks by combining their ideas with high-dimensional tensor factorization.

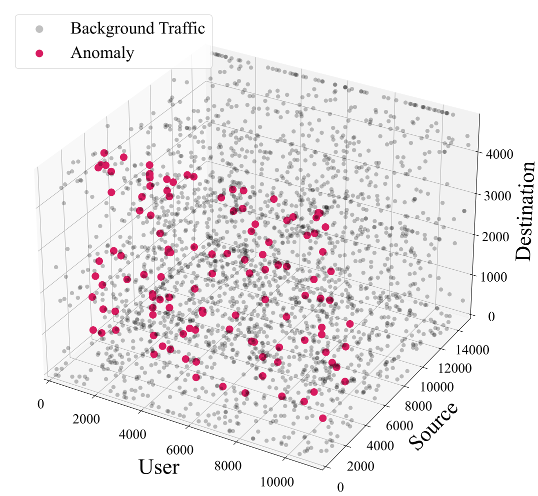


Fig. 2. Binary tensor with the dimensions *User - Source - Destination* from the LANL authentication data. The background traffic is shown with gray and anomalies are highlighted in red. 2% of the original background traffic is shown.

3.2 Tensor Factorization

Tensor factorization is a cutting-edge method for uncovering hidden patterns in data. Tensors are higher order extensions of matrices [35]. Cyber event logs can naturally be encoded as tensors. For example, users authenticating between two devices can be represented by a 3rd-order tensor with dimensions *User*, *Source*, and *Destination*,

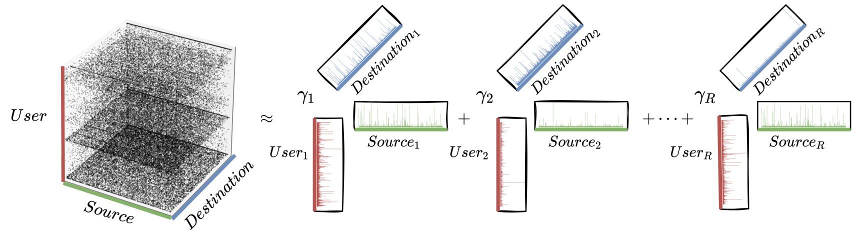


Fig. 3. Rank R Canonical Polyadic Decomposition (CPD) of a tensor with dimensions *User* (u), *Source* (s), and *Destination* (d).

where an index $\mathcal{X}_{u,s,d}$ in this tensor represents the number of authentications that user u performs, going from source device s to destination device d. In this work, we use *binary tensors*, where the element $\mathcal{X}_{u,s,d}$ is set to 1 if user u authenticates from source device s to destination device d at least once, and is set to 0 otherwise. Figure 2 shows such a tensor from the test period of the LANL dataset. We can extend this notation to include D dimensions. For a D dimensional tensor, an index entry is denoted $\mathcal{X}_{i_1,\ldots,i_D}$, where $i_1, \ldots, i_D \in [1 \leq i_1 \leq N_1, \ldots, 1 \leq i_D \leq N_D]$. Following the multi-index notation, we represent a tensor index (i.e. coordinate of non-zero value) with $\mathbf{i}$.

Tensor factorization decomposes high-dimensional data into lower-dimensional components (usually 2D factor matrices), where the factor matrices carry the latent features in each tensor dimension. Specifically, we use a form of non-negative tensor factorization called Poisson tensor factorization. We choose non-negative factorization because our datasets are inherently non-negative (i.e., our datasets involve counts of event types, and a negative event count would be impossible). Importantly, non-negativity requires the extracted latent features to be additive components of the original data, which improves interpretability [37].

Previously, Dunlavy et al. used tensor factorization to perform temporal link prediction of future time steps [25]. In their approach, the temporal profiles captured in the time dimension of a 3-dimensional tensor are utilized as a weighting heuristic or forecasting basis. Differently, we use non-negative tensor factorization and directly incorporate the latent temporal profiles in our link prediction under a statistical framework. Bruns-Smith et al. originally applied Poisson tensor factorization in the cyber security domain [14], but they manually analyzed their resulting factors to find malicious activity. While the authors successfully identified indicators of malicious incidents, their manual analysis does not scale to large data. Our work leverages Canonical Polyadic Decomposition (CPD) to extend existing Poisson matrix factorization models and thus automate ranking and scoring. Similarly, Kanehara et al. used Tucker tensor decomposition [56] with thresholding over the components for automatic detection of botnets in darknet traffic [30]. Outside the cyber domain, Bayesian Poisson Tucker tensor decomposition was previously used by Schein et al. for modeling international relations [54].

CPD [28] is an important tool for unsupervised learning, feature extraction, and dimensionality reduction. By definition, if a tensor can be written as a single outer product of vectors, it has rank 1. Any arbitrary tensor can be decomposed as a weighted sum of rank-1 tensors, which is called Polyadic Decomposition. If the number R of rank-1 tensors is minimal, then the decomposition is a CPD. Importantly, in the non-negative case, a best rank-R approximation always exists [38], and it is almost always unique [51]. Usually, each factor is normalized to sum to 1, and weight is absorbed by γ_r to achieve a unique solution. For example, an order 3 tensor $\mathcal{X}$ with dimensions u, s, d and shape N_u, N_s, N_d can be approximated by a sum of R rank-1 tensors, each called a *component*. Each component is encoded as the outer product of 3 factor vectors, $\theta_r^{(u)}, \theta_r^{(s)}, \theta_r^{(d)}$, with lengths N_1, N_2, and N_3, respectively. Equation (2) shows the CPD tensor approximation, where $\circ$ represents vector outer product, and Figure 3 illustrates the equation.

$$\mathcal{X} \approx \hat{\mathcal{X}} \equiv \sum_{r=1}^{R} \gamma_r \cdot \theta_r^{(u)} \circ \theta_r^{(s)} \circ \theta_r^{(d)} \tag{2}$$

Furthermore, we can write $\hat{\mathcal{X}}$ in KRUSKAL tensor format as $\hat{\mathcal{X}} \equiv \mathcal{M} = [\![\gamma \; ; \; \mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \cdots, \mathbf{A}^{(d)}]\!]$. Here $\mathbf{A}^{(d)} = [\theta_1^{(d)}, \theta_2^{(d)}, \ldots, \theta_R^{(d)}] \in \mathbb{R}^{\mathrm{R} \times \mathrm{N_d}}$ is a matrix of R latent factors for the dimension d. We can let $nnz(\mathcal{X})$ be the number of non-zero entries in $\mathcal{X}$, and let Ω be the set of all entries in the tensor including the zeros, such that for the sparse tensors $nnz(\mathcal{X}) \leq |\Omega|$. We can acquire $|\Omega|$ by taking the product of each dimension's size as shown in Equation (3).

$$|\Omega| = \prod_{d=1}^{D} N_d \tag{3}$$

The problem of selecting the optimal rank R for a specific application is essential in finding low-dimensional latent tensor representations. If we perfectly reconstruct the input tensor, our tensor factorization carries little information about peer groups or other shared structures; if our rank is too low, we lose vital information. Zhado et al. discussed this problem for CP decomposition, and introduced a tuning parameter-free probabilistic model for automatic rank determination of incomplete tensors [65]. Truong et al. has also discussed this problem, and introduced the Non-Negative RESCAL [55]. Minimal multi-rank in RESCAL is chosen to be the rank with low relative error and high silhouette score. This technique was later used to discover the latent topics in a corpus via an ensemble of frequency–inverse document frequency (TF-IDF) matrices by Vangara et al. [59], and a distributed software package implementing the method was released [10, 11, 19]. Similarly, our model attempts to automatically find the best rank on the LANL dataset so as to avoid arbitrary rank selection. Rather than RESCAL, we use log-likelihood on held-out validation data to find the optimal tensor rank for link prediction and anomaly detection. When computationally feasible (only for the LANL dataset), we performed automatic rank selection.

We compare the anomaly detection performance of the tensor with automatically discovered optimal rank to an ensemble of tensors with different ranks. Ensemble tensor learning was first proposed by Kisil et al. [34]. The authors re-grouped the ensemble of latent factors from tensor decomposition to train classifiers, where each decomposition carried particular hypotheses about the data. The trained models then performed majority voting during classification to boost prediction accuracy by exploiting the idea of *wisdom of the crowd*. The ensemble approach, with a probabilistic framework, has also been previously used to improve intrusion detection by combining results from multiple ML algorithms [52]. Alternatively, in our work, we train an ensemble of tensors with different ranks and apply p-value fusion over their predictions to capture the hypothesis carried by each rank.

4 MULTI-DIMENSIONAL ANOMALY DETECTION

Simultaneous extraction of latent features by tensor factorization enhances the detection of unusual activities by making the system sensitive to different correlations between the dimensions. For example, we can train our models to learn user patterns and daily/hourly periodicity jointly.

Our model is based on Poisson CPD. For a D-dimensional tensor with shape $N_1, \ldots, N_D$, we model each element as an independent draw from a Poisson distribution, where the rate $\lambda_{\mathbf{i}}$ is determined by a CPD of rank R:

$$\mathcal{X}_{\mathbf{i}} \sim \text{Poisson}\,(\lambda_{\mathbf{i}}) \tag{4}$$

$$\lambda_{\mathbf{i}} = \sum_{r=1}^{R} \gamma_r \prod_{d=1}^{D} \theta_{r,i_d}^{(d)} \tag{5}$$

where $\theta_r^{(d)}$ is r-th component in the d-th dimension (or factor).

During training, we learn latent factors to maximize the *joint log-likelihood* of all observed counts:

$$\log P(\mathcal{X}) = \sum_{i_1=1}^{N_1} \cdots \sum_{i_D=1}^{N_D} ((\mathcal{X}_{\mathbf{i}} \cdot \log \lambda_{\mathbf{i}}) - \log \Gamma(\mathcal{X}_{\mathbf{i}} + 1)) - \Lambda \tag{6}$$

where

$$\Lambda = \sum_{r=1}^{R} \gamma_r \left[\left(\sum_{i_1=1}^{N_1} \theta_{r,i_1}^{(d)} \right) \cdot \ldots \cdot \left(\sum_{i_D=1}^{N_D} \theta_{r,i_D}^{(d)} \right) \right] \tag{7}$$

and Gamma function

$$\Gamma(n) = \int_0^{\infty} x^{n-1} \cdot \epsilon^{-x} dx \tag{8}$$

Note that this log-likelihood function is efficient to compute on sparse data because the first two terms are 0, whenever the count $\mathcal{X}_{\mathbf{i}}$ is 0. Therefore, the sum can be implemented efficiently by summing only over non-zero (i.e. observed) counts. We use one of the most efficient algorithms to optimize this sparse Poisson likelihood function: CANDECOMP-PARAFAC Alternating Poisson Regression (CP-APR), which minimizes the Kullback–Leibler divergence with a non-negativity constraint via a modified multiplicative update (MU) algorithm [20].

4.1 Rank Selection

CPD is a non-convex problem, where it is assumed that the tensor rank is known. We use log-likelihood Equation (6) evaluation on held-out time periods (i.e. validation data) to find the rank that best predicts future user actions. We fit tensor factorization using the training set on all ranks from 1 to 100 (with a step size of 5 between 10-100), and evaluate log-likelihood on the validation set. The rank with the highest log-likelihood is chosen as R during our subsequent training and testing procedures over the dataset used for identification of compromised credentials. After the rank selection, we combine the train and validation sets to fit the final model. On the other datasets, we choose the rank based on the GPU memory space availability.

4.2 Poisson Rate Smoothing

Because tensors representing cyber security logs are extremely sparse, we encounter numerical underflow when estimating tensor factorization. In order to alleviate this problem, we inflate our binary entries such that the mean value in the tensor is approximately 1:

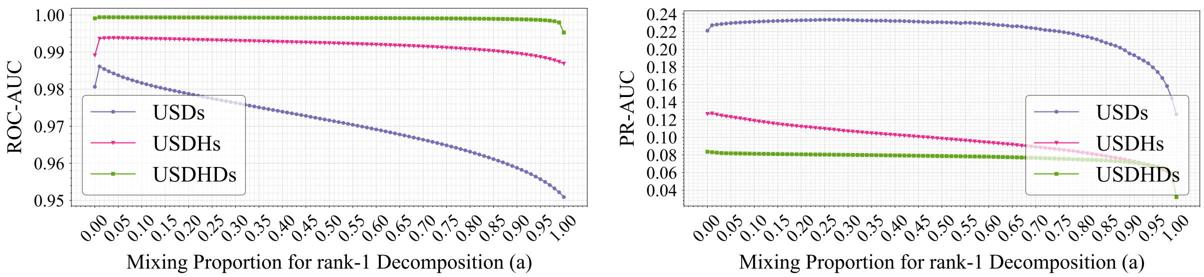
$$\mathcal{X}_{\mathbf{i}} = \frac{\prod_{d=1}^{D} N_d}{nnz(\mathcal{X})} \tag{9}$$

Additionally, because of the sparse structures of these tensors, many of the estimated factors are also sparse (i.e. have a large number of zero values). Zero values in the factors result in estimated Poisson rates of 0 during the testing phase; thus, we need to regularize our estimation procedure. We do this by estimating a rank-1 factorization and a rank-R factorization of the training tensor, where the optimal R is computed by maximizing validation log-likelihood or it is chosen to be the largest R where the tensor can still fit on GPU memory. Since the sum of counts across any axis of our tensor is non-zero, we are guaranteed to have non-zero factors in our rank-1 factorization. We use this fact to regularize our estimation of the Poisson rate $\lambda_{\mathbf{i}}$ based on the rank-1 rate $\lambda_{\mathbf{i}}^1$ and the rank-R rate $\lambda_{\mathbf{i}}^R$:

$$\lambda_{\mathbf{i}} = a \cdot \lambda_{\mathbf{i}}^1 + (1 - a) \cdot \lambda_{\mathbf{i}}^R \tag{10}$$

Here a is used as a mixing proportion to select the amount of information to be used from the rank-1 factorization. Throughout our experiments, we chose $a = 0.1$ heuristically as a type of smoothing. Since rank-1 factorization would be under-fitting the solution, we choose a to be a small value such that our solution contains a higher proportion of the rank-R factorization. For many problems, this parameter could be optimized via cross-validation, but we find that the performance is fairly stable for lower values of a as shown in Figure 4a and Figure 4b.

We perform anomaly detection by computing the *p-value* of each observed count during our test period. The p-value is the probability of observing a count *at least as extreme* as the observed value, under the model learned



(a) ROC-AUC

(b) PR-AUC

Fig. 4. ROC-AUC (a) and PR-AUC (b) scores as a function of the mixing proportion for rank-1 decomposition (a from Equation 10) for the LANL authentication tensors. ROC-AUC and PR-AUC scores are fairly stable for lower values of a, and begin to drop as more information from rank-1 decomposition is used.

during training time[2]: $P(X_{\mathbf{i}} \geq x \mid \lambda_{\mathbf{i}})$. That is, our null hypothesis is that a user's behavior will follow our previously learned activity profile. A lower p-value is an indication of an anomalous event.

4.3 P-value Fusion over an Ensemble of Tensors

Choosing an optimal tensor rank is necessary to find the decomposition that best describes the latent space in the data; we lose crucial information if the rank is too low (under-fitting), and unwanted noise is extracted if the rank is too high (over-fitting) [59]. Therefore, rank selection is an important open research area that is widely studied. We use log-likelihood to find the optimum rank for the LANL dataset in this paper. Additionally, in our analysis, we show that the knowledge extracted from decomposition with different ranks can be combined to enhance prediction accuracy.

Tensor decomposition with different ranks extracts distinct hidden features of the data, each capturing unique patterns. Therefore, each rank carries a certain hypothesis about the underlying information. We use p-value fusion techniques to unify these extracted patterns to improve our decisions. We define an ensemble of tensors to be a group of tensor decomposition of rank 2 through R, such that $G_{\mathcal{M}} = \{\mathcal{M}^2, \mathcal{M}^3, \mathcal{M}^4, \ldots, \mathcal{M}^R\}$. Each tensor in this ensemble follow the smoothing and regularization steps outlined in Section 4.2. Using the ensemble of tensors, we can calculate group of Poisson λ parameters for each link (or tensor entry) $\mathbf{i}$, in the test set, such that $G_{\lambda,\mathbf{i}} = \{\lambda_{\mathbf{i}}^2, \lambda_{\mathbf{i}}^3, \lambda_{\mathbf{i}}^4, \ldots, \lambda_{\mathbf{i}}^R\}$. With $G_{\lambda,\mathbf{i}}$, we can calculate the group of p-values for the non-zero value on each link $\mathbf{i}$, $G_{p,\mathbf{i}} = \{p_{\mathbf{i}}^2, p_{\mathbf{i}}^3, p_{\mathbf{i}}^4, \ldots, p_{\mathbf{i}}^R\}$. These p-values are then fused with Fisher Equation (11), Harmonic mean Equation (12 where $w_r = \dfrac{1}{R-1}$), and Arithmetic mean Equation (13) methods to get a combined solution:

$$X_{2R_{\mathbf{i}}}^2 \sim -2 \sum_{r=2}^{R} ln(p_{\mathbf{i}}^r) \qquad (11)$$

$$\mathring{p}_{\mathbf{i}} = \frac{\sum_{r=2}^{R} w_r}{\sum_{r=2}^{R} \frac{w_r}{p_{\mathbf{i}}^r}} \qquad (12)$$

$$\mathring{p}_{\mathbf{i}} = \frac{1}{R-1} \sum_{r=2}^{R} p_{\mathbf{i}}^r \qquad (13)$$

Note that each method for p-value fusion is based on statistical assumptions that are invariably violated to some extent in practice. For instance, Fisher p-value fusion (Equation 11), assumes that the p-values to be fused are independent, which is clearly not the case when fusing tensor decomposition results over different ranks for the

[2]This p-value can be computed by the Poisson survival function.

same data. Thus, we experimentally evaluate multiple p-value fusion methods, providing empirical evidence to inform the choice of p-value fusion method.

4.4 Scoring

To better understand the performance of our methods, we score the anomaly detection results both based on *link* and *event* prediction, and consider the cases when the system cannot produce a decision for a particular entity based on the information collected at training time, for example for new users or devices on the network. We also apply p-value fusion technique over the tensor dimensions to identify anomalous entities such as a single compromised host or user.

4.4.1 Accounting for Unknown Instances: When we construct our tensors, we create mappings of each categorical or numerical instance to certain indices in the dimensions. These mappings are created using the information in the training set since it is impossible to know the data that we will see in the test set. This method is analogous to a real-world scenario where the tensors are built using the data at hand (training set) and used to evaluate new logs in the future (test set). With this setup, it is possible to encounter an entity that was not seen previously; such that it does not have a corresponding index mapping in the tensor. In this case, we cannot evaluate an event associated with that entity. For example, a new user might be added to the network after the tensor is built. Since we cannot evaluate events for this user, they would be missed by our system, which is equivalent to classifying them as benign.

In a production environment, our system would be periodically updated to carry the most recent information; however, it is still important to account for such cases when evaluating the anomaly detection performance of our methods because we want to penalize the system for each anomalous activity that we could not detect. This is accomplished by adding back all the skipped instances during scoring and setting their predicted labels to benign, or equivalently to a value that is greater than the maximum p-value for the scored activities.

4.4.2 Event Prediction vs. Link Prediction: *Event* scoring accounts for each observed action (e.g. log event) individually[3]. The benefit of event scoring is that it rewards the system for each benign action that is not detected, and penalize it for each malicious activity that is not detected. For example, if a stolen credit card is used multiple times to purchase goods, we would like to reward the system for detecting each unauthorized transaction. The sample log lines in Figure 5 contains 3 individual events.

```
...
U:User000089 S:Comp703328 D:ActiveDirectory H:1 D:0 s:Fail
U:User000089 S:Comp703328 D:ActiveDirectory H:1 D:0 s:Fail
U:User000089 S:Comp703328 D:ActiveDirectory H:1 D:0 s:Success
...
```

Fig. 5. Example event logs for anonymized user User000089 from LANL authentication dataset. User000089 successfully logs in to Active Directory from the source device Comp703328 on the third try at 1am (H:1) on a Monday (D:0).

Link scoring corresponds to evaluating each *unique* log (coordinate of non-zero value **i**) within the test set, similar to distinct edges in a graph. This is equivalent to rewarding the system once for each day during which it detects a stolen credit card number (assuming the card is not automatically disabled upon the first detection).

The authentication logs in Figure 5 show 2 unique links for the 6 dimensional tensor: *(User000089 - Comp703328 - ActiveDirectory - 1 - 0 - Fail)* and *(User000089 - Comp703328 - ActiveDirectory - 1 - 0 - Success)*. Differently, there is a single unique link for a 3 dimensional tensor: *(User000089 - Comp703328 - ActiveDirectory)*.

[3]Event scoring can be efficiently implemented by setting sample weights, e.g. using Scikit-learn's `sample_weight` parameter when computing the ROC and PR curves [48].

Table 1. Tensor details and test set p-value statistics for anomalous and benign events

	Tensor Details			Anomaly p-value			Benign p-value		
Dataset & Tensor	Dimensions Size	% Non-Zero	Decomposed Rank	Mean	Std	Count	Mean	Std	Count
LANL US	11260 x 15055	2.57×10^{-4}	20	.1993	.3253	76	.8945	.2421	31,241
LANL UD	11260 x 4796	1.51×10^{-3}	20	.6399	.3315	117	.9489	.1829	69,596
LANL USDs	11260 x 15055 x 4796 x 2	1.02×10^{-7}	4	.2721	.4090	119	.9575	.1677	125,166
LANL USDHs	11260 x 15055 x 4796 x 24 x 2	3.04×10^{-8}	5	.1062	.2621	137	.9801	.1215	955,808
LANL USDHDs	11260 x 15055 x 4796 x 24 x 7 x 2	1.60×10^{-8}	45	.0175	.0765	138	.9946	.0664	3,513,527
UGR'16 Neris 3&2 Octet Src&Dest IP Mapping	7453770 x 65536 x 24 x 7	7.32×10^{-7}	8	.0465	.1998	3,001	.9717	.1516	20,117,426
UGR'16 Neris 20-Bits IP Mapping	655360 x 522429 x 24 x 7	1.04×10^{-6}	10	.0262	.1425	6,381	.9659	.1478	23,383,989
UGR'16 Neris 24-Bits IP Mapping	3865526 x 848382 x 24 x 7	1.09×10^{-7}	7	.1464	.3008	2,369	.9822	.1034	21,847,564
UGR'16 Neris 4 Character IP Hash Mapping	65536 x 65536 x 24 x 7	8.04×10^{-5}	10	.0292	.1246	8,381	.9447	.2065	23,189,409
UGR'16 Neris 5 Character IP Hash Mapping	1048487 x 663889 x 24 x 7	5.16×10^{-7}	7	.0330	.1599	5,781	.9732	.1262	23,250,847
UGR'16 Neris 6 Character IP Hash Mapping	7477572 x 1019015 x 24 x 7	4.72×10^{-8}	6	.2813	.4315	495	.9857	.0922	19,481,318
UGR'16 Spam E-Mail	55287 x 65536 x 24 x 7	2.66×10^{-5}	20	.3814	.2165	2,495	.9791	.1220	1,909,544
PaySim Credit Card	100 x 5 x 24 x 7 x 100 x 100	9.00×10^{-6}	25	.6826	.4387	4,391	.9998	.0058	1,224

4.4.3 Entity Prediction: To make operational use of the anomaly scores produced by our system, we need to summarize these results into the detection of malicious entities, such as stolen user credentials, compromised bastion hosts, or stolen credit card numbers. We achieve this summarization using p-value fusion of *dependent* p-values over the dimensions of the tensor [62]. This fusion is accomplished by taking the harmonic mean over all p-values, including the p-values for unobserved events (which are, by definition, 1). Note that fusion can either produce a ranked list of entities (e.g. users, source devices, destinations, days, etc.) or reduce to a lower-dimensional set of events (e.g. user-source and user-destination interactions, etc.) using:

$$\mathring{p}_T = \frac{\prod_d^{\mathcal{D}-\mathcal{T}} N_d}{\sum^{\mathcal{D}-\mathcal{T}} \frac{1}{P(X_\mathbf{i} \geq x_\mathbf{i})}} \tag{14}$$

where $\mathcal{T}$ is set of target dimensions that we want to fuse down to, $\mathcal{D}$ is the set of all dimensions, and N_d is the size of dimension d.

Finally, we find that fusing the ranked lists produced by multiple multi-dimensional tensors allows us to identify multiple complementary aspects of anomalous behavior, and thus achieve better results than identifying anomalies with any one tensor alone. For fusing ranked lists, we use mean reciprocal rank (MRR), where rank_i is the rank of the entity in the i^{th} ranked list [22]:

$$\text{MRR} = \frac{1}{|N|} \sum_{i=1}^{|N|} \frac{1}{\text{rank}_i} \tag{15}$$

5 DATASETS

We select three diverse, cyber-relevant datasets to test our methods, in order to demonstrate the generality and novelty of our approach. We show that our methods generalize to different anomaly detection problems by naturally learning activity patterns from past behavior and detecting deviations from these learned behavior profiles. Specifically, we study three types of data with unique properties: host authentication events, netflow records, and banking transactions. These datasets are used for four tasks, detecting compromised hosts and users, botnets, spam e-mails, and fraudulent credit card transactions. In this section, we describe each of the datasets, provide the relevant statistics of the data, and explain our pre-processing steps.

Table 2. LANL Authentication dataset attribute counts and selected days for dataset splits

Set	User	Source	Destination	Benign Events	Anomalous Events	Fail %	Days
Train	11,118	14,705	4,698	166,712,680	0	2.13	1-48
Validation	9,181	10,778	3,508	28,013,171	0	12.68	49-56
Train + Validation	11,260	15,055	4,796	194,841,640	0	1.82	1-56
Test	10,165	12,526	4,176	91,547,561	179	3.88	57-82

5.1 Los Alamos National Laboratory (LANL) Authentication Dataset

Detailed and diverse datasets at the enterprise scale are rare in the cyber security domain due to privacy and security concerns. Turcotte et al. introduced the publicly available Unified Host and Network Dataset[4] to address this critical need [58]. The dataset contains host events and netflow logs collected over a 90-day period at Los Alamos National Laboratory (LANL), including red team activity that occurred from days 57 to 82. This red team activity provides ground truth information for the evaluation of anomaly detection techniques. Attributes in the dataset are anonymized, but the dataset curators evaluated the anonymization with a small set to ensure that entities can be joined across the dataset, thus ensuring that the collection remains meaningful for research purposes.

While the LANL dataset contains both network and host data, our work focused on a subset of the host data. We base our work on the 3.5 million daily average user authentication events collected by the Windows Logging Service (WLS) at endpoint devices in the LANL dataset. We filter the dataset to include only *EventID* 4624 and 4625 which are collections of various types of successful and failed logon records. In particular, we limit *LogonType* to *Interactive*, *Network*, *Batch*, *Service*, *Unlock*, *NewCredentials*, *RemoteInteractive*, and *CachedInteractive*[5]. We disregard events performed by local and system processes (instances where the *UserName* ends with "$") to minimize the presence of automated activity. We extract the following attributes from the remaining authentication data, to be used as dimensions in our tensors:

- *UserName*, user which initiates the log-on.
- *Source*, device where the authentication originates.
- *LogHost*, destination device to be authenticated to.
- *EventID*, fail or success status of the authentication.
- *Time*, timestamp of the event.

Additionally, daylight savings time occurs at day 42 in the LANL dataset, shown in Figure 1. As a result, we increment hours by 1 after day 42 at 2:00 am to normalize the time. Finally, we drop any data instances with missing values. We split all extracted data instances by time into training, test, and validation sets. The validation set is used in rank selection. The attribute sizes and day distributions for each split are shown in Table 2.

We build three separate binary tensors with dimensions *User - Source - Destination - status* (USDs), *User - Source - Destination - Hour - status* (USDHs), and *User - Source - Destination - Hour - Day - status* (USDHDs). We also build two matrices *User - Source* (US) and *User - Destination* (UD)[6]. The *status* indicates failed or succeeded logon activity. The *Day* dimension is day of the week (Monday through Sunday), and *Hour* is hour of the day (0 through 23). *User* represents the account which initiates the authentication event (e.g., *UserName*), *Source* and *Destination* are the origin and target devices of the log-on event (*Source* and *LogHost*, respectively, in our data). Tensor entries are binary: an entry of 1 indicates the presence of at least one event. Table 1 shows statistics for each tensor.

---

[4]The LANL dataset is available at https://csr.lanl.gov/data/2017/.
[5]Detailed attribute descriptions can be found in [58].
[6]Matrices with the *User*, *Source*, and *Destination* dimensions were first used by Turcotte et al. [57], and Sanna Passino et al. [46].

Table 3. UGR'16 attribute counts and selected weeks for dataset splits

Set	Source IP	Destination IP	Benign Events	Anomalous Events	Number of Weeks
Neris Botnet Train	9,900,350	1,050,838	1,326,645,343	0	14
Neris Botnet Test	3,471,677	678,513	405,532,210	50,185	5
Spam E-Mails Train	124,762	2,473,478	107,084,432	22,868,251	14
Spam E-Mails Test	53,993	1,222,161	30,397,634	36,159,948	5

Authentication events result in immensely sparse problems, where a large fraction of the tensor is composed of zeros. This is common for cyber data because the majority of network resources communicate with a only small set of devices or users. Zero values that comprise the majority of the tensor do not need to be stored in memory, allowing us to deviate from dense tensor representation. Instead, sparse tensors can be stored as a list of coordinates and a corresponding list of non-zero values, known as the COOrdinate *COO* format. We store coordinates of the non-zero values with element-to-index mappings of the categorical dimensions. Entities that do not exist in the training data are not mapped in the corresponding test set; however, we also evaluate our system for each skipped unknown entry as described in Section 4.4.1.

5.2 UGR 16 NetFlow Dataset

The UGR'16 dataset[7], was developed by Fernandez et al. to evaluate cyclostationary intrusion detection systems (i.e. IDSs that analyze events based on long-term temporal activity patterns such as day/night, weeks, and months) [41]. While the LANL dataset contains the authentication event logs originated at the endpoint devices, the UGR'16 dataset is made up of anonymized NetFlow events collected at the border routers of a Tier 3 Internet Service Provider (ISP). Several properties of this dataset including the real 19-week long background network traffic make it an ideal candidate for a second (and third) realistic application of our methods.

The background traffic in this dataset originates from different companies and a variety of applications. Thus, it represents network traces from a wide range of real Internet user profiles. Heterogeneous network traffic introduces a real-world challenge to the evaluation of our methods. Another important distinction of this dataset is its size, reaching up to 1.3 billion events in the 14-weeks long training period as shown in Table 3, further emphasizing the importance of speed and memory efficiency of tensor factorization methods.

We use the last 5 weeks as the test set, which includes both the synthetic and real attack traffic. Controlled attack traffic is generated with modern techniques and tools by the curators to address the need for datasets with up-to-date incidents. Meanwhile, the real attack traffic is labeled via anomaly detection tools based on PCA and One-Class Support Vector Machine (OCSVM), and cross-referencing open-source threat intelligence feeds for blacklisted IP addresses. Importantly, the background traffic is not proven to be benign. This means that it is possible to have unlabeled malicious activities in both the training and test period making our reported scores relatively inaccurate. However, we see this fact as an added benefit since it closely resembles a real-world scenario where clean data is not guaranteed.

We ignore all of the known malicious incidents during the training period and extract two types of network traffic from the dataset targeting Botnet and SPAM e-mail detection. Each of the binary tensors that are built using this dataset are represented in sparse format and has the dimensions *Source IP -Destination IP - Hour - Day*. The first two dimensions, *Source IP* and *Destination IP* represent the source and destination IP addresses of the devices from network activity. Temporal dimensions *Hour* and *Day* follow the same day/hour format as the tensors from the LANL dataset. Similar to the LANL dataset, the entities that are encountered for the first time during the test period are skipped.

[7]The UGR'16 dataset is available at https://nesg.ugr.es/nesg-ugr16/.

Table 4. IP mapping schemes used in the UGR'16 dataset for the *Source IP* and *Destination IP* dimensions on the tensors with the dimensions *Source IP - Destination IP - Hour - Day*.

Dataset & Tensor	**Source IP**	**Destination IP**
UGR'16 Neris 3&2 Octet Src&Dest IP Mapping	Lower 3 octets	Lower 2 octets
UGR'16 Neris 20 Bits IP Mapping	Lower 20 bits	Lower 20 bits
UGR'16 Neris 24 Bits IP Mapping	Lower 24 bits	Lower 24 bits
UGR'16 Neris 4 Character IP Hash Mapping	Lower 4 characters of the MD5 hash	Lower 4 characters of the MD5 hash
UGR'16 Neris 5 Character IP Hash Mapping	Lower 5 characters of the MD5 hash	Lower 5 characters of the MD5 hash
UGR'16 Neris 6 Character IP Hash Mapping	Lower 6 characters of the MD5 hash	Lower 6 characters of the MD5 hash
UGR'16 Spam E-Mail	Lower 2 octets	Lower 2 octets

5.2.1 Neris Botnet: The first task we tackle using the UGR'16 dataset is detecting the Neris Botnet activity hidden in benign HTTP traffic. Hosts infected with Neris sends SPAM and performs Click Fraud [41]. During the pre-processing, we keep the connections where the destination port is 80 (HTTP traffic). This way, our goal is to identify the 61 bots (i.e. compromised devices) that establish an HTTP connection to one of the 70 Command and Control devices. The training period is utilized to learn the expected benign traffic (*background* traffic) over port 80, such as website visits, and the test period is filtered to contain the connections labeled both *background* and *nerisbotnet*.

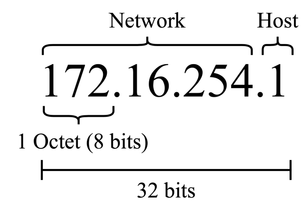


Fig. 6. Sections of an IP address for IPv4 Class C subnet. The lower two octets are used together as the subnet and host identifier.

We map the IP addresses to tensor indices targeting two outcomes: lowering the tensor size, and grouping the IP address communities based on both the network and host identifiers. Using an IP address directly as the tensor dimension would not scale in a production system. Because an IP address contains a total of 32-bits, the size of the tensor dimension for the IP address could grow up to 2^{32}. This could cause memory space and computation speed issues as the new IP addresses are introduced to the system. Therefore, we apply different mapping techniques and measure each of their performance when detecting anomalies.

An IP address contains 4 sections, each of which is 1 octal or 8 bits. In the Class C subnet, the upper 3 octets are the network field and the lower octal is the host address field. This is also illustrated in Figure 6. We target different lower number of bits that contain both the network and host field identifiers during mapping. This mapping forms groups for different physical locations in the network. The mapping technique is also used by Kanehara et al. to build a tensor to analyze the botnet patterns in the latent components extracted using the Tucker decomposition [30]. Differently, they map the upper 16-bits of the IP addresses in their tensor, which misses the information about the host device. We extend this method to map different numbers of lower bits from the IP address. We further extend this mapping method by hashing the IP addresses to analyze the performance of the resulting IP groupings. We build 6 tensors with different mapping schemes for the first two dimensions *Source IP* and *Destination IP*, where the tensor is named based on the mapping scheme, as shown in Table 4. Note that the hashing-based mapping schemes will best extend to 128-bit IPv6 addresses.

5.2.2 Spam E-Mail: We use SMTP traffic in the UGR'16 dataset to identify SPAM e-mails. While the Neris traffic was synthetically injected in the dataset, all of the SPAM e-mails that are labeled by the dataset curators are real attack traffic and include major SPAM campaigns. During the parsing, only the connections to source port 25 are kept. Similar to the Neris pre-processing step, the training set only contains the events labeled as *background* to learn the normal SMTP traffic patterns. The test set contains both the *background* and *anomaly-spam* connections to detect 466 devices targeted by 15 attackers with nearly 36 million SPAM messages, as shown in Table 3. We built a single tensor and used the lower 2 octets of the IP addresses to map the first two dimensions.

5.3 PaySim Banking Transaction Dataset

Banking is another field that faces open-source data scarcity due to privacy concerns. Public credit card transaction datasets are commonly presented in PCA transformed format. This format extracts the latent features of the data and enables the analysis of methods such as classification, while preserving the privacy of the original resource. However, the abstracted data loses interpretability and cannot be used in graph-based analysis methods.

Lopez-Rojas et al. developed *PaySim*, a synthetic transaction generator in response to address the unavailability of public data [40]. Given real transaction logs, *PaySim* can synthesize a new dataset that carries the statistical properties of the original dataset. The generated dataset hides the private information of actual transactions while keeping the analytic value by basing the synthetic instances on real transactions. We use the mobile banking dataset *Synthetic Financial Datasets For Fraud Detection*[8] that is based on real transaction data provided by *Ericsson* to the dataset authors [40].

The data includes 5 types of banking transactions performed between 6,353,307 source and 2,722,362 destination accounts over 743 hours. We use the first 600 hours as the training set to learn the benign/normal 6,252,434 transactions. 6,613 fraudulent events in the training period are moved to the test set for performance evaluation. After moving the anomalies, the test period contains a total of 101,973 benign and 8,213 fraudulent transactions.

With this split, we build an order 6 tensor with the dimensions *Amount* - *Type* - *Hour* - *Day* - *Origin Balance Error* - *Destination Balance Error*. The *Amount* dimension is the total dollars associated with the transaction. *Type* is a categorical dimension mapped from one of the classes: *CASH_OUT*, *PAYMENT*, *CASH_IN*, *TRANSFER*, or *DEBIT*. *Hour* and *Day* are the temporal dimensions for the hour of the day and day of the week. The last two dimensions represent the transaction error rate, inspired by the popular *Kaggle* kernel written for this dataset by Joshua [29], calculated by subtracting the new and the old balance in the account. Finally, the numerical values are mapped to the bins in the dimensions: *Amount*, *Origin Balance Error*, and *Destination Balance Error*, where the bin range is between 0 and 99.

6 EXPERIMENTS AND ANOMALY DETECTION RESULTS

We conducted experiments targeting two main tasks: (1) detecting anomalous events, and (2) detecting anomalous entities. Anomalous events are analogous to log messages. For example, a single anomalous *User*, *Source*, *Destination*, *Hour*, and *Day* combination. As described in Section 4.4.3, anomalous entities are higher-level abstractions, discovered by finding commonalities between multiple anomalous links, such as a single malicious user or a single malicious device.

Our model *does not train against any labels for malicious activity*. Following common practice in user behavior analysis, we assume that the vast majority of events during the training period are benign, and observations during the training period are used to establish a baseline activity profile. Labels were used for the Neris botnet, SPAM e-mail, and credit card fraud tensors during the training time to remove the malicious activity. This is analogous to a security team removing known anomalies from the training data when using our model in a production environment. Incident response teams can use our model as a streaming detector, where daily activities are scored against an existing model, and a batch re-training procedure is undertaken repeatedly to refine the model.

We quantitatively evaluate our detections using the area under the receiver operating characteristic (ROC) curve (ROC-AUC), which evaluates the extent to which the model assigns lower p-values to red team events than benign events, and precision-recall curve (PR-AUC), which essentially measures the model's sensitivity to false positives. In addition, we compare our results to prior anomaly detection research that used the same datasets [15, 24, 42, 44, 46, 63].

[8]The PaySim dataset is available at https://www.kaggle.com/ntnu-testimon/paysim1/.
*The score is reported on a manually balanced set; therefore, not directly comparable to our result.

Table 5. Comparison with state-of-the-art prior methods for anomaly detection using **ROC-AUC**. Across all four tasks studied, we find that our method compares favorably with unsupervised baselines. We provide a best effort to fairly compare our results in terms of *event* and *link* prediction with and without skipped instances. A check-mark (✓) under the **Events** column indicates that the prior research evaluated their method over each event log. In contrast, an X mark (✗) is used if the other work did not consider the particular scoring paradigm. A check-mark under the **Links** column means the method was evaluated when detecting anomalies over unique links (Section 4.4). We place a check-mark under the **Skips** column if the evaluation considered the skipped instances during testing time. Not applicable (or *NA*) is used if the type of scoring does not apply to a particular method. We use **?** if the information was not clearly communicated by the authors, and thus we report it to the best of our understanding.

	Scoring			Their Result				Our Result	
Dataset & Target	Events	Links	Skips	Method	Num. Features	Reference	Score	Tensor	Score
UGR'16 Neris Botnet	✓	*NA*	**?**	PCA (*semi-supervised*)	11	[15]	.884	Neris 4 Character IP Hash Mapping	**.998**
UGR'16 Neris Botnet	✓	*NA*	**?**	Random Forest (*supervised*)	132	[42]	.961	Neris 4 Character IP Hash Mapping	**.998**
UGR'16 Neris Botnet	✓	*NA*	**?**	Variational Autoencoder	53	[44]	.936	Neris 4 Character IP Hash Mapping	**.998**
UGR'16 Neris Botnet	✓	*NA*	**?**	Gaussian Based Thresholding	53	[44]	**.970**	Spam E-Mail	.966
UGR'16 SPAM E-Mails	✓	*NA*	**?**	Variational Autoencoder	53	[44]	.908	Spam E-Mail	**.966**
UGR'16 SPAM E-Mails	✓	*NA*	**?**	Random Forest (*supervised*)	132	[42]	.857	Spam E-Mail	**.966**
UGR'16 SPAM E-Mails	✓	*NA*	**?**	PCA (*semi-supervised*)	11	[15]	.79**?**	Spam E-Mail	**.966**
LANL Auth. (UD)	✗	✓	✗	Poisson Matrix Factorization	2	[46]	.902	USDs	**.956**
LANL Auth. (US)	✗	✓	✗	Poisson Matrix Factorization	2	[46]	.863	US MRR	**.952**
PaySim Credit Card	✓	*NA*	*NA*	L-SVM (*supervised*)	6**?**	[24]	**.978***	Credit Card	.815
PaySim Credit Card	✓	*NA*	*NA*	Ensemble of DBNs (*supervised*)	5	[63]	**.961***	Credit Card	.815

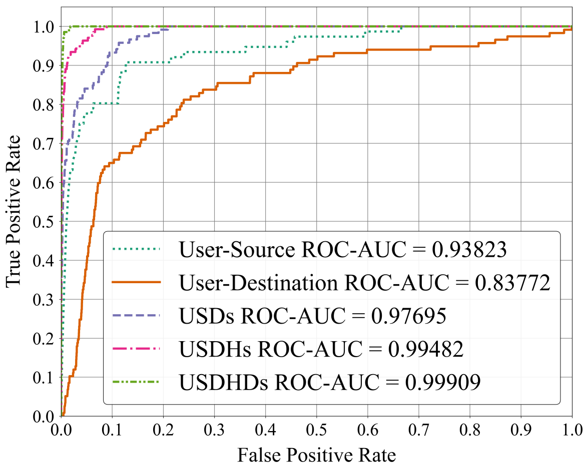
6.1 Anomalous Activity Detection

We demonstrate the value of our anomaly detection methodology through two specific evaluation tasks: (1) comparing tensor-based anomaly detection with prior supervised and unsupervised learning approaches, (2) exploring the performance of our methods for *link* and *event* prediction with and without skipped instances.

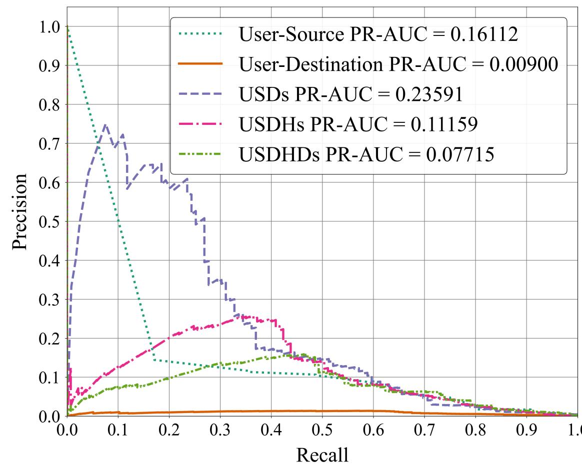
6.1.1 Improved Detection: We compared our tensor-based anomaly detection methods with state-of-the-art baselines on each of our datasets. Table 5 demonstrates that our detection performance, as evaluated using ROC-AUC, matches or surpasses most of the baselines. This comparison includes state-of-the-art supervised learning detection systems, demonstrating surprisingly strong performance for our unsupervised learning method. The experiments demonstrate that our tensor factorization model can identify anomalies across multiple modalities and that adding dimensions to the analysis improves the learning of detailed activity profiles. We detect anomalous events by calculating a p-value for each element in the observed tensor. Table 1 shows statistics for the p-values inferred across anomalous and benign links. Anomalous links have substantially lower average p-values than benign events across all the tensors, which shows that our model discovers meaningful anomalies in an unsupervised manner. Because the LANL authentication dataset has been used in anomaly detection via matrix factorization methods previously [46, 50, 57], we can use the prior work as a reference point in our results. Therefore, we specifically look at the authentication tensors *USDs*, *USDHs*, and *USDHDs* to analyze the added benefit of using tensors in user behavior analysis. We also look at two matrices, *US* and *UD*, that are factorized using pyCP_APR.

When we move from the tensor *USDs* to tensors *USDHs*, and *USDHDs* we add dimensions representing the hour of the day and the day of the week to consideration. Adding the temporal dimensions to the tensor decreases the average p-values for red team events and increases p-values for benign events. Simultaneously, the standard deviation of the p-values drops when the new dimensions are added to the tensor. This result indicates that learning the temporal characteristics of the connections jointly with the peer structure connecting users and their devices enhances detectability. This time-based anomaly detection is novel within a joint statistical framework, and greatly enhances capabilities in applications such as insider threat detection.

Previous work that applied non-negative matrix factorization (NMF) detected anomalies using only two dimensions at a time, such as a *User-Destination* pair [46]. Two-dimensional link prediction methods cannot extract



(a) ROC



(b) PR

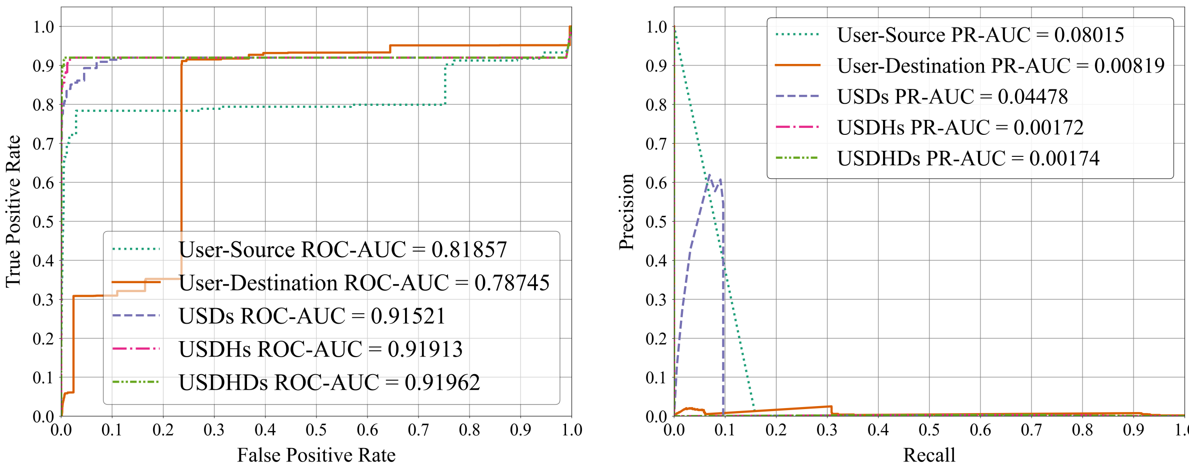
Fig. 7. ROC (a) and PR (b) curve for the LANL authentication tensors with increasing dimension. ROC curve shows that increasing dimensions improves overall ranking of red team events. PR plot highlights the cost of higher false positive rates on red team event detection. Link scoring is performed, and skipped instances are not accounted for.

the multi-faceted details of a user's activity profile. For instance, detection of anomalies via *User-Destination* dimensions alone will miss a malicious event if the only abnormal characteristic of the connection is the *Source* of the authentication event. Using tensors, we jointly learn activity profiles that include all dimensions of a user's behavior, and our experimental results show that the detection improves, with a minimal increase in the computational cost, for tensors with up to 6 dimensions.

Figure 7a shows that adding dimensions improves the *ranking* of links created by the red-team events within the full list of links observed on the network. However, ROC curves for each tensor cannot be compared directly due to the differing number of benign links. Figure 7b shows that our detections are sensitive to *class imbalance*. Although we see an improvement in PR-AUC going from *US* and *UD* to *USDs*, there is an increased false positive rate for the tensors that also include temporal information. For example, with a p-value threshold of 0.001, the *USDs* tensor identifies 41 of the 119 anomalies, while falsely classifying 128 out of 125,285 links. With the same p-value threshold, the *USDHDs* tensor can detect 108 of the 138 red team links while falsely classifying 3,483 out of nearly 3.5 million links. The insight that lower-dimensional tensors yield better performance, when evaluated in terms of false positives, leads us to develop our anomalous entity detection method to reduce the workload for analysts. We discuss the entity detection in Section 6.2.

While figures 7a and 7b show the scores when predicting *links* and not accounting for the skipped instances, Figures 8a and 8b display the scores for the same tensors on *event* prediction with skipped instances. Here we notice a drop in performance since we penalize the model for each of the skipped anomalies (i.e. the anomalies with any categorical features that we encounter for the first time in the test set).

6.1.2 Link and Event Prediction with and without Skipped Instances: We report our results both when scoring each individual cyber event log (*event* prediction), and each unique tensor entry for the events (*link* prediction). Both of these evaluation methods provide different insights into our results. *Link* prediction allows us to directly compare the performances of tensors with different dimensions, and to the prior work that used matrices. It also allows us to



(a) ROC

(b) PR

Fig. 8. ROC (a) and PR (b) curves for the LANL authentication matrices and tensors with event scoring and accounting for skips.

conceptually understand the anomaly detection performance of each tensor individually. For example, if we detect an anomaly once, we do not need to reward the system for detecting the same connection multiple times. However, we also need to penalize the system if we miss each individual malicious communication. *Event* prediction provides insights into how well our method would perform in a production environment when blocking each individual malicious connection, since every undetected communication between the compromised host could correspond to loss of more information. Additionally, we consider the cases where it is impossible for the system to provide information for an entity, and thus skips scoring an event. This occurs when we encounter an entity during the test period for the first time, causing it to fail to map to an index.

Table 6 shows the number of anomalous and benign skipped links and the total number of occurrences (event count) in the test set for each tensor. Table 7 gives a comprehensive summary of each tensors' performance, and the matching metric is compared to prior research in Table 5.

The first point to note in Table 6 is that although the number of skipped links differ for the authentication tensors *USDs*, *USDHs*, and *USDHDs*, the number of occurrences of these links remain the same. This happens because categorical mapping was used in the first three dimensions, and the difference between these tensors is the temporal dimensions. Therefore, we observe additional unique links as we increase the number of dimensions and the event count remains the same. The *Credit Card* tensor contains no skip counts because only the transaction type dimension was mapped categorically, and the training set contains all possible transaction types. All numerical dimensions across all datasets we considered map to a valid index because they were binned. *Neris 5 Character IP Hash Mapping* tensor missed nearly 36 thousand botnet connections due to failed mapping. Meanwhile, *Neris 6 Character IP Hash Mapping* skipped 49 thousand events; therefore, our event scoring that accounts for the skipped instances will penalize this mapping scheme significantly more. Note that the number of skipped events can be greater for some mappings, even when the number of skipped links is lower. This occurs due to the difference in the total number of links. Finally, *Neris 4 Character IP Hash Mapping* contains no skips during the test period. This likely occurs because 4 characters extracted from the hash of the IP address cover all possible IP addresses due to the increased collision rate (i.e. all addresses map to an index).

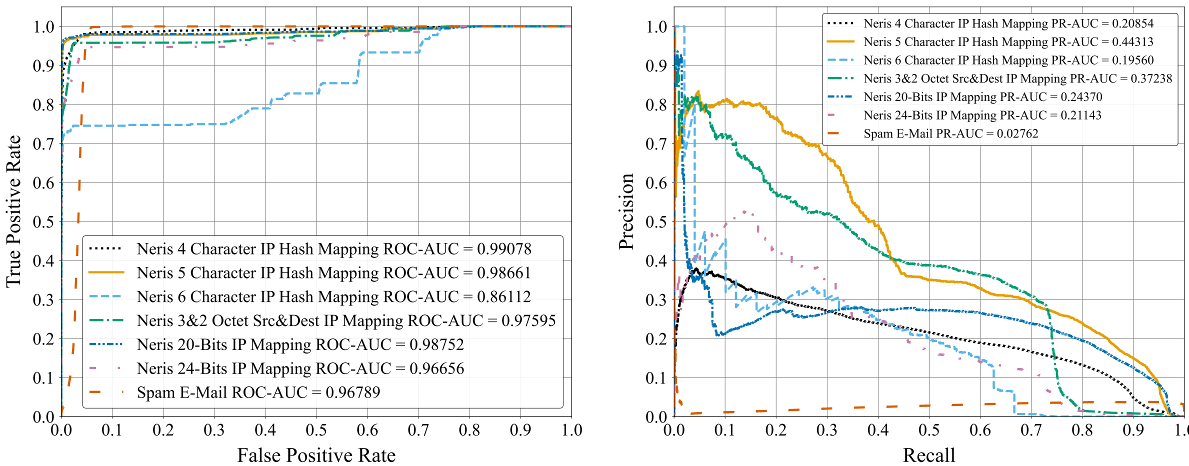
Table 6. Tensor link and event skip counts for anomalies and benign instances

	Anomaly Skips		Benign Skips	
Dataset & Tensor	Skipped Links	Total Num. of Occurrences (Event)	Skipped Links	Total Num. of Occurrences (Event)
LANL US	8	13	2,692	985,364
LANL UD	13	55	2,264	269,362
LANL USDs	13	15	8,428	990,531
LANL USDHs	13	15	38,205	990,531
LANL USDHDs	13	15	91,118	990,531
UGR'16 Neris 3&2 Octet Src&Dest IP Mapping	450	32,554	489,209	59,344,373
UGR'16 Neris 20-Bits IP Mapping	200	13,000	16,055	3,372,841
UGR'16 Neris 24-Bits IP Mapping	212	38,212	211,344	32,669,071
UGR'16 Neris 4 Character IP Hash Mapping	0	0	0	0
UGR'16 Neris 5 Character IP Hash Mapping	200	35,600	31,148	6,185,934
UGR'16 Neris 6 Character IP Hash Mapping	166	49,030	469,466	69,831,846
UGR'16 Spam E-Mail	65	763	16,061	116,535
PaySim Credit Card	0	0	0	0

Table 7. Tensor anomaly detection performance for *event* and *link* prediction with and without skipped occurrences

	With Skips				Without Skips			
	Event Score		*Link Score*		*Event Score*		*Link Score*	
Dataset & Tensor	ROC-AUC	PR-AUC	ROC-AUC	PR-AUC	ROC-AUC	PR-AUC	ROC-AUC	PR-AUC
LANL US	.8361	.0802	.8618	.1473	.8946	.0859	.9439	.1625
LANL UD	.8267	.0108	.7487	.0079	.8264	.0108	.8246	.0085
LANL USDs	.9152	.0448	.8851	.2127	.9945	.0487	.9769	.2359
LANL USDHs	.9191	.0017	.9104	.1019	.9988	.0019	.9948	.1115
LANL USDHDs	.9196	.0017	.9141	.0705	.9994	.0019	.9990	.0772
UGR'16 Neris 3&2 Octet Src&Dest IP Mapping	.3975	.2109	.8507	.3238	.9959	.6003	.9759	.3723
UGR'16 Neris 20-Bits IP Mapping	.7404	.1312	.9575	.2363	.9978	.1771	.9875	.2437
UGR'16 Neris 24-Bits IP Mapping	.2677	.0483	.8878	.1941	.9932	.2023	.9665	.2114
UGR'16 Neris 4 Character IP Hash Mapping	.9981	.0998	.9907	.2085	.9981	.0999	.9907	.2085
UGR'16 Neris 5 Character IP Hash Mapping	.2946	.0919	.9536	.4283	.9950	.3161	.9866	.4431
UGR'16 Neris 6 Character IP Hash Mapping	.1060	.0033	.6502	.1465	.9458	.1405	.8611	.1955
UGR'16 Spam E-Mail	.9661	.9152	.9640	.0275	.9661	.9152	.9678	.0276
PaySim Credit Card	.8147	.6987	.9025	.9742	.8147	.6987	.9025	.9742

The *Credit Card* tensor achieves a PR-AUC score of 0.9742 with unsupervised learning, as shown in Table 7, performing nearly as well as the supervised methods that see the labels [29]. We also compare the *Credit Card* tensor-based statistical anomaly detection to prior work in Table 5. Du et al. use a variant of Support Vector Machines with LogDet term (L-SVM) to perform supervised classification of the credit card fraud on the *PaySim* dataset [24]. Similarly, Xenopoulos takes a supervised approach using an ensemble of Deep Belief Networks (DBNs) [63]. For comparison, we use our score that accounts for events and skipped links with the ROC-AUC score of 0.815. Both of the supervised methods introduced by Du et al. and Xenopoulos yield better performance with ROC-AUC scores of 0.978 and 0.961 respectively. However, their scores are reported on manually balanced datasets (i.e. the number of instances in each class during testing are made equivalent). This adjustment gives a clear advantage to their results over our scoring with unbalanced data. In addition, tensor factorization extracts the latent patterns representing the normal or expected behavior in an unsupervised manner. Unsupervised methods have the advantage of yielding better results against novel attacks since the goal is to identify the abnormal phenomena. Because the trained model represents normal activities, anything that deviates from the norm, including novel attacks, can be detected. Another benefit of unsupervised methods is that they do not need labels during training. In comparison, obtaining fully labeled datasets to train supervised models is often expensive, and supervised methods



(a) ROC (b) PR

Fig. 9. ROC (a) and PR (b) curves for the Neris botnet detection tensors with different mapping schemes, and for the spam e-mail detection tensor. ROC curves show the ability of different mapping schemes to assign lower probability to botnet and spam e-mail links. PR curves show the false positive sensitivity of each tensor with different IP address mapping methods.

often do not perform well against unseen data, and struggle to perform on the highly unbalanced datasets common in anomaly detection tasks.

Figure 9a shows that *Neris 4 Character IP Hash Mapping* performs the best when assigning botnet activities lower p-values. Table 1 also shows that *Neris 4 Character IP Hash Mapping* has a lower standard deviation for the anomalous p-values, which indicates that the predictions yield a more precise decision boundary. On the other hand, Figure 9b shows *Neris 5 Character IP Hash Mapping* returns lower false-positive rates with a PR-AUC of 0.4431, if we score it over the links without accounting for the skips. However, accounting for the skips and scoring each event drops this score to .0919. Therefore, *Neris 4 Character IP Hash Mapping* is the most stable botnet detection tensor, as all the IP addresses find mapping in the dimensions due to the reduced mapping space capturing all possible outcomes during the training time. As the mapping space grows, the chance of collisions decreases; therefore, the number of skipped instances and events increases. This results in reduced performance when scoring the *events* and accounting for the skipped instances as shown in Table 7. *Neris 6 Character IP Hash Mapping* yields the worst performance when scoring the events since this tensor contains the highest number of skips.

Also from the UGR’16 dataset, the *Spam E-Mail* tensor has a balanced number of malicious and benign events, as shown in Table 3. Therefore, *event* scoring with the skips yields PR-AUC of 0.9152. The high PR-AUC score for this tensor indicates that we are able to classify the events with a low number of false positives, and that our method works on balanced data. We compare SPAM e-mail and botnet detection performance to prior work in Table 5.

Camacho et al. used a semi-supervised variant of PCA to detect both botnet and SPAM e-mail activities [15]. They form a derived dataset using the Feature as a Counter (FaaC) method [9]. FaaC forms a features vector of counts by aggregating the flows in 1-minute intervals. Each 1-minute interval produces a vector-sized 138 from 11 network related features. They train on half of the 12,000 observations extracted from the *TEST* portion of the UGR 16’ dataset (the 5 weeks used in testing in our paper). The other half was used in parameter optimization. Magán-Carrión et al. also use FaaC with a 1-minute window to form their features vectors [42]. Magán-Carrión et al. use various supervised learning methods on this dataset, here we report their best performing model (Random

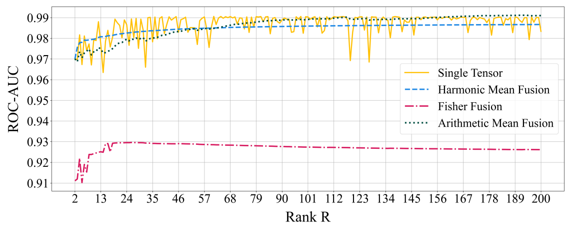


Fig. 10. *USDs*: ROC curves for p-value fusion methods on ensemble of tensors until rank R.

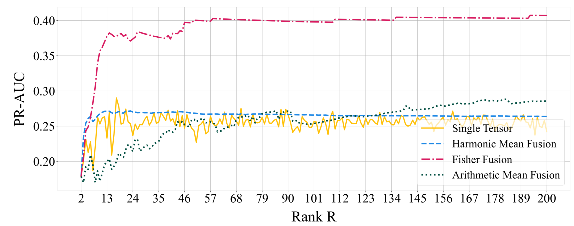


Fig. 11. *USDs*: PR curves for p-value fusion methods on ensemble of tensors until rank R.

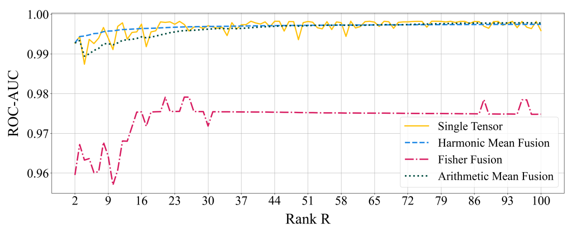


Fig. 12. *USDHs*: ROC curves for p-value fusion methods on ensemble of tensors until rank R.

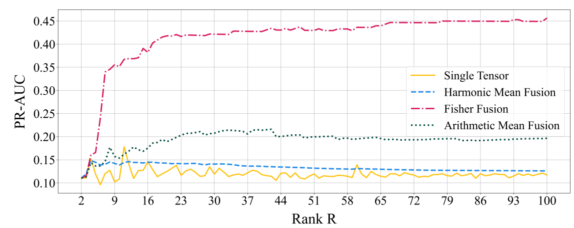


Fig. 13. *USDHs*: PR curves for p-value fusion methods on ensemble of tensors until rank R.

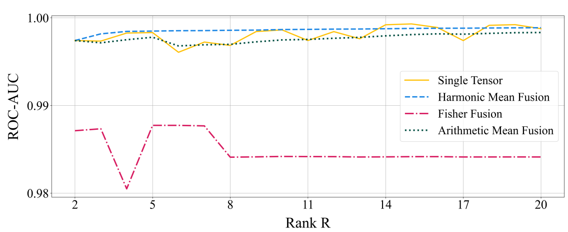


Fig. 14. *USDHDs*: ROC curves for p-value fusion methods on ensemble of tensors until rank R.

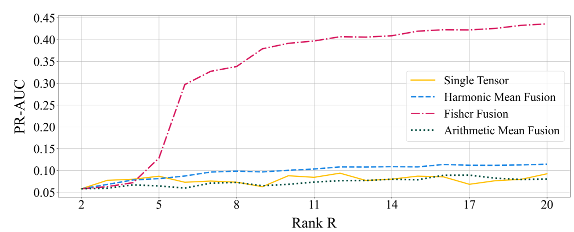


Fig. 15. *USDHDs*: PR curves for p-value fusion methods on ensemble of tensors until rank R.

Forest). Since Magán-Carrión et al. analyzes the UGR'16 using supervised methods, they also limit their analysis to the *TEST* portion of the dataset which includes the majority of the target malicious activities. In addition, they utilize Least Absolute Shrinkage and Selection Operator in feature selection and apply Bayesian optimization to identify the optimal hyperparameters. FaaC that is used in both of these prior works generates a number of data instances that are lower than the original number of flows in the dataset, with the greater number of features [42]. Magán-Carrión et al. state that when a time interval, or the new data instance, contains flows from multiple classes, the instance is labeled the class with the majority flow count within the time interval. Finally, Nguyen et al. apply an unsupervised technique, Variational Autoencoder, to detect botnet and SPAM emails [44]. Nguyen et al. also report results on Gaussian Based Thresholding, which performs better than their method when detecting SPAM e-mails; therefore, we also include it in our comparison.

They use a total of 5 days from the dataset with a 3-minute sliding window in their analysis. It is not clear if all prior research that we compare our results against considered the scoring of skipped instances during the test period. Therefore, we identify that tensors with event scoring including the skipped instances are the best choice for impartial comparison to each prior research introduced above. We also note that, in addition to ROC-AUC scores, Magán-Carrión et al. report their performance using the F1 metric, showing good performance [42]; however, they

Table 8. P-value fusion scores for botnet and spam e-mail detection tensors on ensemble of R tensors

		Arithmetic Mean		**Harmonic Mean**		**Fisher**	
Dataset & Tensor	Rank R	ROC-AUC	PR-AUC	ROC-AUC	PR-AUC	ROC-AUC	PR-AUC
UGR'16 Neris 3&2 Octet Src&Dest IP Mapping	8	.9780	.4064	.9780	.4728	.9778	.4563
UGR'16 Neris 20-Bits IP Mapping	10	.9661	.2397	.9671	.2459	.9885	.3003
UGR'16 Neris 24-Bits IP Mapping	7	.8675	.2139	.9961	.2247	.9244	.2283
UGR'16 Neris 4 Character IP Hash Mapping	10	.9914	.2290	.9914	.2248	.9894	.2482
UGR'16 Neris 5 Character IP Hash Mapping	7	.9868	.5087	.9874	.4751	.6316	.4952
UGR'16 Neris 6 Character IP Hash Mapping	6	.8601	.1649	.8624	.2070	.8644	.1997
UGR'16 Spam E-Mail	20	.9781	.0430	.9699	.0300	.8456	.0474

compare their results to Camacho et al. [15] using the ROC-AUC scores. Therefore, we perform the comparison using ROC-AUC scores in this paper.

Our multi-dimensional analysis method yields better performance in most of the cases as shown in Table 5 where the winning score is highlighted. Gaussian Based Thresholding performs slightly better than our tensor-based method with a ROC-AUC score of 0.970 versus 0.966, respectively. With the caution in the false-positive realm, we note that it is also interesting to see the latent patterns extracted with tensor factorization can yield better results when assigning lower p-values to anomalies (high ROC-AUC score) compared to both supervised and semi-supervised models. These results are accomplished using the same detection methodology across all datasets, showing that our method is highly effective and general across applications in cyber security.

6.1.3 Ensemble of Tensors with Different Ranks: Anomalies are rare, but malicious events. Because they are rare, datasets for anomaly detection suffer from high class imbalance problems. This can be seen in tables 2 and 3 with event counts in the test set for each tensor, and in Table 1 when looking at the counts for anomaly and benign links (with the exception of *Credit Card* tensor, where the number of benign links is less than the anomalous ones). Class imbalance problems result in high sensitivity due to false positives. False positives are especially problematic for the incident responders; therefore, one of the focuses of this expansion paper is to reduce the false-positive rate of our multi-dimensional anomaly detection methods. To this extent, we show that the knowledge extracted from tensor decomposition with different ranks can be combined to achieve a better prediction.

We unify the predictions made from ensemble of tensors using p-value fusion methods (*Harmonic mean*, *Fisher*, and *Arithmetic mean*). The improvement in anomaly detection, especially a drop in false positive rate, is demonstrated with the increasing prediction scores as we add more tensors to the ensemble and comparing the results to the predictions made from a single tensor. For instance, we get the ROC-AUC and PR-AUC scores from fusing p-values for each link **i** extracted from tensors rank 2 through 100, and compare these scores to a tensor decomposed with rank 100 alone. We first look at the LANL authentication tensors *USDs*, *USDHs*, and *USDHDs*. Figures 10, 12, and 14 display the ROC-AUC scores, and figures 11, 13, and 15 provide the PR-AUC values.

As we add more tensors to the group, *Harmonic Mean* continues to closely follow the score from the tensor rank *R* alone, staying slightly above or below the AUC value of a single tensor for both ROC and PR curves. Therefore, *Harmonic Mean* can be used as a fusion method to smooth out the final results. We observe this for each of the authentication tensors. We also see that *Fisher* fusion results in a lower ROC-AUC score; however, it significantly reduces the false-positive rates. With an ensemble of 200 tensors, the PR-AUC score for *USDs* increases above .41, from around .25 obtained when scoring a single rank 200 tensor. Similarly, an ensemble of *USDHDs* tensors ranks 2 through 20 returns a PR-AUC score of around .45, which is a significant increase from around .10 of a single tensor with rank 20.

Another observation to be made from these plots is how many ensembles are needed to reach a better performance for different fusion techniques. For instance, *Arithmetic mean* fusion needs a multitude of tensors to yield improved performances. In comparison to the other fusion methods, *Fisher* requires fewer tensors in the ensemble to perform better than a single tensor decomposition. We see a similar improvement in performance when ensemble learning is applied to the UGR'16 dataset.

Table 8 shows the ROC-AUC and PR-AUC scores obtained with fusing the p-values using the ensemble of tensors rank 2 through R, where R was again chosen based on the GPU memory space. We again see the increase in PR-AUC scores for each tensor. For instance, *Neris 5 Character IP Hash Mapping* increases the score up to PR-AUC of .5087. Similarly, *Neris 20-Bits IP Mapping* yield .3003 PR-AUC. In addition to applying p-value fusion techniques on tensors with different ranks, we also perform p-value fusion over the tensor dimensions for entity detection.

6.2 Anomalous Entity Detection

Table 9. LANL authentication tensor entity counts for fusion dimension(s)

Target Dimensions	Total Entities	Red Team Entities
User	10,129	76
Source	12,519	1
Destination	4,167	93
User-Source	31,287	76
User-Destination	69,045	117
Source-Destination	70,533	93

Table 10. Neris botnet and spam e-mail detection tensors entity counts for fusion dimension(s)

	Source		Destination		Source-Destination	
Dataset & Tensor	Total Entities	Anomalous Entities	Total Entities	Anomalous Entities	Total Entities	Anomalous Entities
UGR'16 Neris 3&2 Octet Src&Dest IP Mapping	1,913,701	40	65,532	61	5,556,593	393
UGR'16 Neris 20-Bits IP Mapping	652,285	70	374,830	51	7,347,752	776
UGR'16 Neris 24-Bits IP Mapping	1,693,756	64	410,362	31	7,050,719	364
UGR'16 Neris 4 Character IP Hash Mapping	65,536	70	65,530	61	4,854,559	976
UGR'16 Neris 5 Character IP Hash Mapping	1,010,027	70	408,404	48	7,583,050	716
UGR'16 Neris 6 Character IP Hash Mapping	1,922,410	53	399,520	23	5,791,258	173
UGR'16 Spam E-Mail	33,453	441	65,536	15	1,876,160	1,140

We detect anomalous entities by fusing p-values over tensor dimensions to assign anomaly scores to one or more target dimension(s). In our results, an entity can be a single physical object in the network, such as a *User*, or a combination of physical objects, such as a *User-Source*. Table 9 shows the total number of entities for the LANL authentication dataset, and Table 10 has the entity counts for the tensors created from the UGR'16 dataset. Fusing down to more than one dimension allows us to compare our results directly to previous performance benchmarks with matrix factorization.

Figure 16 shows ROC-AUC and PR-AUC scores for each tensor when detecting anomalous entities from the authentication events via score fusion. ROC-AUC scores indicate an improved ability to capture anomalous entities with the introduction of time-based dimensions. PR-AUC scores reveal an increase in false positives with increasing dimension; we suspect this increase is due to added sensitivity to temporal user characteristics. We obtain increased ROC-AUC and PR-AUC scores by scoring ranked lists across all tensors via MRR, demonstrating that each tensor captures complementary anomalies.

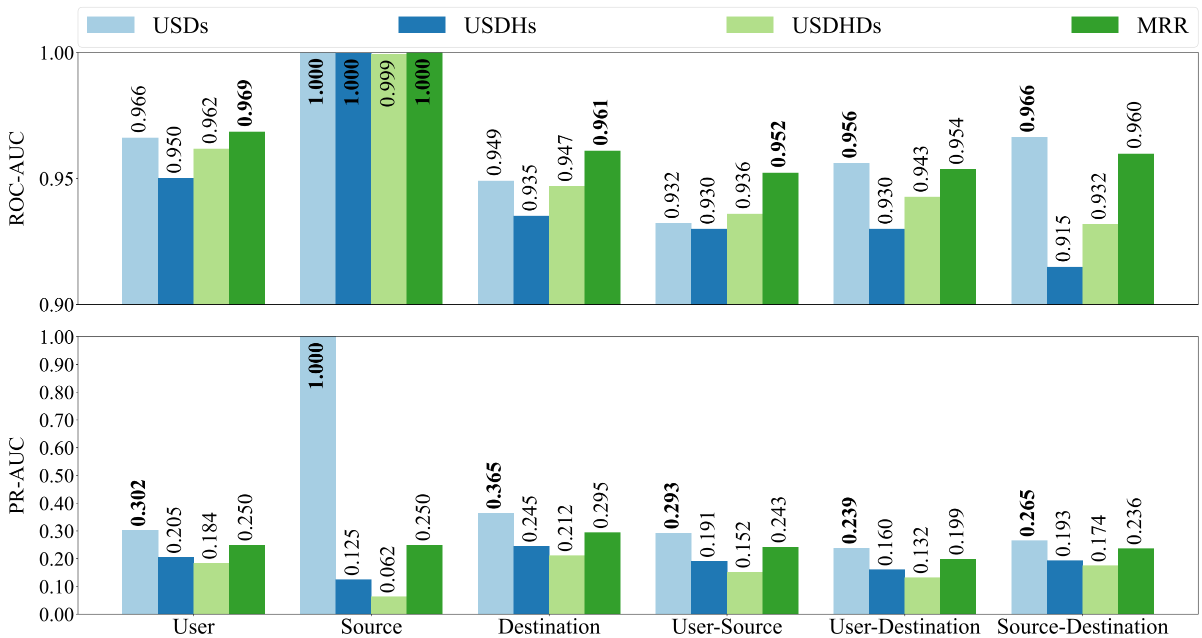


Fig. 16. ROC-AUC and AP scores for anomalous entity detection using tensors with varying dimension. "MRR" shows fusion ranked lists produced by all other tensors.

The previous benchmark for red team detection on the LANL Unified Host and Network Dataset was established by Sanna Passino et al. with AUC scores of 0.863 and 0.902 when detecting red team events[9] over the matrix with dimensions *User - Source* and *User - Destination*, respectively [46]. Our fusion AUC scores, at 0.952 for *User - Source* and 0.954 for *User - Destination*, demonstrating that jointly learning user behavior patterns over multiple dimensions significantly enhances anomaly detection performance. We also look at the anomalous entity detection performance for botnet events and spam e-mails.

Figure 17 shows the fusion scores for *Source*, *Destination*, and *Source-Destination* dimensions for each of the tensors. Here *Source* and *Destination* refers to the dimensions for the source and destination IP addresses. *Neris 5 Character IP Hash Mapping* performs the best when detecting anomalous entities in terms of lower false positives. Additionally, using tensors, we can reach a ROC-AUC score of .946 when detecting anomalous spam e-mails between the source and destination IP addresses. These results indicate that our methods can be used in various types of applications, making it like a "swiss army knife" tool for anomaly detection.

7 PYTHON CP-APR

The CP-APR algorithm was originally written in MATLAB [27] and released in the Tensor Toolbox [8]. The MATLAB Tensor Toolbox is a popular software for tensor factorization, but MATLAB runs slowly when analyzing large datasets. To support reproducibility and provide an easy path to operationalization, we implemented a Python version of the CP-APR algorithm, named pyCP_APR, and released the code on GitHub. The software package supports three objectives: providing an easy to use API, reducing the training time by supporting GPU computation, and introducing an interface that easily supports anomaly detection workflows. pyCP_APR comes with *Numpy* and

[9] We compare to the PMF model variant that does not have access to covariate information.

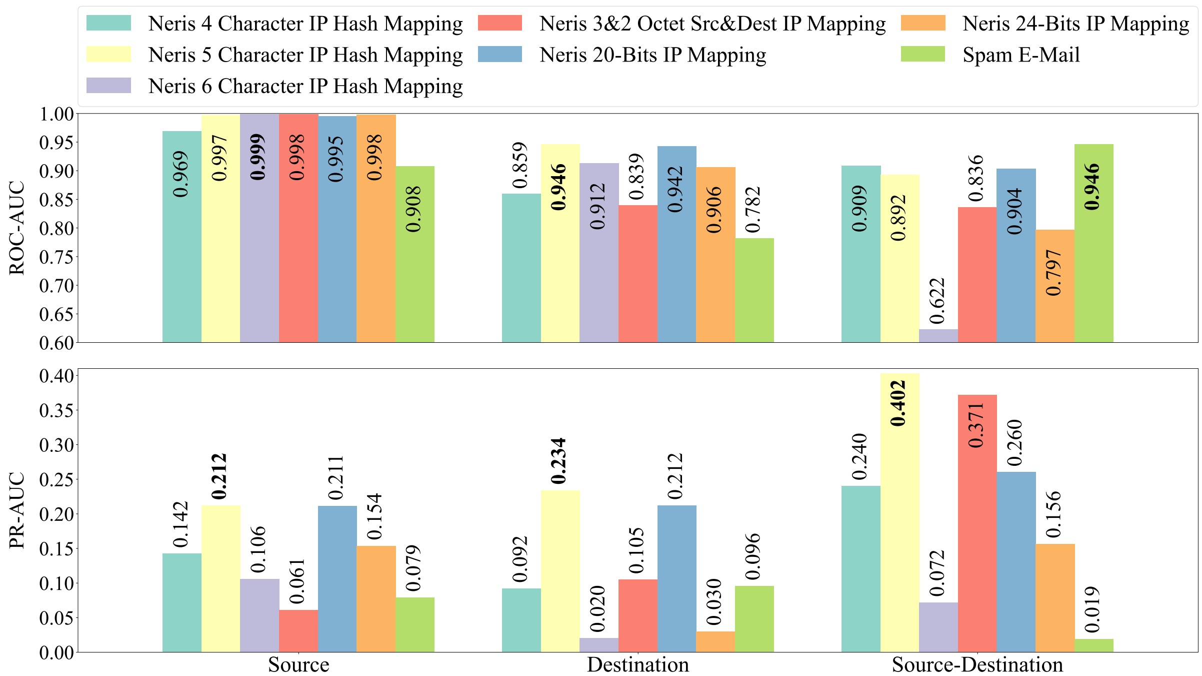


Fig. 17. ROC-AUC and AP scores for anomalous entity detection using tensors with varying dimension.

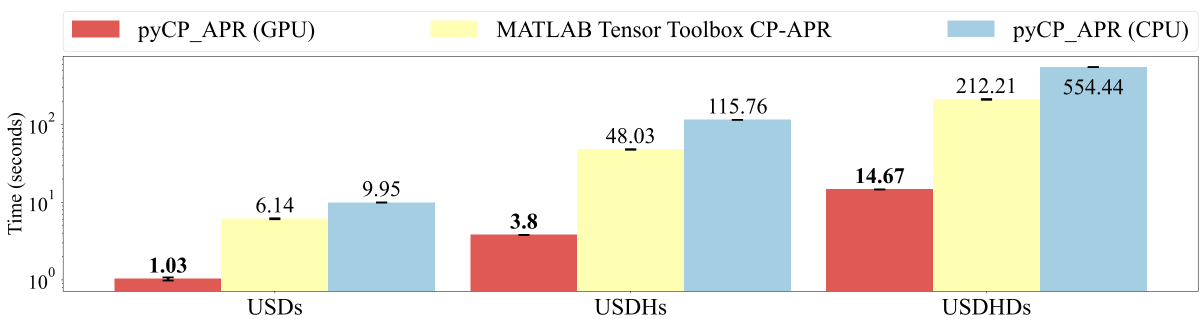


Fig. 18. Comparision of runtimes for USDs, USDHs, and USDHDs tensors on PyTorch, Numpy, and MATLAB CP-APR. Results are presented for the average of 10 runs. Each run is for a maximum of 10 epochs with rank 10.

PyTorch back-end options, and provides a simple API, similar to Scikit-learn. It also includes an interface to predict anomalies over the fitted tensor, using the new methods presented in this paper. The *PyTorch* back-end allows utilization of a GPU to reduce the runtime when factorizing sparse tensors. Since the number of cyber event logs regularly reaches into the millions or billions, the fast training and inference capabilities available in pyCP_APR are critical.

In Figure 18, we compare the runtimes of pyCP_APR with the PyTorch backend on a TITAN RTX GPU, the Numpy backend on an Intel Skylake CPU, and the CP-APR MATLAB implementation run on an Intel Skylake CPU.

The average of 10 runs with 95% confidence interval (CI) is reported for factorizing the LANL authentication tensor, using a rank 10 with a maximum of 10 iterations[10]. While MATLAB CP-APR performs better than Numpy, we can see a significant improvement in runtime using a GPU. For instance, PyTorch back-end using GPU is approximately 15 times faster when taking the decomposition of the 6 dimensional tensor *USDHDs*.

pyCP_APR is available to download from our repository; https://github.com/lanl/pyCP_APR. The repository includes example Jupyter Notebooks and the API documentation.

8 LESSONS LEARNED AND FUTURE WORK

We noticed that anomaly detection results vary slightly depending on the initialization of the tensor factorization algorithm. Throughout this paper, we randomly initialize the latent factors with values between 0 and 1 drawn from a uniform distribution. Future work can consider better initialization techniques such as using the latent factors extracted from another CP tensor decomposition algorithm. In such a method, if the algorithm used is not non-negative, the negative values in the latent factors could be replaced with an epsilon value.

We also note that the size of the tensor on the GPU increases as we increase the rank. The large tensors built from cyber data push the limits on GPU memory space. To better handle the tensor size, and potentially to remove noise from the data, future work will consider building the tensor by aggregating the events over a time interval, similar to the pre-processing steps performed by the prior work [15, 42, 44], and by utilizing feature transformation and discretization methods discussed by prior work [16].

Extensions to our model include augmenting it to handle "cold starts" by incorporating Sanna Passino's covariate regression model [46]. Also, it should be noted that our model is fully compatible with a Bayesian extension, which would elegantly perform model averaging, similar to our smoothing and rank-fusion steps [53]. We leave both of these extensions to future work.

Finally, to further mitigate the possible false positives from our method, in production environments our framework can be integrated with existing rule-based and statistical intrusion detection systems where post-processing can ease the workload on analysts. For instance, the devices from an anomalous authentication alert can be correlated with other weak indicators, such as anomalous process start events [57], and netflow traffic logs such as website visits.

9 CONCLUSION

In this paper, we introduced a generalized version of our tensor decomposition method for unsupervised anomaly detection. With an unsupervised approach, our solution matches or surpasses state-of-the-art supervised and semi-supervised learning baselines across several challenging and diverse cyber application areas with the added potential benefit of better generalizability to unseen types of attacks by detecting deviations from normal or expected behavior. Our tensor analysis based method is sensitive to anomalous activity over a diverse set of attributes. We show that higher-order representations enhance the detection of anomalies due to the ability of tensor factorization techniques to extract more predictive activity profiles that describe events simultaneously over multiple dimensions. We also showed that the Python library that we have developed, named pyCP_APR, can significantly reduce the tensor factorization time using a GPU. pyCP_APR also adds operational value to the CP-APR algorithm by combining the factorization with an anomaly scoring and prediction interface.

Combining information across multiple tensor dimensions demonstrates state-of-the-art results for red team detection on the LANL authentication dataset. Our methods generalize to the botnet, spam e-mail, and credit card fraud detection problems, showing the multipurpose application of the system. The datasets used in our analysis include both synthetic and real activities, balanced and unbalanced classes, and data collected on different resources (host activity and netflow events). The diverse set of data used in this paper further supports the generalizability of our methodology. The performance is reported accounting for both the events and links in the dataset to give a

[10]The variance across runs is practically negligible, as shown by the small confidence intervals.

comprehensive view of the results. We also show that p-value fusion methods using an ensemble of tensor ranks yield lower false-positive rates that are essential for incident responders.

ACKNOWLEDGMENTS

We thank Francesco Sanna Passino and Melissa Turcotte for providing valuable feedback and shared attribute mappings [46]. We also thank Neale Pickett for providing helpful feedback on the IP address hashing schemes, and Austin Thresher for the feedback on our software design. Finally, we thank the anonymous reviewers for their valuable suggestions. The research presented in this paper was supported by the Information Science and Technology Institute at Los Alamos National Laboratory (LANL) through its Cyber Research school, by the Laboratory Directed Research and Development program of LANL under project numbers 20190020DR and 20210043DR, and by the LANL Institutional Computing Program project number 89233218CNA000001. LANL is operated by Triad National Security, LLC, for the National Nuclear Security Administration of the U.S. Department of Energy (Contract No. 89233218CNA000001).

REFERENCES

[1] 2019. *Cost of a Data Breach Report*. Technical Report. IBM. 23 pages. https://www.accenture.com/_acnmedia/PDF-96/Accenture-2019-Cost-of-Cybercrime-Study-Final.pdf

[2] 2019. *Insider Threat Report*. Technical Report. Verizon. 71 pages. https://enterprise.verizon.com/resources/reports/insider-threat-report/

[3] 2020. *Cyber Espionage Report*. Technical Report. Verizon. 60 pages. https://www.verizon.com/business/resources/reports/cyber-espionage-report/

[4] 2020. *Data Breach Investigations Report 2020*. Technical Report. Verizon. 119 pages. https://enterprise.verizon.com/resources/reports/dbir/

[5] 2020. *Mandiant Security Effectiveness Report*. Technical Report. FireEye. 23 pages. https://www.accenture.com/_acnmedia/PDF-96/Accenture-2019-Cost-of-Cybercrime-Study-Final.pdf

[6] 2020. *State of Malware Report*. Technical Report. Malwarebytes Labs. 57 pages. https://www.accenture.com/_acnmedia/PDF-96/Accenture-2019-Cost-of-Cybercrime-Study-Final.pdf

[7] Edward G Allan, Michael R Horvath, Christopher V Kopek, Brian T Lamb, Thomas S Whaples, and Michael W Berry. 2008. Anomaly detection using nonnegative matrix factorization. In *Survey of Text Mining II*. Springer, 203–217.

[8] Brett W. Bader, Tamara G. Kolda, et al. 2017. MATLAB Tensor Toolbox Version 3.0-dev. Available online. https://gitlab.com/tensors/tensor_toolbox

[9] James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *Journal of machine learning research* 13, 2 (2012).

[10] Manish Bhattarai, Gopinath Chennupati, Erik Skau, Raviteja Vangara, Hristo Djidjev, and Boian S Alexandrov. 2020. Distributed Non-Negative Tensor Train Decomposition. In *2020 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE, 1–10.

[11] Manish Bhattarai, Ben Nebgen, Erik Skau, Maksim Eren, Gopinath Chennupati, Raviteja Vangara, Hristo Djidjev, John Patchett, Jim Ahrens, and Boian Alexandrov. 2021. pyDNMFk: Python Distributed Non Negative Matrix Factorization. https://github.com/lanl/pyDNMFk. https://doi.org/10.5281/zenodo.4722448

[12] K Bissell, R LaSalle, and P.D. Cin. 2020. *Innovate for Cyber Resilience*. Technical Report. Accenture, Ponemon Institute.

[13] K. Bissell and L. Ponemon. 2019. *The Cost of Cybercrime*. Technical Report. Accenture, Ponemon Institute. https://www.accenture.com/_acnmedia/PDF-96/Accenture-2019-Cost-of-Cybercrime-Study-Final.pdf

[14] D. Bruns-Smith, M. M. Baskaran, J. Ezick, T. Henretty, and R. Lethin. 2016. Cyber Security through Multidimensional Data Decompositions. In *2016 Cybersecurity Symposium (CYBERSEC)*. 59–67. https://doi.org/10.1109/CYBERSEC.2016.017

[15] José Camacho, Gabriel Maciá-Fernández, Noemí Marta Fuentes-García, and Edoardo Saccenti. 2019. Semi-Supervised Multivariate Statistical Network Monitoring for Learning Security Threats. *IEEE Transactions on Information Forensics and Security* 14, 8 (2019), 2179–2189. https://doi.org/10.1109/TIFS.2019.2894358

[16] Salvatore M. Carta, Alessandro Sebastian Podda, Diego Reforgiato Recupero, and Roberto Saia. 2020. A Local Feature Engineering Strategy to Improve Network Anomaly Detection. *Future Internet* 12 (2020), 177.

[17] Laetitia Chapel and Chloé Friguet. 2014. Anomaly Detection with Score Functions Based on the Reconstruction Error of the Kernel PCA. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2014, Nancy, France, September 15-19, 2014. Proceedings, Part I (Lecture Notes in Computer Science, Vol. 8724)*. Springer, 227–241. https://doi.org/10.1007/978-3-662-44848-9_15

[18] J. Chen, Saket K. Sathe, Charu C. Aggarwal, and Deepak S. Turaga. 2017. Outlier Detection with Autoencoder Ensembles. In *SDM*.

[19] Gopinath Chennupati, Raviteja Vangara, Erik Skau, Hristo Djidjev, and Boian Alexandrov. 2020. Distributed non-negative matrix factorization with determination of the number of latent features. *The Journal of Supercomputing* (2020), 1–31.

[20] Eric C. Chi and Tamara G. Kolda. 2012. On Tensors, Sparsity, and Nonnegative Factorizations. *SIAM J. Matrix Anal. Appl.* 33, 4 (2012), 1272–1299. https://doi.org/10.1137/110859063

[21] John M. Conroy. 2018. *Classification of Red Team Authentication Events in an Enterprise Network*. Chapter Chapter 9, 179–194. https://doi.org/10.1142/9781786345646_009 arXiv:https://www.worldscientific.com/doi/pdf/10.1142/9781786345646_009

[22] Gordon V. Cormack, Charles L. A. Clarke, and Stefan Büttcher. 2009. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2009, Boston, MA, USA, July 19-23, 2009*. ACM, 758–759. https://doi.org/10.1145/1571941.1572114

[23] Kaize Ding, Jundong Li, Rohit Bhanushali, and Huan Liu. 2019. Deep Anomaly Detection on Attributed Networks. In *SDM*.

[24] Jia-Zhi DU, Wei-Gang LU, Xiao-He WU, Jun-Yu DONG, and Wang-Meng ZUO. 2018. L-SVM: A radius-margin-based SVM algorithm with LogDet regularization. *Expert Systems with Applications* 102 (2018), 113–125. https://doi.org/10.1016/j.eswa.2018.02.006

[25] Daniel M. Dunlavy, Tamara G. Kolda, and Evrim Acar. 2011. Temporal Link Prediction Using Matrix and Tensor Factorizations. *ArXiv* abs/1005.4006 (2011).

[26] M. E. Eren, J. S. Moore, and B. S. Alexandrov. 2020. Multi-Dimensional Anomalous Entity Detection via Poisson Tensor Factorization. In *2020 IEEE International Conference on Intelligence and Security Informatics (ISI)*. 1–6. https://doi.org/10.1109/ISI49825.2020.9280524

[27] Samantha Hansen, Todd Plantenga, and Tamara G. Kolda. 2015. Newton-Based Optimization for Kullback-Leibler Nonnegative Tensor Factorization. *Optimization Methods and Software* 30, 5 (April 2015), 1002–1029. https://doi.org/10.1080/10556788.2015.1009977

[28] Frank L. Hitchcock. 1927. The Expression of a Tensor or a Polyadic as a Sum of Products. *Journal of Mathematics and Physics* 6, 1-4 (1927), 164–189. https://doi.org/10.1002/sapm192761164 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/sapm192761164

[29] Arjun Joshua. 2018. Predicting Fraud in Financial Payment Services. https://www.kaggle.com/arjunjoshua/predicting-fraud-in-financial-payment-services.

[30] Hideaki Kanehara, Yuma Murakami, Jumpei Shimamura, Takeshi Takahashi, Daisuke Inoue, and Noboru Murata. 2019. Real-Time Botnet Detection Using Nonnegative Tucker Decomposition. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing* (Limassol, Cyprus) *(SAC '19)*. Association for Computing Machinery, New York, NY, USA, 1337–1344. https://doi.org/10.1145/3297280.3297415

[31] Kaspersky. 2020. *Machine Learning Methods for Malware Detection*. Technical Report.

[32] Fabian Keller, Emmanuel Müller, and Klemens Böhm. 2012. HiCS: High Contrast Subspaces for Density-Based Outlier Ranking. *2012 IEEE 28th International Conference on Data Engineering* (2012), 1037–1048.

[33] Tung Kieu, B. Yang, Chenjuan Guo, and Christian S. Jensen. 2019. Outlier Detection for Time Series with Recurrent Autoencoder Ensembles. In *IJCAI*.

[34] I. Kisil, A. Moniri, and D. P. Mandic. 2018. TENSOR ENSEMBLE LEARNING FOR MULTIDIMENSIONAL DATA. In *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. 1358–1362. https://doi.org/10.1109/GlobalSIP.2018.8646694

[35] Tamara G. Kolda and Brett W. Bader. 2009. Tensor Decompositions and Applications. *SIAM Rev.* 51, 3 (2009), 455–500. https://doi.org/10.1137/07070111X

[36] Aleksandar Lazarevic and Vipin Kumar. 2005. Feature Bagging for Outlier Detection. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining* (Chicago, Illinois, USA) *(KDD '05)*. Association for Computing Machinery, New York, NY, USA, 157–166. https://doi.org/10.1145/1081870.1081891

[37] Daniel D Lee and H Sebastian Seung. 1999. Learning the parts of objects by non-negative matrix factorization. *Nature* 401, 6755 (1999), 788–791.

[38] Lek-Heng Lim and Pierre Comon. 2009. Nonnegative approximations of nonnegative tensors. *Journal of Chemometrics: A Journal of the Chemometrics Society* 23, 7-8 (2009), 432–441.

[39] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2012. Isolation-Based Anomaly Detection. *ACM Trans. Knowl. Discov. Data* 6, 1, Article 3 (mar 2012), 39 pages. https://doi.org/10.1145/2133360.2133363

[40] E. A. Lopez-Rojas, Ahmad Elmir, and S. Axelsson. 2016. Paysim: a financial mobile money simulator for fraud detection.

[41] Gabriel Maciá-Fernández, José Camacho, Roberto Magán-Carrión, Pedro García-Teodoro, and Roberto Therón. 2018. UGR'16: A new dataset for the evaluation of cyclostationarity-based network IDSs. *Comput. Secur.* 73 (2018), 411–424. https://doi.org/10.1016/j.cose.2017.11.004

[42] Roberto Magán-Carrión, Daniel Urda, Ignacio Díaz-Cano, and Bernabé Dorronsoro. 2020. Towards a Reliable Comparison and Evaluation of Network Intrusion Detection Systems Based on Machine Learning Approaches. *Applied Sciences* 10, 5 (2020). https://doi.org/10.3390/app10051775

[43] Md. Ochiuddin Miah, Sakib Shahriar Khan, Swakkhar Shatabda, and Dewan Md. Farid. 2019. Improving Detection Accuracy for Imbalanced Network Intrusion Classification using Cluster-based Under-sampling with Random Forests. In *2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT)*. 1–5. https://doi.org/10.1109/ICASERT.2019.8934495

[44] Quoc Phong Nguyen, Kar Wai Lim, Dinil Mon Divakaran, Kian Hsiang Low, and Mun Choon Chan. 2019. Gee: A gradient-based explainable variational autoencoder for network anomaly detection. In *2019 IEEE Conference on Communications and Network Security*

(CNS). IEEE, 91–99.

[45] Guansong Pang, Chunhua Shen, Longbing Cao, and Anton van den Hengel. 2021. Deep Learning for Anomaly Detection. *ACM Computing Surveys (CSUR)* 54 (2021), 1 – 38.

[46] Francesco Sanna Passino, Melissa J. M. Turcotte, and Nicholas A. Heard. 2020. Graph link prediction in computer networks using Poisson matrix factorisation. *CoRR* abs/2001.09456 (2020). arXiv:2001.09456 https://arxiv.org/abs/2001.09456

[47] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.). Curran Associates, Inc., 8024–8035. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[48] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12, Oct (2011), 2825–2830.

[49] Rifki Indra Perwira, Yuli Fauziah, I Putu Retya Mahendra, Dessyanto Boedi Prasetyo, and Oliver Samuel Simanjuntak. 2019. Anomaly-based Intrusion Detection and Prevention Using Adaptive Boosting in Software-defined Network. In *2019 5th International Conference on Science in Information Technology (ICSITech)*. 188–192. https://doi.org/10.1109/ICSITech46713.2019.8987531

[50] Matthew Price-Williams, Melissa J. Turcotte, and Nick Heard. 2018. Time of Day Anomaly Detection. In *European Intelligence and Security Informatics Conference, EISIC 2018, Karlskrona, Sweden, October 24-25, 2018*. IEEE, 1–6. https://doi.org/10.1109/EISIC.2018.00009

[51] Yang Qi, Pierre Comon, and Lek-Heng Lim. 2016. Semialgebraic Geometry of Nonnegative Tensor Rank. *SIAM J. Matrix Anal. Appl.* 37, 4 (2016), 1556–1580. https://doi.org/10.1137/16M1063708

[52] Roberto Saia, Salvatore Carta, and Diego Reforgiato Recupero. 2018. A Probabilistic-driven Ensemble Approach to Perform Event Classification in Intrusion Detection System. 8. https://doi.org/10.5220/0006893801410148

[53] Aaron Schein, John Paisley, David M. Blei, and Hanna Wallach. 2015. Bayesian Poisson Tensor Factorization for Inferring Multilateral Relations from Sparse Dyadic Event Counts. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Sydney, NSW, Australia) *(KDD '15)*. Association for Computing Machinery, New York, NY, USA, 1045–1054. https://doi.org/10.1145/2783258.2783414

[54] Aaron Schein, Mingyuan Zhou, David M. Blei, and Hanna M. Wallach. 2016. Bayesian Poisson Tucker Decomposition for Learning the Structure of International Relations. In *ICML*.

[55] Duc P Truong, Erik Skau, Vladimir I Valtchinov, and Boian S Alexandrov. 2020. Determination of latent dimensionality in international trade flow. *Machine Learning: Science and Technology* 1, 4 (2020), 045017.

[56] Ledyard R Tucker. 1966. Some mathematical notes on three-mode factor analysis. *Psychometrika* 31, 3 (1966), 279–311.

[57] Melissa J. Turcotte, Juston Moore, Nick Heard, and Aaron McPhall. 2016. Poisson factorization for peer-based anomaly detection. In *IEEE Conference on Intelligence and Security Informatics, ISI 2016, Tucson, AZ, USA, September 28-30, 2016*. IEEE, 208–210. https://doi.org/10.1109/ISI.2016.7745472

[58] Melissa J. M. Turcotte, Alexander D. Kent, and Curtis Hash. 2018. *Unified Host and Network Data Set*. World Scientific, Chapter Chapter 1, 1–22. https://doi.org/10.1142/9781786345646_001 arXiv:https://www.worldscientific.com/doi/pdf/10.1142/9781786345646_001

[59] R. Vangara, E. Skau, G. Chennupati, H. Djidjev, T. Tierney, J. P. Smith, M. Bhattarai, V. G. Stanev, and B. S. Alexandrov. 2020. Semantic Nonnegative Matrix Factorization with Automatic Model Determination for Topic Modeling. In *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*. 328–335. https://doi.org/10.1109/ICMLA51294.2020.00060

[60] Parag Verma, Shayan Anwar, Shadab Khan, and Sunil B. Mane. 2018. Network Intrusion Detection Using Clustering and Gradient Boosting. *2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT)* (2018), 1–7.

[61] Maksims Volkovs, Guang Wei Yu, and Tomi Poutanen. 2017. DropoutNet: Addressing Cold Start in Recommender Systems. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*. 4957–4966. http://papers.nips.cc/paper/7081-dropoutnet-addressing-cold-start-in-recommender-systems

[62] Daniel Wilson. 2019. The harmonic mean p -value for combining dependent tests. *Proceedings of the National Academy of Sciences* 116 (01 2019), 201814092. https://doi.org/10.1073/pnas.1814092116

[63] Peter Xenopoulos. 2017. Introducing DeepBalance: Random deep belief network ensembles to address class imbalance. In *2017 IEEE International Conference on Big Data (Big Data)*. 3684–3689. https://doi.org/10.1109/BigData.2017.8258364

[64] Houssam Zenati, Manon Romain, Chuan-Sheng Foo, Bruno Lecouat, and Vijay Ramaseshan Chandrasekhar. 2018. Adversarially Learned Anomaly Detection. *2018 IEEE International Conference on Data Mining (ICDM)* (2018), 727–736.

[65] Qibin Zhao, Liqing Zhang, and Andrzej Cichocki. 2015. Bayesian CP Factorization of Incomplete Tensors with Automatic Rank Determination. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37, 9 (2015), 1751–1763. https://doi.org/10.1109/TPAMI.2015.2392756