Machine Learning (2020) 109:1127–1143
https://doi.org/10.1007/s10994-020-05870-y

Detecting anomalous packets in network transfers: investigations using PCA, autoencoder and isolation forest in TCP

Mariam Kiran[1] · Cong Wang[2] · George Papadimitriou[3] · Anirban Mandal[2] · Ewa Deelman[3]

Received: 2 February 2019 / Revised: 16 October 2019 / Accepted: 13 February 2020 /
Published online: 12 March 2020

Abstract
Large-scale scientific workflows rely heavily on high-performance file transfers. These transfers require strict quality parameters such as guaranteed bandwidth, no packet loss or data duplication. To have successful file transfers, methods such as predetermined thresholds and statistical analysis need to be done to determine abnormal patterns. Network administrators routinely monitor and analyze network data for diagnosing and alleviating these, making decisions based on their experience. However, as networks grow and become complex, monitoring large data files and quickly processing them, makes it improbable to identify errors and rectify these. Abnormal file transfers have been classified by simply setting alert thresholds, via tools such as PerfSonar and TCP statistics (Tstat). This paper investigates the feasibility of unsupervised feature extraction methods for identifying network anomaly patterns with three unsupervised classification methods—principal component analysis, autoencoder and isolation forest. We collect file transfer statistics from two experiment sets—synthetic iPerf generated traffic and 1000 Genome workflow runs, with synthetically introduced anomalies. Our results show that while PCA and a simple autoencoder finds it difficult to detect clusters, the tree-variant isolation forest is able to identify anomalous packets by breaking down TCP traces into tree classes early.

Keywords PCA · Autoencoders · Isolation forest · Network traffic

1 Introduction

Today's high-performance networked systems are highly complex, consisting of many heterogeneous hardware devices and software working together. These systems are essential to daily lives, from mobile connectivity to inter-cloud communications, from checking emails to running complex scientific workflows on distributed cloud infrastructures.

Editor: Samir Khan.

✉ Mariam Kiran
mkiran@es.net

Extended author information available on the last page of the article

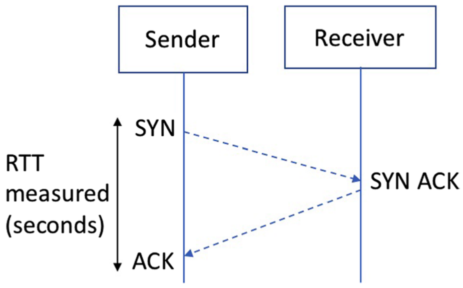


Fig. 1 Round trip time (RTT) measured in the three way communication in TCP

Reliable and high throughput connectivity, security and 99.9% availability are just some of the guarantees warranted. Networking infrastructures support diverse user requirements and are constantly under pressure to provide end-to-end connectivity that is reliable, with high performance, minimum packet loss, and sometimes providing dedicated links for certain data-intensive applications.

Performance and reliability of high-speed networks are affected by many factors, such as congestion, packet loss, end-host I/O performance, end-host network tuning, and more. The complexity of data movement also brings increased potential for failures and performance problems that need to be detected and mitigated. Majority of transfers rely on the TCP protocol to guarantee packet delivery with minimum loss. These have been studied by anomaly detection and reliable packet delivery using monitoring tools such as PerfSonar (Hanemann et al. 2005), sFlow (Jasinska 2006), TCP statistics (Mellia 2002) and more (Hofstede et al. 2014). Multivariate machine learning techniques in finding outliers in traffic, hardware/software failures in security software, and improving utilization (Zander et al. 2005).

Usually, identification of abnormal transfers is done through classification or clustering methods. Examples include using Naive Bayes Theorem, Support Vector Machines, Random Forests or rule-based approaches to detect traffic anomalies (Bansal and Kaushal 2015; Mirza et al. 2010). But using these methods requires large amounts of data about packet numbers, flow direction, IP addresses, QoS values, latency or file traces to characterize normal traffic behavior. These methods also require prior and domain knowledge to help classify what is "normal". However, networks are extremely dynamic and what is normal for one, may not be true for others. Distributed, heterogeneous and complex nature of end-to-end networked systems makes it difficult to collect labeled data about anomalies and failures. Additionally, dynamic characteristics of the platform-induced anomalies often manifest themselves in unknown manners. Hence, lack of labeled data sets makes it impossible to appropriately train and validate any machine learning models for detecting transfer anomalies. Some researchers have used formula-based methods to understand packet behaviors or history-based methods (Mirza et al. 2010). However, again these methods would work only under certain conditions, and fail when network systems change or data set becomes too big to process expeditiously.

In this work, we analyze anomalous network transfers by utilizing data collected using Tstat (Mellia 2002), which is a tool to collect TCP traces for transfers. Figure 1 shows how TCP works. The server sends a packet to a client with a packet load (in bytes), when the client receives this, it sends an "acknowledgment" (ACK) back. The total time for receiving the ACK is recorded as round trip time (RTT). The TCP protocol also uses time windows to allow servers to wait for the ACK, before deciding to resend the packet again. The longer the wait, the window size grows over time, allowing the protocol to adjust its waiting time. This not only affects the total RTT recorded but also builds up retransmissions and overflowing flows in both TCP directions.

The TCP protocol is designed to ensure all packets are delivered reliably. It does this by tracking various packet information, for example window timeouts, packet numbers, RTT and more, to calculate if loss has occurred. If yes, it triggers the host to resend the packets again. Collectively, Tstat traces contain about 150 variables per packet on both server and client sides.

Anomalous network transfers have been classified into three major groups—packet loss, packet duplication and retransmissions (Mellia et al. 2008a; Casas et al. 2016). The retransmissions could be triggered by lack of client acknowledgments or some other error in the link. The TCP window size measures the waiting time before TCP sends the packet again. For instance, this may cause more retransmissions and eventually congestion on the link, causing packet loss eventually. Current approaches classify anomalous behaviors using thresholds and prior known formulas, but due to TCP complexity, it is difficult to understand root causes and relevant features/variables.

This paper departs from using prior knowledge, and uses unsupervised feature extraction to learn normal and abnormal features. Based on three techniques—Principle Component Analysis (PCA), autoencoders and isolation forest—we build and train classifiers to help identify anomalous transfers. The goal of this study is to understand what can be used to classify TCP abnormal behavior and the feasibility of unsupervised classification techniques for doing this.

Approach and contributions We approach the problem of anomaly detection by performing unsupervised feature extraction on TCP traces collected from simple transfers between two nodes (iPerf (Iperf 2000) transfer) and from transfers in a real scientific workflow [1000 Genome workflow (1000 Genomes Project Consortium 2012)], to find unique characteristics from normal transfers and from transfers with synthetically introduced anomalies (e.g. packet loss, packet duplication and synthetic reordering).

To inform the machine learning models with some labeled data sets, the experiments are set up to generate data in a real network environment. File transfers using the iPerf tool and a workflow manager are set up on the ExoGENI networked cloud testbed (exo), where the traces in these controlled conditions, are used to inform common features in a reliable transfer. The purpose of doing two kinds of transfers is (1) to compare the results of workflow transfer versus iPerf transfer experiments, and (2) to learn common features of normal transfers given whatever the infrastructure setup is.

By leveraging innovations in machine learning and existing research on TCP protocols, we make the following contributions in this paper:

- We develop and test multiple unsupervised feature extraction methods, including Principle Component Analysis, isolation forest and autoencoders, to study TCP traces and deduce anomalous features.
- We perform experiments on a real testbed (ExoGENI) with iPerf transfers and real workflow network transfers. Each experiment was run for 1 h with synthetic anomalies like packet loss, duplication, and reordering. We extract key features from TCP statistics to understand packet behavior to improve transfer performance.
- We perform in-depth analysis of issues and data dependencies of the algorithms, laying the foundation for further research on feature extraction and TCP performance.

Motivation for this work This work is unique to TCP anomaly detection. With increasing network complexity and transfer sizes, it is imperative to find classification techniques that can detect bad transfers as soon as they happen. Exploring multiple techniques, we can determine what issues are faced while building these classifiers and whether such tools can be built to work in tandem with monitoring tools.

2 Background and related work

2.1 TCP transfers

In this study we focus on TCP Cubic. Figure 1 describes how generally the TCP handshake works (Mellia et al. 2008b). Researchers have used these statistics to classify normal versus anomalous segments and build predictor systems to see if there is packet loss or alterations (Mellia et al. 2008b; Trevisan et al. 2017).

Most TCP research has focused on understanding if loss is occurring. The length of the congestion window and RTT can give an indication of how throughput is performing, and different TCP variants handle this differently. TCP cubic, in particular, is less aggressive than other variants and more systematic in approach. The congestion window is calculated as a cubic function of time since when loss is observed. This results in two portions, the first being a concave, where the window rapidly grows until before the loss event occurred, and the convex region keeps probing for more bandwidth slowly at first, then rapidly. Cubic eventually reaches a stability between the concave and convex regions, before looking for more bandwidth and growing its throughput. This allows TCP to exhibit the wave behavior between the slow start, ramping up and coming down again. Other research has explored how different TCP variants perform in different congestion conditions (Parichehreh et al. 2018).

2.2 Network anomalies

Anomalies can create bogus traffic causing network congestion and utilization in a router, impacting customer experience. Identifying these in advance is imperative to improving network operations. In their seminal work, Lakhina et al. (2004) discussed how difficult diagnosing anomalies is, because of large amounts of high-dimensional and noisy data. Using principal component analysis, the authors showed how relational measurements can be used to isolate anomalous from normal traffic behaviors. The authors showed that on average anomalous traffic will have higher volumes when compared to their normal counterparts. However, even normal traffic has bursty and highly variable behavior, which makes it difficult to use these relational assumptions as the identifying methods. Wang et al. (2008) designed a multistage feature extraction method to filter information entropy from data through multiple stages to filter abnormal traffic. The authors improved the false alarm rate hugely using this method. Further, Barford et al. (2002) used signal analysis of outages, flash crowds, attacks and measurement failures, based on IP and SNMP measurements. In this paper, we are only focused on TCP and not analyzing further measurements.

2.3 Studying TCP behavior

Although a large amount of literature is devoted to studying how TCP algorithm can be designed to optimize the bandwidth utilized, there is little work to understand TCP behavior itself, when anomalies exist. Some researchers have focused on comparing and understanding day-to-day normal behavior to find abnormal patterns. Machine learning approaches are, in particular, very useful to observe recurring phenomena and extract non-stationary transition patterns (Palmieri and Fiore 2010).

Computing workflows leverage a collection of many elements including compute, storage and networks. In a recent paper, Singh et al. (2017) presented a framework that used machine learning to predict workflow performance and forecast workflow behavior. Presenting results on independent workflows like on the XSEDE SDSC Comet cluster, the paper highlights the dependency of multiple components when trying to predict overall performance. Although most workflow performance improvement techniques have looked at distributed computing to improve performance of computing jobs, networks are becoming overloaded and are prone to breakdowns affecting overall performance. Additionally, Gaikwad et al. (2016) presented an anomaly detection method based on autoregression and time-series prediction to understand how workflow anomalies can be detected on networked cloud infrastructures. This highlights the different experimental setups and the anomalies that can affect a healthy workflow based on network topologies.

Jiang et al. (2014) used network topology and destination data for detecting anomalous behaviors. This method was able to identify ill behaving devices and users. Similarly, Lakhina et al. (2005) proposed using entropy methods to extract multiple traffic features about flows. While entropy was successful in detecting low-volume anomalies, this method was unable to detect anomalies in high-volume data. In these approaches, the feature extraction method was dependent on the current traffic's nature, which can vary dynamically in a large-scale distributed network system. Therefore, these features were not sufficient to identify common anomalies in abnormal file transfers.

Further, Zander et al. (2005) also performed a classification of anomalies based on application identification. Using feature selection techniques for finding optimal flow attributes, the algorithm was able to predict accurate anomalous behavior up to 86%.

Traffic anomalies can also be a result of malicious user or faulty device behavior and can lead to network disruption. Often they can show a degradation of networks, as investigated by Yang et al. (2018) detecting slow file transfers as symptoms of failures. The authors detect slow transfers, based on a performance model and on the observed distribution of file transfer rates. The authors find that both methods found different results in different workflow transfers, based on the nature of the transfer.

Just analyzing file transfer, researchers Gunter et al. (2007) have been investigating anomalies in GridFTP transfers, by creating controlled experiments and injecting faults such as disk injection, latency perturbations and loss, to measure effects on performance. Although this study is focused on GridFTP and statistical methods, it discusses the nature of how anomalies can affect healthy transfers. But et al. (2005) showed that just monitoring RTT is not enough for studying network issues. Applying the Jacobson's algorithm, the authors showed the effect of minimizing packet transfer times on TCP stream jitters, which showed additional factors affecting network behaviors. Muscariello et al. (2006) proposed a heuristic classification approach to identify possible anomalies by quantifying the sensitivity of certain parameters on flow measurements. Although based on established literature data, this technique showed that RTT had a high dependence on current traffic load that affected prediction accuracy.

Feature extraction can also be used to investigate user behavior with traffic models (Rossi et al. 2003). Zhang and Zulkernine (2006) described anomaly detection as a significant step for network intrusion detection systems (NIDS) to work, but highlighted that these systems are based on known or supervised data sets. The authors recognize that a more robust unsupervised feature extraction method is needed, with features learned from real network data sets, to make the NIDS systems more reliable.

In this paper, we focus on using simple unsupervised feature extraction, based on domain knowledge, to extract certain characteristics from known transfer datasets. Using feature extraction and dimension reduction we build relationships among sensitive parameters such as congestion and availability with a transfer file type. These feature filters can be leveraged for any future machine learning methods. With the goal for understanding packet loss, congestion and impact on end-to-end performance. Our results show positive results with isolation forest in identifying packet reordering features, where PCA and autoencoders struggle.

3 Problem formulation

Networks have a significant impact on scientific workflow performance. Gaikwad et al. (2016) show how science workflows can suffer on multiple levels of hardware infrastructure, software, middleware, application and workflows reacting to anomalous events during execution. To monitor these behaviors, researchers deploy measurement tools (e.g. perfSONAR, Tstat, netflow, sflow, to name a few), monitoring various properties like usage traffic, router health, link performance, and more. Each of these tools focuses on different aspects: PerfSONAR actively probes packet loss, delay, jitter and utilization, whereas netflow records packet data with IP addresses, bytes sent and the hops taken. In this paper, we primarily focus on TCP statistics to determine features for file transfers.

The Tstat tool records TCP statistics such as client/server IP address, packets, RTT, TCP window and more (Mellia 2002). Mellia et al. (2008a) analyze TCP packet anomalies, while Vassio et al. (2017) and Trevisan et al. (2018) mine Tstat to understand passive and active network attacks. They use thresholds and prior knowledge to determine if an anomaly is present.

3.1 What makes a good data set for feature extraction?

Machine learning techniques need correct and large data sets for training purposes. A better training set can greatly improve the accuracy of the machine learning model. In this paper, we use two sets of network experiments: (1) controlled experiment with two nodes generating synthetic iPerf traffic, and (2) real scientific workflow experiment using the 1000 Genome workflow software (1000 Genomes Project Consortium 2012). Traces are recorded for successful transfers and anomalous transfers with packet loss, duplication and reordering packets. These labeled traces are then fed to unsupervised feature extraction methods to extract features. Table 1 and 2 document the traces collected per experiment run. Both experiments, have different network topology setups and different transfers.

3.2 Data preparation: skewed iPerf transfers

iPerf sometimes randomly sends very large packets in one flow. This creates various throughput discrepancies and affects the overall sample reliability. To prevent this, large transfers were removed using k-means clustering. These clusters help group the very large transfers as shown in Fig. 2, ensuring the generation of correct data samples. The metric used to remove these bad transfers was packet size larger than the expected.

Table 1 Scenario 1: iPerf transfer

No.	Experiment type (runs for 1 h)	Tstat sample
1	Normal traffic	152
2	No flow	120
3	1% Loss	120
4	5% Loss	120
5	1% Packet duplication	60
6	5% Packet duplication	60
7	25–50% Packet reordering	50
8	50–50% Packet reordering	60

Table 2 Scenario 2: 1000 Genome workflow

No.	Experiment type	Tstat sample
1	Normal workflow runs	1475
2	1% Loss	1563
3	2% Loss	1636
4	3% Loss	1948
5	1% Packet duplication	1574
6	5% Packet duplication	1527
7	25–50% Packet reordering	1491
8	50–50% Packet reordering	1592

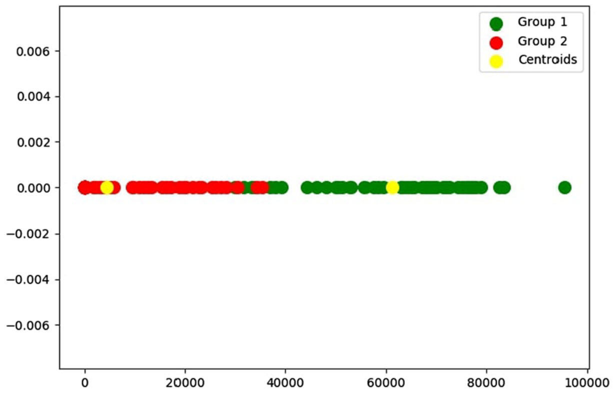


Fig. 2 K-means to identify bad iPerf transfers

3.3 Tstat logs

Tstat collects logs where each record corresponds to a single TCP flow. It collects data in both directions: (1) Client-to-Server (C2S) and (2) Server-to-Client (S2C), while each Tstat record contains approximately 150 variables. Among these variables there are details on the source and destination IP addresses, ports, number of bytes transferred, completion times, congestion window sizes and more. Each TCP connection is established by the first SYN segment and ends either when a FIN segment is observed or the default timeout of 10 s is reached. Tstat discards all connections for which the three-way handshake is not properly seen in the log_tcp_nocomplete log file, while it logs the rest of them in the log_tcp_complete file. A comprehensive list of Tstat variables in Finamore et al. (2011).

3.4 Unsupervised feature extraction algorithms

A feature is a property of a data sample. Average, mean, median and standard deviation can also be features. Unsupervised feature extraction can be used to identify these features from a trace log data.

3.4.1 Principal component analysis

Principal component analysis (PCA) (Jolliffe 2011) is an unsupervised feature extraction algorithm that can be used to reduce dimensions and extract vectors summarizing data properties and correlation among variables. PCA depends on linear combinations and constructs new features (e.g. eigenvectors) that summarize variation among data variables. These eigenvectors can help find the most influential variables (out of 150) and also reduce the data dimension for faster processing. In this paper, we use PCA to extract eigenvectors as features of the packet transfer. The kth principal component of a data vector $x(i)$ can be given a score $tk(i) = x(i) \otimes w(k)$ in a transformed coordinates space. The corresponding vector in this space is $x(i) \otimes w(k)$, where $w(k)$ is the kth eigenvector of $X^T X$.

The transfers are labeled per scenario and fed into the PCA algorithm. To visualize these eigenvectors, we can plot these to identify if two clear clusters are formed.

3.4.2 Autoencoders

Autoencoders (Bengio 2009) belong to the class of deep learning algorithms and are used for data or image compression and decompression. Autoencoders take training data input, compress it to reduce the dimensions into a compressed representation. These are then decoded from the compressed version to find the original data set, by calculating a loss between the result and the original input. Therefore, the algorithm removes noise (or randomness) in the input data and learns key features only. Autoencoders are used in image recognition and text analysis, where they can successfully compress features into learned sets.

Autoencoders take a high-dimensional space [e.g. 150×100 (150 variables and 100 transfers)] and reduce it to vectors of size 15,000. We train the autoencoder to minimize the loss rate between the original input and output data. The reduced dimensions can then be visualized on a 2D space to visualize clusters formed. Autoencoders consist of two parts: (1) the encoder ϕ and (2) the decoder ψ of the input data set X, represented by:

$$\phi : X \rightarrow F \tag{1}$$

$$\psi : F \rightarrow X \tag{2}$$

$$\phi, \psi = argmin||X - (\psi \cap \phi)X||^2 \tag{3}$$

In our case, the transfers are labeled based on their experiment number and fed to autoencoder to reduce the dimensions and extract features.

3.4.3 Isolation forest

Isolation forest (Liu et al. 2008) is a relatively new, unsupervised classification algorithm that is commonly used for outlier detection in high-dimensional data sets. This algorithm is based on the fact that anomalies are data points that are usually fewer in number and have different distribution patterns from the normal data. The algorithm constructs a separation by first creating isolation trees, or random decision trees, and then calculates the score as the path lengths to isolate observations. It isolates these observations by randomly selecting a feature and then randomly selecting a split value between the maximum and minimum values of the selected feature. Usually, random partitioning produces noticeably shorter paths for anomalies. Hence, a forest of random trees producing shorter path lengths for some samples are highly likely to be anomalies.

```
Initialize Forest = ( )
Set $i$ tree height $h = ceiling(log_2 N)$
**for** $i$=1 to $L$ **do**
    $X = sample(X, N)$
    Forest = $iTree(X, 0, h)$
**end for**
**return**
```

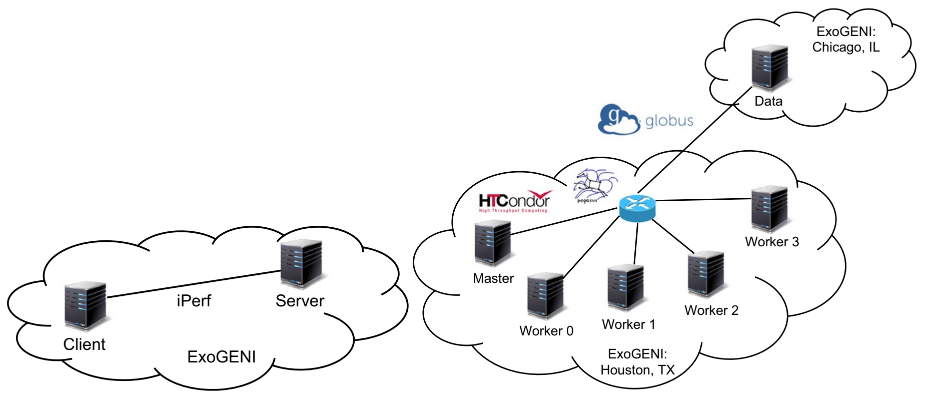
With X as input data, L as a number of trees and N as training samples, the isolation forest algorithm is shown in Algorithm above. We train the isolation forest using data collected from normal workflow transfers. The other experiments are then fed as test data, to check if the isolation forest can predict whether the corresponding network flow is normal or anomalous.

4 Experimental evaluation

4.1 Experiment setup

In the iPerf transfer experiment (Fig. 3a), we set up two nodes on ExoGENI testbed, each node has 4 cores and 12 GB RAM. The two nodes are connected via a dedicated 500 Mbps network link. We use iPerf to generate TCP traffic and use Linux TC (Hubert et al. 2002) to introduce synthetic network anomalies, like packet loss, duplication and reordering.

For the second experiment, we set up an HTCondor cluster with 4 worker nodes on ExoGENI, (shown in Fig. 3b). The bandwidth is set to 500 Mbps for all links and we use the 1000 Genome workflow composed of five different tasks: (1) *individuals*—task fetches and parses the data from the 1000 Genomes project per chromosome; (2) *populations*—the task fetches and parses five super populations (African, Mixed American, East Asian, European, and South Asian), and a set of all individuals; (3) *sifting*—task computes the SIFT scores of all of the SNPs (single nucleotide polymorphisms) variants, as computed by the Variant Effect Predictor; (4) *pair overlap mutations*—task



(a) iPerf Transfer experiment. **(b)** 1000 Genome Workflow experiment.

Fig. 3 Experiment set up for iPerf transfers and 1000 Genome workflow

measures the overlap in mutations (SNPs) among pairs of individuals; and (5) *frequency overlap mutations*—task calculates the frequency of overlapping mutations across subsamples of certain individuals.

4.2 Extracting TCP features

4.2.1 PCA: dimension reduction for identifying variance features

PCA is a commonly used dimension reduction method allowing large sets of variables to be reduced to smaller sets containing most information. The new dimensions are sectioned into principal components, with first component showing major variability and remaining components containing the rest of variability. Mainly these two principal components can be used to create a linear combination of variables based on the maximum variance among them. The second linear combination explains the maximum proportion of the remaining variance. PCA produces eigenvectors which plot the variance and weight correlations explaining the weighted component directions. By applying PCA on our two datasets of approx. 150 variables, we are able to extract useful information about them.

Figure 4 shows the 2D PCA plots of both iPerf transfers and 1000 Genome workflow. Reducing the feature dimensions, Fig. 4a shows that all normal transfers lie along a diagonal of the two components, while all transfers with anomalies have a higher variance in the second principal component. However, this behavior is not observed in the 1000 Genome workflow. Here all good and bad transfers, all lie along the diagonal of the two principal components (Fig. 4b). Therefore the variance in the data is less distinct in the 1000 Genome workflow transfers.

Further, PCA clusters can be analyzed to study how many clusters contain the most variance. Figure 5 shows that in the 1000 Genome workflow, about 60% of the variance is captured in the first two components, justifying the 2D plots. However, unique variance features are still not detected.

PCA is also able to extract the most relevant variables out of the 150 variables we collected to describe the transfer. These influential variables are reduced into 26 variables that include Round Trip time (RTT), Return time Observed (RTO), congestion window details

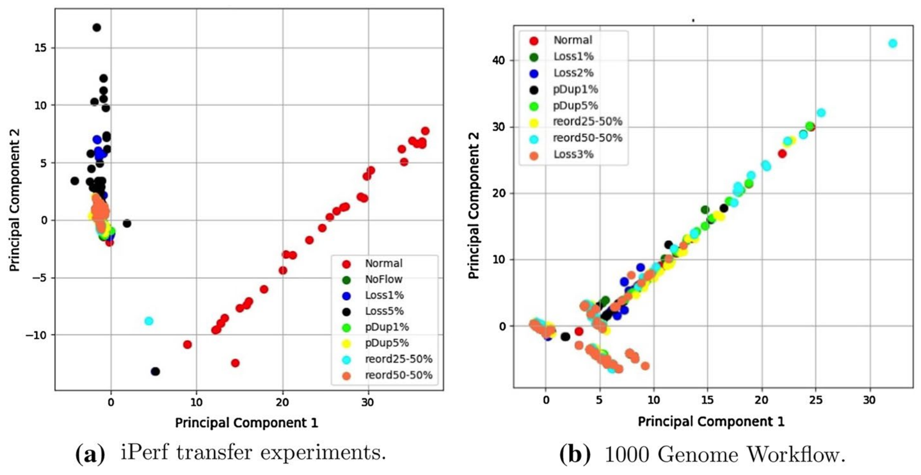


Fig. 4 Feature extraction using PCA

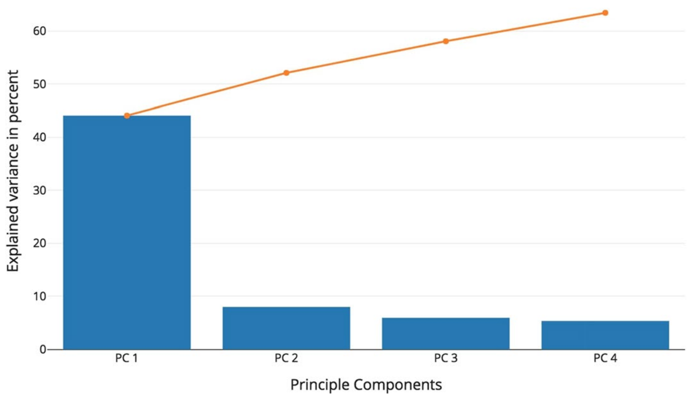


Fig. 5 Optimal PCA clusters in 1000 Genome workflow based on variance

(including size, scaling and segment sizes), which are known in TCP research (Mellia et al. 2008a).

4.2.2 Autoencoders: compressing and decompressing to find features

Autoencoders use artificial neural networks to learn unique data features by first compressing the data into lower dimensions and then reconstructing the compressed data into the original data that are as close to the original set as possible. Using backpropagation, most autoencoder architectures extract features of the input distribution as a *good representation* of the data and remove any noise by computing the loss between the reconstructed output and original input. Trained on linear combinations, autoencoders behave very similar to PCA, where the weights of the hidden layers represent the variance among the principal components (Bengio 2009). However, these weights are not the same as principal components. In our experiments, we encoded the TCP traces into a three-layer fully connected

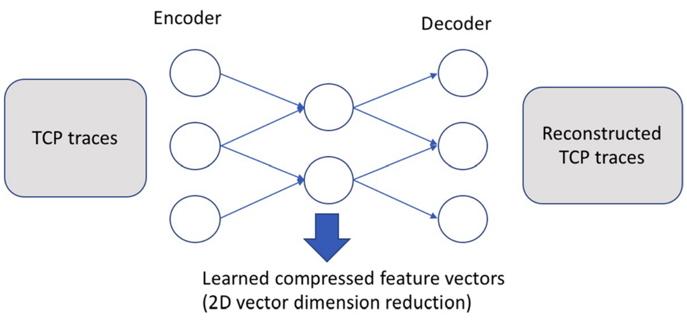

Encoder
Decoder
TCP traces
Reconstructed TCP traces
Learned compressed feature vectors
(2D vector dimension reduction)

Fig. 6 Autoencoder architecture constructed to extract TCP features

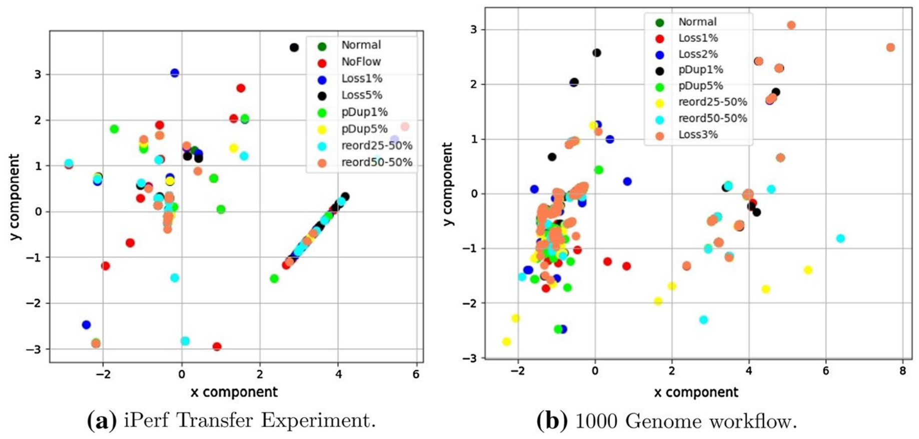


Fig. 7 Feature extraction using autoencoder

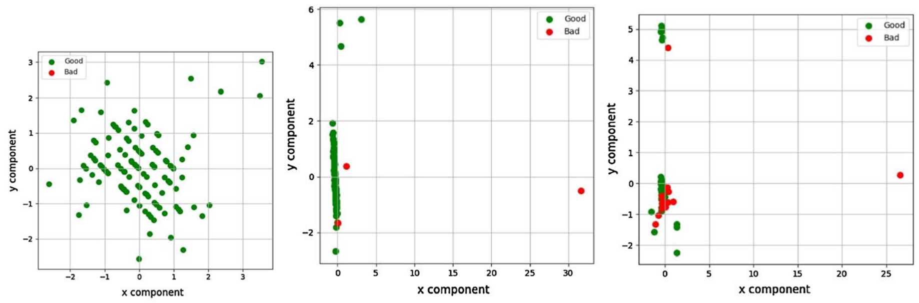
dense neural network, using one layer to encode the traces, one hidden layer and one output layer to decode them. Figure 6 shows the layer architecture of the autoencoder used. This autoencoder had only three layers–layer 1 for encoder using RELU activation to process the input data, two neuron layer for learning representations and, finally third layer for the decoder that uses the SIGMOID activation function.

During the training phase, the autoencoders learn a 2-dimension feature vector, extracted as learned features. These are presented in Fig. 7.

Comparing to PCA results, the autoencoders also reduce the TCP dimension. however, no clear clusters are formed for good and bad file transfers. While autoencoder have given successful results in images, for text data they are more complicated to work with and require additional layers and training.

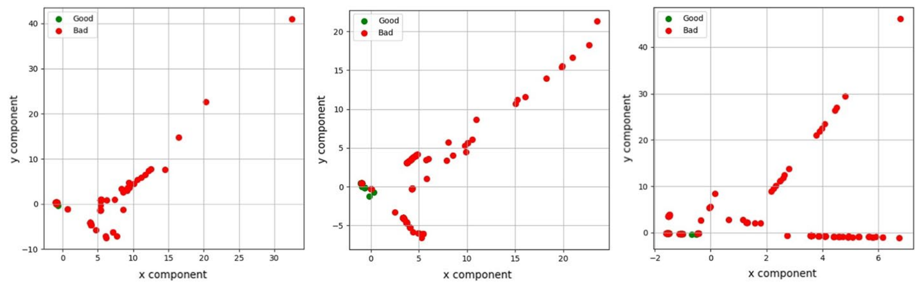
4.2.3 Isolation forest: detecting anomalies through tree classification and scoring

Isolation forest works by building a tree and classifying data into known nodes on a normal data set. Test data are then parsed using the same data assigning scores of how close the data point lies on the tree. This score can be used to identify data points which lie far from the tree classes (or anomalies) or not. We optimized the tree by performing hyperparameter



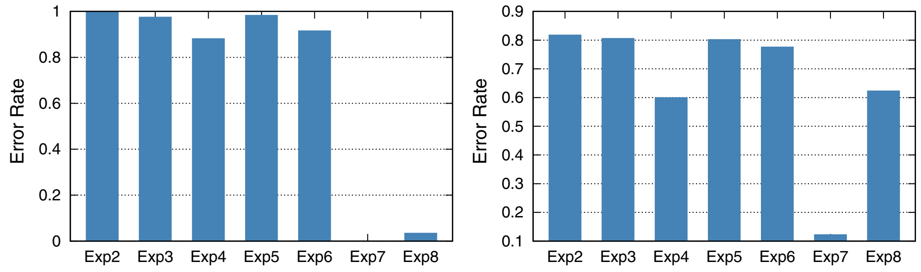
(a) No Flow. **(b)** Packet loss 1%. **(c)** Packet Loss 5%.

Fig. 8 Isolation forest identifying good and bad transfers in iPerf transfer experiments with packet loss



(a) Packet Loss 1%. **(b)** Packet loss 2%. **(c)** Packet Loss 3%.

Fig. 9 Isolation forest identifying good and bad transfers in 1000 Genome workflow in packet loss



(a) iPerf transfer experiment. **(b)** 1000 Genome Workflow.

Fig. 10 Isolation forest prediction error rate

tuning of the number of depth and leaves specified. These results are presented for optimal conditions found.

Figures 8 and 9 show the results of isolation forest on the two experiments. In Fig. 8a, the no-flow experiment shows that there is some TCP noise in the system, which are random monitoring tool transfers recorded in the link.[1] As packet loss is increased, most transfers are not detected as bad, which is also the case with packet duplication. In packet

[1] PerfSonar tool generates TCP transfers for its monitoring purposes.

reordering, the anomaly score is quite high and catches most of the transfers as bad, giving an error rate of 0.05% (Fig. 10a).

In the 1000 Genome workflow (Fig. 9), the error rate is slightly better, as it is able to predict more bad transfers. However, it is still unable to extract key features between the experiments with loss and duplication.

5 Discussions

In this section, we discuss the lessons learned from building unsupervised feature extraction classifiers.

Transfer learning does not work in networks Transfer learning can be used to generate training data from smaller sets of experiments to use machine learning models to learn from the representative sets (or simulation-based scenarios) about the real-world problems, where data is often difficult to collect. For example, using simulators to learn real-world driving conditions help improve the models about certain road conditions, for which data does not exist at present (Bengio 2009).

Most machine learning approaches assume that the training and the test data belong to the same distribution. However, in situations where labeled data is generated via transfer learning, it is often that machine learning models trained on controlled experiments cannot recognize reasonable conclusions and features in real-world examples. This implies that test data is a different data distribution than the training data. In traditional transfer learning, the labeled and unlabeled data, both come from the same class to aid with classification (Raina et al. 2007). However, in situations where this is not possible, Dai et al. (2007) talks about using a learner which can calculate the difference between training and test data, to improve the overall predictions.

This behavior was also witnessed in both network transfer experiments. Any features learned in the iPerf transfers were not detected in the 1000 Genome workflow experiments, which means transfer learning cannot be used here to learn unique features with different network topologies.

PCA, as a feature extraction, is not enough Dimension reduction is a useful technique to extract higher-level features of a data set. PCA is a useful technique which caters to this by using variance matrix, covariance matrix, eigenvector and eigenvalues. However, for our application, we see that PCA is highly dependent on the numerical data. Thus, these cannot be general vector spaces, that can be applied in all situations.

The feature extraction method allows us to reduce the dimension space of 150 TCP variables to 26 key variables, including RTT, RTO, segment size and TCP window details. Additionally, both scenarios have different network topologies. The iPerf transfers had two nodes versus the 1000 Genome workflow experiment used ExoGENI sites in Jacksonville, FL and Chicago, IL. This makes the RTT in the workflow to be much higher than the iPerf experiment. Because PCA is dependent on variables, this completely threw off the results and was not able to classify good or bad transfers.

Autoencoders cannot capture behavior data through compressed features Similar to PCA, autoencoders capture the learned weights of the hidden layers to determine the best reconstruction ratio from input to output. Results show that although these do not capture variance, the learned representations are not sufficient to identify unique clusters of good and bad transfer characteristics.

Isolation features show that high-level relationships exist Our experiments are able to capture packet reordering cases which shows that decision trees are able to build some high-level relationships among the data set. The anomaly score to reordered packets is very high and extracting this feature can help researchers to look for particular relationships. Further work in this area is needed to extract these features separately.

6 Conclusion

Feature extraction is a powerful technique to identify behavior characteristics, which can be used to identify anomalous transfers as they happen. In this paper, we used a variety of unsupervised feature extraction, where algorithms were fed only training data sets, to learn and extract what they thought was a feature to predict an anomaly. Using PCA we were able to find clear clusters of normal and abnormal transfer, but this was not seen for the 1000 Genome workflow experiment. Alternatively, autoencoders completely failed in extracting clusters on normal or abnormal transfers in both experiments. Isolation forest, being trained on the normal transfers, was able to give better results, but sometimes did not work (Fig. 10).

The results give a number of unique conclusions. The iPerf transfer experiment was run on two VMs that existed on the same rack, whereas the 1000 Genome workflow involved geographically distributed nodes, Jacksonville and Chicago respectively. The added RTT in the second scenario affected the performance of the PCA. Isolation forest is worth exploring further with more extensive experiments across multiple workflows. In the future, we will be extending the isolation forest result with more loss, duplication and reordering experiments, to find key relationships to reduce error rates as experiments have more anomalies in them.

Acknowledgements This work was funded by DOE contract number #DE-SC0012636M, “Panorama 360: Performance Data Capture and Analysis for End-to-end Scientific Workflows”. Any opinions, findings, or conclusions in this article are those of the authors and do not necessarily reflect the views of the sponsor.

References

1000 Genomes Project Consortium. (2012). A global reference for human genetic variation. *Nature*, *526*(7571), 68–74.

Bansal, N., & Kaushal, R. (2015). Unusual internet traffic detection at network edge. *International Conference on Computing and Network Communications (CoCoNet).*

Barford, P., Kline, J., Plonka, D., & Ron, A. (2002). A signal analysis of network traffic anomalies. In *SIGCOMM Work. on Internet Measurement* (pp. 71–82). ISBN 1-58113-603-X. https://doi.org/10.1145/637201.637210.

Bengio, Y. (2009). Learning deep architectures for AI. *Foundations and Trends in Machine Learning*.

But, J., Keller, U., Kennedy, D., & Armitage, G. (2005). Passive TCP stream estimation of RTT and jitter parameters. In *The IEEE conference on local computer networks (LCN)*.

Casas, P., Fiandino, P., Wassermann, S., Traverso, S., D’Alconzo, A., Tego, E., et al. (2016). Unveiling network and service performance degradation in the wild with mplane. *IEEE Communications Magazine*, *54*, 71–79.

Dai, W., Yang, Q., Xue, G.-R., & Yu, Y. (2007). Boosting for transfer learning. In *Proceedings of the 24th international conference on machine learning*, ICML (pp. 193–200). ISBN 978-1-59593-793-3.

Finamore, A., Mellia, M., Meo, M., Munafo, M. M., Torino, P. D., & Rossi, D. (2011). Experiences of internet traffic monitoring with tstat. *IEEE Network*, *25*(3), 8–14. https://doi.org/10.1109/MNET.2011.5772055. ISSN 0890-8044.

Gaikwad, P., Mandal, A., Ruth, P., Juve, G., Krol, D., & Deelman, E. (2016). Anomaly detection for scientific workflow applications on networked clouds. In *International conference on high performance computing and simulation*.

Gunter, D., Tierney, B. L., Brown, A., Swany, M., Bresnahan, J., & Schopf, J. M. (2007). Log summarization and anomaly detection for troubleshooting distributed systems. In *Proceedings of the 8th IEEE/ACM international conference on grid computing*, GRID (pp. 226–234). ISBN 978-1-4244-1559-5.

Hanemann, A., Boote, J. W., Boyd, E. L., Durand, J., Kudarimoti, L., Łapacz, R., et al. (2005). PerfSONAR: A service oriented architecture for multi-domain network monitoring. In B. Benatallah, F. Casati, & P. Traverso (Eds.), *Service-Oriented Computing—ICSOC 2005*. Berlin: Springer.

Hofstede, R., Celeda, P., Trammell, B., Drago, I., Sadre, R., Sperotto, A., & Pras, A. (2014). Flow monitoring explained: From packet capture to data analysis with NetFlow and IPFIX. In *IEEE Communications Surveys and Tutorials* (pp. 2037–2064). IEEE Communications Society.

Hubert, B., Graf, T., Maxwell, G., van Mook, R., van Oosterhout, M., Schroeder, P., Spaans, J., & Larroy, P. (2002). Linux advanced routing and traffic control. In *Ottawa Linux Symposium* (vol. 213).

Iperf. (2000). https://iperf.fr/.

Jasinska, E. (2006). Sflow, I can feel your traffic. In *Amsterdam Internet Exchange (AMS-IX)*.

Jiang, D., Zhengzheng, X., Zhang, P., & Zhu, T. (2014). A transform domain-based anomaly detection approach to network-wide traffic. *J. Netw. Comput. Appl.*, *40*(C), 292–306.

Jolliffe, I. (2011). Principal component analysis. In *International Encyclopedia of Statistical Science* (pp. 1094–1096).

Lakhina, A., Crovella, M., & Diot, C. (2004). Diagnosing network-wide traffic anomalies. In *SIGCOMM* (pp. 219–230). ISBN 1-58113-862-8. https://doi.org/10.1145/1015467.1015492.

Lakhina, A., Crovella, M., & Diot, C. (2005). Mining anomalies using traffic feature distributions. In *Proceedings of the 2005 conference on applications, technologies, architectures, and protocols for computer communications (SIGCOMM)* (pp. 217–228).

Liu, F., Ming, K. T., & Zhou, Z.-H. (2008). Isolation forest. In *Proceedings of the 2008 eighth IEEE international conference on data mining*, ICDM (pp. 413–422). ISBN 978-0-7695-3502-9.

Mellia, M. (2002). TCP statistic and analysis tool. *IEEE Network*, *16*.

Mellia, M., Meo, M., Muscariello, L., & Rossi, D. (2008a). Passive analysis of TCP anomalies. *Computer Networks*, *52*, 663–2676.

Mellia, M., Meo, M., Muscariello, L., & Rossi, D. (2008b). Passive analysis of TCP anomalies. *Computer Networks*, *52*(14), 2663–2676. https://doi.org/10.1016/j.comnet.2008.05.010. ISSN 1389-1286.

Mirza, M., Sommers, J., Barford, P., & Zhu, X. (2010). A machine learning approach to TCP throughput prediction. *IEEE/ACM Transactions on Networking*, *18*(4), 1026–1039.

Muscariello, L., Mellia, M., & Meo, M. (2006). Passive identification and analysis of tcp anomalies. *Distributed Cooperative Laboratories: Networking, Instrumentation, and Measurements*.

Palmieri, F., & Fiore, U. (2010). Network anomaly detection through nonlinear analysis. *Computers and Security*, *29*(7), 737–755.

Parichehreh, A., Alfredsson, S., & Brunstrom, A. (2018). Measurement analysis of TCP congestion control algorithms in LTE uplink. In *Network traffic measurement and analysis conference*.

Raina, R., Battle, A., Lee, H., Packer, B., & Ng, A. Y. (2007). Self-taught learning: Transfer learning from unlabeled data. In *Proceedings of the 24th international conference on machine learning*, ICML (pp. 759–766).

Rossi, D., Mellia, M., & Casetti, C. (2003). User patience and the web: A hands-on investigation. *Global Telecommunications Conference*.

Singh, A., Rao, A., Purawat, S., & Altintas, I. (2017). A machine learning approach for modular workflow performance prediction. In *Proceedings of the 12th workshop on workflows in support of large-scale science*, WORKS (pp. 7:1–7:11). ISBN 978-1-4503-5129-4.

Trevisan, M., Drago, I., & Mellia, M. (2018). Measuring web speed from passive traces. In *ACM, IRTF and ISOC applied networking research workshop 2018 (ANRW 18)*.

Trevisan, M., Finamore, A., Mellia, M., Munafo, M., & Rossi, D. (2017). *IEEE Communications Magazine*, *55*(3), 163–169. https://doi.org/10.1109/MCOM.2017.1600756CM. ISSN 0163-6804.

Vassio, L., Figuereido, F., Paula, A., da Silva, C., Mellia, M., & Almeida, J. (2017). Mining and modeling web trajectories from passive traces. *IEEE Bigtable*.

Wang, H., Gong, Z., Guan, Q., & Wang, B. (2008). Detection network anomalies based on packet and flow analysis. In *International conference on networking*.

Yang, M., Liu, X., Kroeger, W., Sim, A., & Wu, K. (2018). Identifying anomalous file transfer events in LCLS workflow. In *Proceedings of the 1st international workshop on autonomous infrastructure for science*, AI-Science (pp. 7:1–7:4). ISBN 978-1-4503-5862-0.

Zander, S., Nguyen, T., & Armitage, G. (2005). Automated traffic classification and application identification using machine learning. In *Proceedings of the The IEEE conference on local computer networks 30th anniversary*, LCN (pp. 250–257). ISBN 0-7695-2421-4.

Zhang, J., & Zulkernine, M. (2006). Anomaly based network intrusion detection with unsupervised outlier detection. In *IEEE Communications*.

Affiliations

Mariam Kiran[1] · Cong Wang[2] · George Papadimitriou[3] · Anirban Mandal[2] · Ewa Deelman[3]

Cong Wang
cwang@renci.org

George Papadimitriou
georgpap@isi.edu

Anirban Mandal
anirban@renci.org

Ewa Deelman
deelman@isi.edu

1 Lawrence Berkeley National Laboratory, Berkeley, CA, USA

2 RENCI, University of North Carolina, Chapel Hill, NC, USA

3 Information Sciences Institute, University of Southern California, Marina del Rey, CA, USA