2D2N: A Dynamic Degenerative Neural Network for Classification of Images of Live Network Data

Kieran Flanagan
Software Research Institute
Athlone Institute of Technology
Athlone, Ireland
k.flanagan@research.ait.ie

Enda Fallon
Software Research Institute
Athlone Institute of Technology
Athlone, Ireland
efallone@ait.ie

Paul Jacob
Department of Electronics, Computer and Software
Athlone Institute of Technology
Athlone, Ireland
pjacob@ait.ie

Abir Awad
Faculty of Science and Technology
Bournemouth University
Dorset, UK
aawad@bournemouth.ac.uk

Paul Connolly
The NPD Group, Inc
Athlone, Ireland
paul.connolly@npd.com

***Abstract*—The detection of new, novel attacks on organizational networks is a problem of ever-increasing relevance in today's society. Research in the area is focused on the detection of "Zero-Day" and "Black Swan" events through the use of machine learning technologies. Where previous technologies needed a known example of malicious behavior to detect a similar event, recent advances in anomaly detection on network activity has shown promise of detecting novel attacks. In a real word environment however, novel behavior occurs relatively frequently as users utilize new software applications and new standards in networking. Changes such as these, while of notable importance to network security technicians, may not present themselves as an imminent threat to a network. This paper proposes a novel method for the detection and classification of changes in networking behavior. Through the use of a Dynamic Degenerative Neural Network (2D2N), changes in recognizable user activity are dynamically classified and stored for future reference. Through the use of a time-based entropy function, infrequent activity can be analyzed and given precedence over frequent activity. This aids in the classification of abnormal activity for fast, efficient assessment by the relevant persons in an organization. The proposed method enables the detection, classification and scoring of any and all user activity on a network. Evaluation of the proposed method is based upon live data gathered from a large, multinational organization.**

Keywords— Convolutional Neural Network, Network Security, NetFlow Analysis, Image Change Detection.

I. Introduction

With increased publicity surrounding the possibility of network intrusions, large organizations are focused, now more than ever, in avoiding a seemingly unavoidable situation. Traditionally, security applications focused on preventing intrusion from external sources through monitoring for previously identified threats. However, with the increase in prominence of the "Insider Threat" and "Zero-Day" vulnerabilities being exploited, both research and industry have now focused on employing anomaly detection mechanisms. The aim of which is to detect previously un-encountered, possibly malicious activity, in real time.

Within a real-world environment, this can become challenging. In a global network, being utilized by both automated services and end points (a human element), creating a pattern of predictability can become challenging. While a single employee may start to use a new online application for streaming music, this change in behavior is not of concern to network security technicians in a strict sense. However, if this new application contains a possible vulnerability, the network security team within the organization would need to be aware of its use as soon as possible to ensure that all precautionary measures are taken in a swift manner.

Anomalies, such as these, that do not pose an imminent threat to the network, are still beneficial from a security standpoint. Within any large organization, it is reasonable to assume that compartmentalization occurs within certain departments. Through the analysis of all network traffic, additions to the network, that are not malicious in any way, may be detected. This improves both the situational awareness of what is happening within the organization and increase the security posture of the company through the identification of possible misconfigurations that, again, are not malicious, but may impact other aspects, such as network performance.

While the analysis of the behavior of individual assets and users on a network can be beneficial, the abstraction of the resultant metrics can also be advantageous. Having the ability to swiftly place context into a detected anomaly can lead to different determinations about the anomaly detected. For example, if a new user accesses a human resources database, who the new user is becomes extremely relevant.

In this paper, we propose the use of a novel Dynamic Degenerative Neural Network (2D2N) to give the ability to detect new, possibly malicious activity on a network, and classify it dynamically with relation to time. This enables the quick assessment of deviations within previously classified network activity. While the classification of activity is

978-1-5386-5553-5/19/$31.00 ©2019 IEEE

abstracted, key metrics in identifying the asset involved are maintained throughout, for example an IP address or a hostname. This, when presented in context of an anomaly, allows for the quick differentiation between new and abnormal activity.

In this paper, a proposed architecture for the generation and analysis of images generated from network data is proposed. Though low-level analysis of granular key performance metrics, this research aims to detect small anomalies in NetFlow data that contribute to the clusters generated through typical clustering mechanisms used in unsupervised anomaly detection for network security [1]–[3]. Through image generation, through self-organizing map techniques, it is possible to detect malicious behavior within small scale networks [4]. However, within a large network, granular metrics on a minute to minute basis can have incredible volume. In the context of this research within the NPD Group, Inc, a given minute may produce a total of between 50000 and 160000 network flow samples. Producing a SOM image that successfully captures the range of traffic that can occur is incredibly difficult. Given the magnitude of the records produced in a given time.

In the proposed architecture, a divide and conquer method is used to reduce the scope of the data to be captured within a self-organizing map. This is done through the pre-processing of the NetFlow samples collected by a clustering mechanism. By utilizing a novel convolutional neural network, it is possible to abstract the granular detection into a single class representation of a divergence from normality. While this divergence from normality may not be malicious in nature, results demonstrate that the detection of non-malicious deviations can provide insight into the behavior of applications used on the network, increasing the situational awareness and security posture of the network.

In Section II, the related work is discussed. Section III introduces the 2D2N architecture and the context in which this research was conducted in. Section IV describes the evaluation of the proposed neural network in a real-world environment and the results therein. Finally, Section V discusses the conclusions of the live experiment and the results gained from it.

II. Related Work

Network anomaly detection has typically consists of comparative analysis of known threats against current activity on a network [5], [6]. However, while highly accurate and effective, it is possible for such mechanisms to fail in the case of a “Black Swan” or “Zero-Day” attack. In such instances, it is not possible to have a previous, known example that is required by signature-based mechanisms. It is this limitation that has led to the rise in anomaly-based detection mechanisms being used for network intrusion detection.

Anomaly detection mechanisms, in general, work off of the principle of detecting new, never before seen activity [7]. More recently, methods in monitoring deviations from normality, as well as new activity, have been gathering momentum [8]–[10]. As a side-effect of such methods, false positives are expected when using such mechanisms, particularly when used within a live environment. While a deviation from normality might not be indicative of malicious behavior, it still may be of concern to network security personnel within a company however. For example, if a new asset appears on the network and generates some quite normal HTTPS traffic, the network activity cannot be considered as malicious. However, the addition of a new asset on the network is significant [11].

To capture this dogma, behavioral analysis of assets on a network was a natural evolution of standard network anomaly detection. In these instances, behavioral patterns describing the assets on the network are generated in order to detect deviations from such traffic [12], [13]. On a live network however, behavior patterns change rapidly, as new software is deployed and new employees are introduced. The balancing of these, technically false-positive results, with the increase in situational awareness they provide, is non-trivial. In a real-world situation, network-based assets, employees and software are continually evolving and ever changing. Generating and relying on behavioral pattern analysis for assets or users can become extremely challenging [8]. This results in a quid pro quo situation.

Through the generalization of granular predictive analytics, such as a per user metrics, false positives are generally reduced at the cost of context regarding specific users being lost. For example, it is common that malware predicative mechanisms do not consider an I.P address or a hostname when classifying network traffic. The representation of this data is also vital to the understanding of it [9]. This is a non-trivial task, as the high level of granularity needed for effective detection of possible misconfigurations and malware acting on the network. To easily represent possibly thousands of records, effective “at a glance” analysis is vital to decide if an immediate response is required.

III. Proposed Architecture

In this section, a proposed architecture for the generation and analysis of images generated from network data is proposed. This multi-stage process is designed to enable the detection of abnormal network activity. Figure 1 outlines the three stages used in this experiment. The objective of each stage is to process the previous stages output, while abstracting the granular information gained from NetFlow data gathered from within the NPD Group’s network. This abstraction is designed to ensure that quick, at-a-glance analysis can be conducted on the detected abnormalities to determine if any action is needed.

The proposed architecture is divided into three distinct layers, each processing the output of the previous layer. The first layer deals with clustering the raw NetFlow data gathered from within NPD. The second layer focuses on the generation of images from data pre-clustered within layer one. The aim of this step is to, after training, map the contributing NetFlow’s of a cluster to an organized image. And finally, the third layer consists of 2D2N, a novel convolutional neural network designed to detect changes in the images generated from the

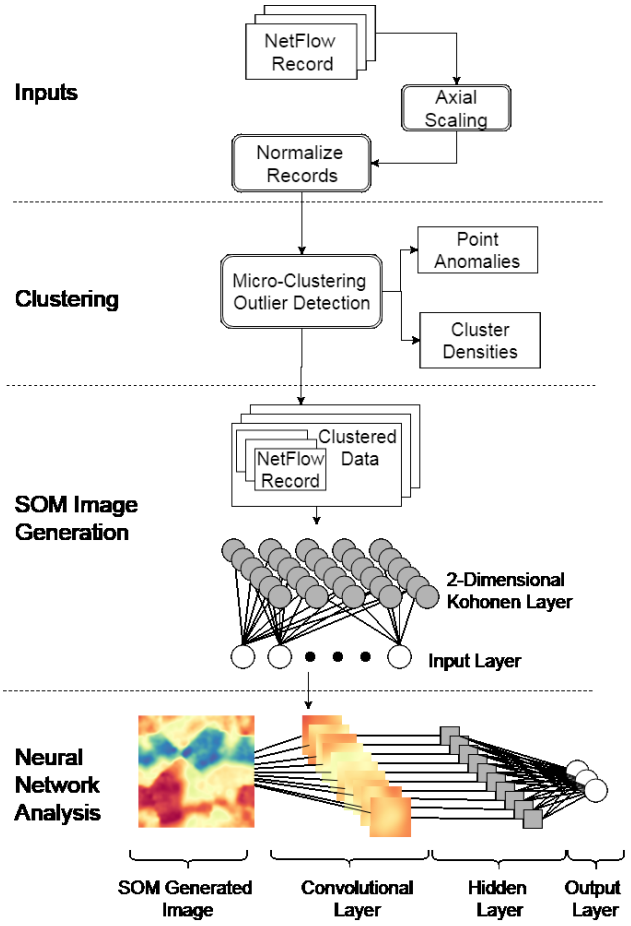

Figure 1: Testing architecture.

previous step. If an anomaly is detected, a new class output is created, and new convolutional layers are added referencing this change in distribution or consistency.

A. Clustering

For the experiment, MCODT [14], Micro-Clustering Outlier Detection in Time Series, was used. This algorithm was used to cluster relative NetFlow together. This enabled the quick correlation of similar NetFlow together into a single cluster. In this experiment, NetFlow data over the course of six weeks was analyzed. Over 6 billion NetFlow records were analyzed by the clustering mechanism. This data contained all network records for all devices on the NPD global network. All data was unlabeled, and no previous assertions were made.

TABLE I: Variables used for Clustering Solutions

Variable	Description
Source IP	The source IP address of the network communication. Represented as the integer value for clustering.
Destination IP	The source IP addressed of the network communication. Represented as the integer value for clustering.
Source Port	The source port of the communication.
Destination Port	The destination Port of the communication.
Source Bytes	The number of bytes sent from the source to the destination.
Destination Bytes	The number of bytes sent from the destination to the source in response.
Protocol ID	The protocol used for the communication.
Timestamp	The time at which the NetFlow was detected.

For analysis, Table 2 outlines the NetFlow attributes used for analysis within the clustering system. While source and destination IP of network traffic was monitored, only those pertaining to internal devices were used for the distance-based calculations within the system. Timestamps were also gathered for informative use only, and not used during the distance based calculations. Processing steps taken during the variable normalization process are described below.

1) Axial Scaling.

Typically, variables used during clustering are normalized within the scale of 0 to 1. However, processing steps were taken during this normalization process in this instance to ensure that the relative metrics were of importance. For example, when processing the port values within the NetFlow, the first 1024 registered ports are more valuable than dynamic ports in terms of information gain. Generally, dynamic ports provide little to no information gain on the type of activity being conducted, with a few notable exceptions. Port 80 specifically describes http traffic for example, while port 50000 does not describe any traffic type in particular. To represent this within the clustering mechanism, the port variables were not scaled from 0 to 1 in a linear fashion. The first 1024 ports were scaled between 0 and 0.5, ports 1025 – 12000 were scaled between 0.51 to 0.9, and ports 12000 plus were scaled between 0.91 to 1.

These scaling values were used to ensure that any port between 0 and 1024 would carry enough distance within the clustering configurations in order to “break out”, ensuring that a unique cluster would be created between adjacent values, such as 80 and 81, where the difference of 1 interval carries a much greater meaning than that between 50000 and 50001, where no additional information can be gained.

2) IP Address Processing

While performing distance-based calculations against all IP addresses would be invaluable, to perform the clustering task in a real time application, this research limited specific analysis of IP address to those internal to NPD. These IP addresses were similarly scaled to the ports mentioned in section 1. The aim is to again increase the distance from each adjacent IP address to ensure that each internal IP would generate a new cluster on its own merits.

All external IP addresses (those not on the 10.0.0.0 network in this case) were limited to the space between 0.9 and 1 of normalized space. This compression of feature space ensures that a unique external IP address will not generate a new cluster if the only change in activity is the external IP address involved. This performance saving measure does however come with degradations in the monitoring capability of the clustering algorithm, as all activity from internal devices to external services through port 80 are now considered equal.

B. Image Generation

As described in [14],deviations in monitored clusters may be indicative of an anomaly occurring. For example, an increase in login attempts made between two devices may be indicative of a possible attack. While the clustering mechanism can detect new assets attempting to login though the generation of a new cluster, a device that commonly logs in will create no new

cluster and will therefore be considered as normal behavior regardless of how many attempts are made. For this paper, we propose that the generation of images based upon the NetFlow previously classified by the clustering mechanism.

The aim is to represent the clusters density (number of NetFlow instances classified) and the consistency (the attributes of the contributing NetFlow to a cluster) of each cluster within time intervals used during configuration. For example, a Windows end-point may send between 1 and 5 DNS requests to a server in a 5-minute interval. If the user changes the DNS server it uses for lookups, the clustering mechanism will detect this and create a new cluster. However, if the DNS request rate increases dramatically, it will not. The SOM generated image will change to reflect this increase in activity. It is this change that this research aims to detect in the final step of the proposed framework.

This process is inherently costly in terms of performance impact. For a single image to be generated, for a particular cluster, NetFlow inputs on a minute to minute biases range from 1 input to over 7000. This image generation performance problem is discussed further in Section IV, B.

C. Image Classification

For this paper, a novel convolutional neural network is proposed. The aim of this network, which is the final step of analysis, is to detect and classify changes within individual clusters. With a SOM image as input, 2D2N is streamlined for improved performance. For example, convolutional layers may be streamlined as to avoid searching the entire image for a match. A single NetFlow will always be represented in the same position on the image; therefore, a search through the image is not necessary. Figure 2 outlines 2D2N. The aim of this step is to:

1. Detect a change in a clusters composition.
2. Classify this change dynamically as a numerical state.
3. Rate this numerical state with respect to time, enabling the measurement of how common a deviation in activity within a cluster occurs.

This dynamic classification of the cluster is key to the approach taken in this paper. The aim is to generically classify shifts in cluster composition with relation to time. For example, DNS systems use port 53 as a destination port when indexing assets on a network. However, the source port for that communications can be anywhere between 1041 and 65535, observed during testing. So, while areas of the image may shift in density (color) in this regard, the destination port will always remain 53. If the destination port is not 53 however, then, this shift will be treated differently (assigned a different numerical class) than a shift in source port.

An input image, Img_{test}, consisting of x_r, x_g, x_b, the rgb values of a pixel within the image, obtain the pixel wise distance, $\sum_p = |x_{img_{test}} - x_{L_n}|$ between the image and the relevant convolutional layer of the 2D2N network (x_{L_n}) gained from training. If the total pixel wise distance from the test image and the convolutional layer is great enough to generate a new output class, the layers with the greatest distance are stored and added to the convolutional pool. This enables the future detection of similar instances.

By using an entropy function correlated with the class output of 2D2N, the result is occurred then put into context of the commonality of the incident. For example, a change in cluster composition that happens relatively frequently, the resulting output will reflect this through a lower anomaly score. However, if a new shift is detected, a larger anomaly score is given. Section V demonstrates this on a live network.

Algorithm 1 2D2N

Input: $Img_{test} = \{x_1, x_2, ...x_n\}$ where $x_n = \{x_r, x_g, x_b\}$
Output: C_{anom} where $C =$ a class of anomaly.
  for $(x_n \in Img_{test})$ **do**
    obtain relevant convolutional layer L_n
    $d_n(Img_{test}, L_n) = \sum_p |x_{img_{test}} - x_{L_n|}$
    where d_n = pixel-wise distance of $x_{img_{test}}$ to x_{L_n}
    if $(max_{arg}(d_n.w_c) \neq C_{!anom}$ **then**
      where w_c = expected weight for class c
      if $(C_{anom} = C_{new})$ **then**
        $w_{c_{anom}} = \{d_1, d_2, d_3, ...d_n\}$ for all n
        for $(L_n \neq Img_{test}.x_n)$ **do**
          $Img_{test}.x_n \in L_n$ is added to convolutional layer pool.
      else
        return C_{anom}
    else
      return $C_{!anom}$

Figure 2: Dynamic Degenerative Neural Network

IV. Framework Configuration and Training

This section describes the configuration of all the steps outlined in Section III. For testing, a total of 6 weeks' worth of live data, gathered from within the NPD Groups Inc. network, was processed. In total, over six billion NetFlow samples were analyzed, covering 17797 unique hosts detected on the network over that time. This resulted in a total of 2018094 clusters generated by the clustering algorithm using the parameters and methodology set out in this section.

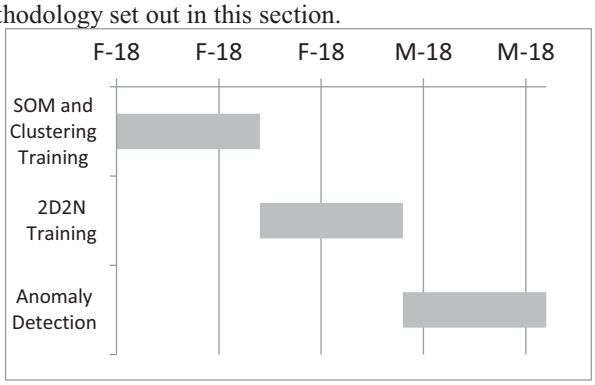

Figure 3: Training Workflow

A. Clustering Parameters

Table 1 outlines the parameters used for this step of NetFlow processing. Point anomalies and clusters generated form the clustering process were stored. Initial training of the clustering mechanism was undertaken over a 2-week period (Table II).

TABLE II Clustering Configurations

Variable	Value
Minimum Density Required	2
Maximum range for Sample	.0025
Size of Window	60s

As outlined in Section III, an axial scaling technique was applied to ensure specific attributes would ensure that a new cluster would be generated when observed.

1) IP Addresses

If the IP address is internal (on the 10.0.0.0/8 network in this case) the IP was converted from its IPv4 representation to its integer format. Then it was scaled between 0 and 9 instead of 0 and 1.

2) Ports

For analysis, ports were not normalized in a linear fashion. If a port, x, table III outlines the scales used for various port values.

Table III: Axial Scaling of Port Values

Port Value (x)	Scaling
$0 \leq x \leq 1024$	$0 \leq x \leq 0.5$
$1025 \leq x \leq 12000$	$0.51 \leq x \leq 0.9$
$12000 \leq x \leq 65535$	$0.91 \leq x \leq 1$

These values were chosen due to the relative information gain they provide about a specific application causing the network activity.

B. Image Generation Perameters

Image generation for the reference SOM images outline in Section III occurred after the first 2 weeks of training on the clustering mechanism. This was done to find established clusters within the network that represent activity that is relatively common on the network. While the clustering layer of the proposed architecture is conducted in real time on the network, the self-organizing maps were trained in an offline capacity. Training was established immediately after the clusters were generated and continued until a stopping measure (Table IV) was successfully reached. Following this training, all NetFlow samples that were classified by the targeted cluster were "binned" into the SOM, ensuring that all traffic was successfully represented in the image. A total of 500 maps were trained, selected at random out of the 2 million clusters generated over the course of testing. Details of the maps are given in Section V.

TABLE IV: SOM Image Training Parameters

Variable	Value
Initial Weights	Random Values between variable limits.
Learning Rate	Windrow-Hoff Method
Iterations	2880
Distance Measure	Euclidean
Stopping Measure	Learning Rate < 0.0001
Image Size	50*50 Pixels

C. 2D2N Configuration

For the training of 2D2N, the SOM was used to organize the data from the middle 2-week section of out 6-week testing period (Table II). NetFlow Data contained within the previously identified clusters were used as input into the SOM generated for that specific cluster. Using this, 2D2N was initially trained as a binary neural network, where all input images were equal to the same class. This step was taken to establish the weights of the initial convolutional layer so that all images would equal the first output class of 2D2N, which will be taken as "normal activity".

TABLE V: 2D2N Configuration

Variable	Value
Input Layer	5*5*4
Hidden Layer	200
Entropy	0.0025
Output Layers	2 during training, N during classification testing.

Following this, all activity for the remaining 2 week period was classified autonomously by the 2D2N network. Any and all deviations in the SOM detected that were above the pixel-wise distance observed during training were classified as a new output. Section V demonstrates the results of this test. Following classification, the output class weighting was then modified through the entropy function. This demonstrates that an activity is either new or repeating consistently. For the purposes of testing, focus was placed on the clusters that presented the maximum values of deviation from the training period in Section V.

V. Experimental Results

Through the 6 week testing period, over 6 billion individual NetFlow samples were analyzed. In total, over 2021786 clusters were generated, and 2134098 point anomalies detected.

A. Clustering

1) Point Anomalies

Point anomalies, or NetFlow samples that do not meet the minimum density requirements, were monitored over the course of testing. Figure 4 outlines the total point anomalies found within a 1-minute window over time. Initially, as expected, point anomalies were expectedly high during initial training. However, after week 3, it began to normalize.

Both durations of high point anomalies, outlined above, pertain to two particular assets on the network. They were not malicious, however were notable. The two assets in question are responsible for scanning the internal network to detect vulnerabilities located on the assets within the company [15]. When a scan is scheduled, the asset scans all other assets on the targeted network for vulnerabilities. The two areas noted above are responsible for the scanning of two independent networks. As an anomaly, they fall into the category of not malicious, but of note. Spikes in point anomalies not attributable to these two assets are discussed in the next section.

2) Clusters Generated

Spikes generated from point anomalies are commonly associated with spikes in the number of clusters generated during the associated time interval. Figure 5 outlines the rise in number of clusters generated over the course of the 6-week testing period. Sudden rises in the number of clusters are also contributable to the spikes seen in Figure 4. These sudden rises, again, are correlated with the activity of the two assets responsible for conducting vulnerability scanning within NPD. It is worth noting, of the 2021786 clusters generated through testing, 1205948 clusters are attributable to these two assets.

This is expected, as when scanning an asset, all ports are scanned on the target asset. When conducted on a regional network, where hundreds of assets, including end-points and servers, are located, large amounts of clusters are expected to be generated.

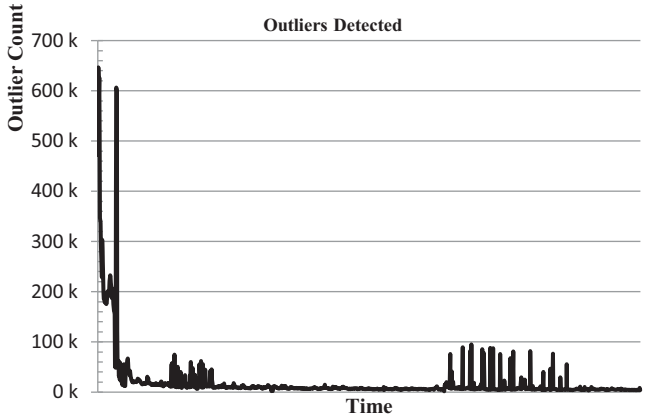

Figure 4: Outlier Count over Testing Period

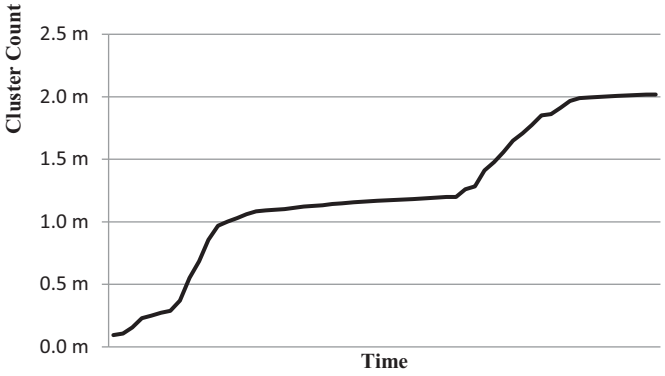

Figure 5: Count of Clusters Generated During Testing

Of note however, are the spikes in point anomalies and cluster generation that are not attributable to these assets. Table VI outlines such cases. In all cases, no previous assumptions were known about the data analyzed.

Clusters A, B and C referenced in Table VI pertain to the same incident. In this case, a windows device located in one remote site started communicating with three other windows-based assets located in a different remote site for the first time. From a security perspective, this is typical of malicious activity propagating through a network. In this case however, after investigation, it was established that the windows machine in question was requesting an update to the McAfee antivirus software located on the device. Up until this point, the device was updated from a known server that pushes McAfee updates to all assets on a local network. However, in this case, McAfee attempted to update through the use of a new peer-to-peer update service that was enabled on the McAfee license. While not malicious activity, this detection was of importance as it immediately made aware the fact that this feature was now in use in some areas of the organization.

Sample E pertained to an asset performing new http activity to external IP addresses. While this kind of activity is extremely common in any organization, what generated this cluster was the source of the communications, Source B. Source B was a server within the organization that, until detected, never performed http activity. In this instance, it was discovered that, during maintenance, a technician briefly opened an internet browser whose homepage was MSN. No other activity occurred, and the browser was closed immediately after opening. Detections such as these, while again, not malicious, provide great insight into activity on the network. Internet access on the server was blocked to prevent further incidents.

Table VI: Sample Clusters Generated Throughout Testing

ID	Source	Destination	Source Port	Dest. Port	Protocol
A	Source A	Destination A	Dynamic	8081	TCP
B	Source A	Destination B	Dynamic	8081	TCP
C	Source A	Destination C	Dynamic	8081	TCP
D	Source B	Destination D	Dynamic	Dynamic	TCP
E	Source C	External Destination A	Dynamic	80	TCP
F	Source D	External Destination B	Dynamic	53	TCP

Within Sample F, a source connected to a wireless network within NPD changed his DNS settings on the device. This bypassed the internal DNS used within the company. Such actions are against security policy, and the owner was made aware as soon as this cluster was detected.

B. 2D2N Classificaion

Sample D was a cluster created during the first hour of training within the clustering mechanism. It pertained to a windows-based asset connecting to a domain server, using Kerberos based authentication. This type of activity is incredibly common among organizational assets, as Kerberos authentication attempts may be made many times per minute. No discernable deviations in activity were detected through either the clustering mechanism, through point anomaly detection or cluster generation, during testing.

However, during the final two weeks of testing, 2D2N classification of the cluster changed. Table VII describes the classification of the cluster over a 15 minute interval where the class of the image changes. The classification changed from a class 1, which is the class that the original SOM image was trained on, to a class 2, a never before seen change in cluster composition. Upon investigation, it was discovered that Source B was the source of a relay attack to the authentication server over a fifteen-minute period. This was corroborated through the detection of a dictionary based attack from rules-based systems currently in place within NPD. However, what is interesting is that during the attack, no noticeable increase in Kerberos activity (commonly associated with port 88) took place. The rules based system detected this incident though log process, which is outside the scope of this investigation.

Table VII: 2D2N Classification of Cluster over Multiple Time Intervals

Output Variable	Intervals (I minute)																	
Output Class	1	1	1	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2**	1	1	1	1
Output Strength	0.91	0.93	0.73	0.61	0.76	0.87	0.96	0.95	0.98	0.96	0.92	0.91	0.84	0.76	0.79	0.91	0.92	.0.94

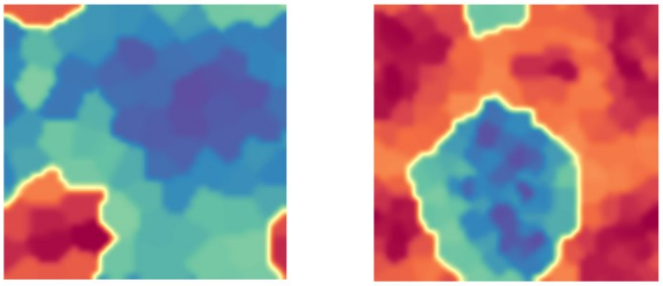

Figure 6: Class1 image (left) vs class 2 image detected (right)

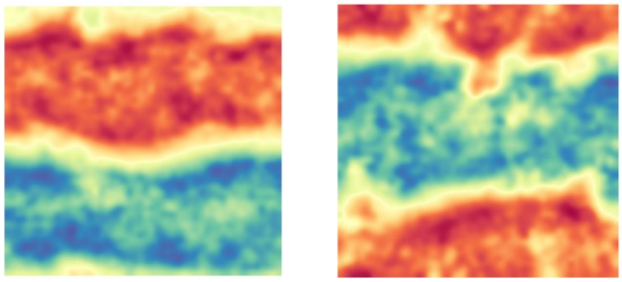

Figure 7: Both images classified as normal for Erratic Cluster

The change in this image is a result of abnormal traffic over port 49159. As noted in section IV, dynamic ports contain relatively little information about the activity taking place. In this case, the shift from relatively little dynamic port activity between two assets to a high amount was indicative of an attack. Figure 6 shows the images of each class for the cluster involved. A clear difference can be seen. Where as in figure 7, a similar change had taken place in another cluster over the same period of time. However, no new class was created, as the variation in this cluster was expected.

VI. Conclusions

In this paper, we proposed the use of 2D2N, a novel convolutional neural network designed to detect divergences in normal patterns of network activity. Through the generation of images on pre-clustered data, normality of intra-cluster activity can be processed and represented as an image. 2D2N has proven capable of detecting deviations in this granular data that was indicative of an attack.

A big repercussion of using such a system is performance. While the clustering mechanism can be used in real time, processing over 60000 samples in under 20 seconds, the image generation portion is computationally expensive, particularly when considering the amount of clustered needed to be mapped. While this was done in an offline manner for the purpose of this test, results have proven that the optimization of image generation techniques, possibly though the use of specialized hardware, it may become more efficient.

References

[1] J. Asmuss and G. Lauks, "Network traffic classification for anomaly detection fuzzy clustering based approach," in *2015 12th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, 2015, pp. 313–318.

[2] M. Bailey, C. Collins, M. Sinda, and G. Hu, "Intrusion detection using clustering of network traffic flows," in *2017 18th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, 2017, pp. 615–620.

[3] K. Nalavade and B. B. Meshram, "Evaluation of K-Means Clustering for Effective Intrusion Detection and Prevention in Massive Network Traffic Data," *Int. J. Comput. Appl.*, vol. 96, no. 7, pp. 9–14, Jun. 2014.

[4] Haviluddin *et al.*, "Modelling of network traffic usage using self-organizing maps techniques," in *2016 2nd International Conference on Science in Information Technology (ICSITech)*, 2016, pp. 334–338.

[5] A. H. Almutairi and N. T. Abdelmajeed, "Innovative signature based intrusion detection system: Parallel processing and minimized database," in *2017 International Conference on the Frontiers and Advances in Data Science (FADS)*, 2017, pp. 114–119.

[6] B. I. Santoso, M. R. S. Idrus, and I. P. Gunawan, "Designing Network Intrusion and Detection System using signature-based method for protecting OpenStack private cloud," in *2016 6th International Annual Engineering Seminar (InAES)*, 2016, pp. 61–66.

[7] W. Zhao, G. Chen, and X. Xu, "AnySCAN: An Efficient Anytime Framework with Active Learning for Large-Scale Network Clustering," in *2017 IEEE International Conference on Data Mining (ICDM)*, 2017, pp. 665–674.

[8] T. Sun, Y. Liu, and J. Chen, "A dynamic network anomaly detection method based on trend analysis," in *2017 3rd IEEE International Conference on Computer and Communications (ICCC)*, 2017, pp. 405–411.

[9] C. Callegari, M. Pagano, S. Giordano, and F. Berizzi, "CUSUM-based and entropy-based network anomaly detection: An experimental comparison," in *2017 8th International Conference on the Network of the Future (NOF)*, 2017, pp. 132–134.

[10] G. Kathareios, A. Anghel, A. Mate, R. Clauberg, and M. Gusat, "Catch It If You Can: Real-Time Network Anomaly Detection with Low False Alarm Rates," in *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2017, pp. 924–929.

[11] F. Azzedin, H. Suwad, and Z. Alyafeai, "Countermeasureing Zero Day Attacks: Asset-Based Approach," in *2017 International Conference on High Performance Computing Simulation (HPCS)*, 2017, pp. 854–857.

[12] C. Schon, N. Adams, and M. Evangelou, "Clustering and monitoring edge behaviour in enterprise network traffic," in *2017 IEEE International Conference on Intelligence and Security Informatics (ISI)*, 2017, pp. 31–36.

[13] J. Ekberg, J. Ylinen, and P. Loula, "Network behaviour anomaly detection using Holt-Winters algorithm," in *2011 International Conference for Internet Technology and Secured Transactions*, 2011, pp. 627–631.

[14] K. Flanagan, E. Fallon, A. Awad, and P. Connolly, "Self-configuring NetFlow anomaly detection using cluster density analysis," in *2017 19th International Conference on Advanced Communication Technology (ICACT)*, 2017, pp. 421–427.

[15] "Top Rated Vulnerability Management Software | Rapid7." [Online]. Available: https://www.rapid7.com/products/nexpose/. [Accessed: 24-Apr-2018].