A Machine Learning Approach to network Security Classification utilizing NetFlow Data

John Watkins
Department of Electrical and Computer Engineering
Naval Postgraduate School
Monterey, California, USA
john.watkins@nps.edu

Murali Tummala
Department of Electrical and Computer Engineering
Naval Postgraduate School
Monterey, California, USA
mtummala@nps.edu

John McEachen
Department of Electrical and Computer Engineering
Naval Postgraduate School
Monterey, California, USA
mceachen@nps.edu

2021 15th International Conference on Signal Processing and Communication Systems (ICSPCS) | 978-1-6654-3699-1/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICSPCS53099.2021.9660294

Abstract— **As an increasing amount of information is passed over physical and wireless networks, opportunities and attempts to steal that information are on the rise. The ever-growing volume of digital information has resulted in a corresponding increase in network traffic to accommodate information flow. This creates an opportunity for adversaries to monitor massive amount of network traffic and steal vital information many times being detected too late or not at all. All computer network traffic can be associated with a specific signature based on a given feature set within its metadata information. Establishing a baseline of normal secure network characteristics provides an observer the ability to determine any deviations from baseline operations and make an educated decision as to the relative security status of the network at any given time. These deviations from normalcy are considered anomalies and could identify unsecure practices within a network exposing significant vulnerabilities to potential malicious actors. The focus of this paper is to develop a machine learning approach to classify and analyze metadata within network traffic to determine the characteristics of a network and the level and degree of secure practices within.**

Keywords— Machine Learning, Deep Learning, Concept Drift, Intrusion Detection, Computer Network Security

I. Introduction

The focus of this work is to develop a machine learning approach for classifying and analyzing metadata within network traffic to determine the characteristics of a network and the level and degree of secure practices within. By training a machine learning algorithm to identify specific feature sets of different network flows, the ability to classify a network as either secure, moderately secure, or unsecure with a high level of accuracy significantly increases. By constructing a dynamic flow of information through the proposed scheme, a determination as to the security status of the network can be made. The data is then stored and combined with the previously known dataset and the model is retrained to establish a new baseline. It is then applied to new incoming data thereby increasing the accuracy and providing a near real time assessment as to the security status of the network.

The objective of this paper is to implement a machine learning algorithm to classify the security status of a computer network based on metadata within its network flows. First, the metadata must be preprocessed into a format that is acceptable for the chosen machine learning algorithm. It is critical that all errors in collection of the data be removed, and the remaining data be prepared for input to the respective machine learning algorithm. Netflow is the primary means of metadata collection, and individual flows are used as input into the models. The prepared flows are then first input into an unsupervised clustering model to identify and extract the relevant classes associated with each cluster within the dataset. Following the development of classes, labels are assigned to the respective flows, and the data is again preprocessed for input into a deep neural network model. The model is then trained and tested on the known dataset, then applied to the newly collected dataset for the purpose of network security classification.

The development of neural networks and their ability to identify hidden features within data have made them ideal for image classification problems. Tiwari discusses the use of stacked convolutional layers used for feature extraction and a fully connected layer for classification [1]. The convolutional layers have an added benefit of feature reduction which is especially useful when dealing with higher resolution digital images. When analyzing the feature space of the input images, it became apparent that this approach could be used with computer network traffic as well. The variety of feature sets that can be extracted from Netflow data can be modeled as feature vectors as is done in Tiwari's paper [1] for classification. The preprocessing of the data is significantly different, but the implementation of the classification model is relatively similar.

Neural networks are now a commonly used tool for network intrusion detection systems and an effort to maximize their effectiveness has led to a great deal of research on the topic. Mohammadpour et al [2] discuss a novel concept for the identification of network intrusions through the use of a convolutional neural networks for feature extraction followed by a fully connected neural network for classification. The author used a pre-labeled dataset of network intrusions and thus did not require an unsupervised learning model. However, the results for feature set selection and the model for classification became the base for the proposed methodology in this work.

Direct capture of computer network traffic is not always feasible, which necessitates the use of more readily accessible metadata that can be found in computer network flow information. Kim and Farhan [3], [4] demonstrate the concept that access to features available within flow-based attacks can identify, with high levels of accuracy, network intrusions of varying types. These feature sets became the basis for the classification of security status of computer networks for this work with respect to the temporal, volumetric, and a myriad of other features associated with network flows.

The remainder of this paper is structured as follows. Section II provides a background of computer network traffic information, deep learning in both supervised and unsupervised implementations, and a detailed discussion of clustering and classification algorithms. Section III presents the details and process for the proposed continuous training methodology, an introduction to the datasets used in this work, and an explanation of the specific machine learning algorithms implemented. Section IV details the analysis and results of the different machine learning models and the levels of accuracy associated with each. Finally, section V concludes the paper with the most significant results and recommendations for future work.

II. Network Traffic Information and Machine Learning Models

Prior to discussing the proposed methodology, it is necessary to discuss the basics concepts of network traffic information and deep learning models used in this paper. It is important to understand how Netflow data is used during data preprocessing and how the computer port information plays a vital role in the classification process. Additionally, a basic understanding of how different clustering and classification algorithms work will facilitate a clearer understanding of the results and analysis that will follow.

A. *Network Traffic information*

A computer network is a group of computers that communicate using a common set of protocols over various mediums for the purpose of sharing resources and information between them. The information shared between these computers is referred to as network traffic and is composed of various amounts of data moving across the network at any given point in time via various computer ports. These ports serve as the interface between computers and play a large role in the security of a network by either permitting or blocking information flow. Computer network traffic can be captured and monitored several different ways. One of the most common and effect ways to monitor networks is to use Netflow protocol which is a protocol developed by Cisco that collects information about all the traffic running through a Netflow-enabled device, records traffic data, and helps discover traffic patterns [5]. As seen in Figure 1 there are three main components to the network that are critical to creating and processing Netflow data: The exporter, collector, and analyzer. The exporter keeps track of the packets moving in and out and creates records to be sent to the collector. The collector then stores the reports and sends them to the analyzer which is an application that can analyze the records for specific information such as anomaly detection [5].

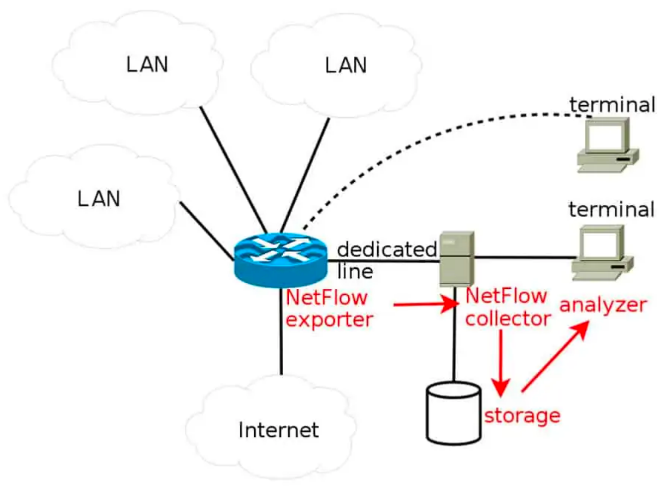


Fig. 1. Netflow Collection Diagram [6]

The Internet Assigned Numbers Authority (IANA) is the recognized entity which controls the assignment of internet protocol resources and port numbers among several other things [6]. These standardized port assignments are used as a basis for the determination of security status within specific network identifiers.

1) *Well known ports*

IANA assigns port numbers based on three different categories. Port numbers between 0-1023 are considered "Well Known" ports. These ports are both assigned to specific agencies and controlled by IANA. Port numbers between 1024-49151 are considered "Registered" ports and are not assigned or controlled but are registered. Port numbers between 49152-65535 are considered "Dynamic or Private" ports and are neither assigned, controlled, or registered [7]. With the advent of new technologies and the increasing demand for information the range of well-known ports has started to expand resulting in regular network traffic flowing through ports as high as 10000. The most frequently accessed well-known ports within the datasets can be seen in Table 1.

TABLE I. SOME WELL-KNOWN PORTS WITHIN THE DATASETS

Port	Description
0	Wildcard port that tells the system to find a suitable port number
22	Secure Shell (SSH)
23	Telnet
25	Simple Mail Transfer Protocol (SMTP)
53	Domain Name Service queries (DNS)
80	Hypertext Transfer Protocol (HTTP)
443	Hypertext Transfer Protocol Secure (HTTPS)
500	Internet Key Exchange (IKE)

2) *Secure Ports*

The increase in sensitive information being transferred over computer networks necessitated the creation of secure ports to protect data as it flows between devices. The assignment of secure ports became standard to ensure some form of data authentication and encryption as it is transferred across a network. Table 2 details the different secure ports that are used in this work and the propose of each.

TABLE II. COMMONLY USED SECURE PORTS

Port	Description
22	SSH (Secure Shell)
443	HTTPS (Hyper Transfer Protocol Secure over TLS/SSL)
465	SMTPS (Simple Mail Transfer Protocol Secure over TLS/SSL
500	ISAKMP (Internet Security Association and Key Management Protocol)
563	NNTPS (Network News Transfer Protocol Secure over TLS/SSL)
636	LDAP (Lightweight Directory Access Protocol over TLS/SSL)
989	FTPS (File Transfer Protocol Secure)
990	FTPS Control
993	IMAPS (Internet Message Access Protocol Secure over TLS/SSL)
994	IRCS (Internet Relay Chat Secure over TLS/SSL)
995	POP3S (Post Office Protocol 3 Secure over TLS/SSL)

B. *Deep Learning: Supervised Versus Unsupervised*

Within the field of machine learning there are two predominant methods for training and testing models: supervised and unsupervised learning. Each method has its own strengths and weaknesses but the main difference between the methods is that supervised learning requires a labeled dataset which means it has a priori knowledge of the classification of a sample. This allows the model to determine relationships between features of a sample associated with a given label in order to make more accurate predictions on unlabeled data [8]. It can then be inferred that a dataset presented to a supervised model of larger size and more robust feature set representations will inherently lead to a more accurate prediction due to the increased amount of information it has access to. Supervised learning is primarily used for the purposes of classification or regression analysis.

Unsupervised learning on the other hand is a method by which a model is presented data that is unlabeled and by developing relationships between different feature sets within the data, an estimate as to the structure of the data is made [8]. The most common use for this method is clustering. This provides a means for dimensionality reduction of higher dimensional data which makes it possible to accurately represent the data with lower dimensional models. Additionally, unsupervised learning is useful for the development of labels associated with different clusters which can be used for classification. A detailed description of the different clustering and classification methods used in this work is below.

1) *k-means Clustering*

k-means clustering is one of the most popular methods for unsupervised clustering available mostly because of the simplicity of the method. The primary objective of the model is to group similar samples together based on identified feature sets within the data as can be seen in Figure 2. The variable "k" is a user defined parameter that establishes the number of centroids to be found within the dataset. The centroid is a variable representing the center (or mean) of a cluster of data. Every point within the dataset is assigned to the nearest centroid thereby creating the desired number of clusters [9]. A noteworthy downfall of the k-means algorithm is that it does not account for noise in the dataset in that every point is assigned to a cluster regardless of whether it may be an outlier or not. The process begins by randomly assigning centroid values and works iteratively to assign each data point to one of the "k" clusters. A new centroid is then calculated, and all data points are reassigned to the new closest centroid. This process works repetitively until either the centroids have stabilized, or a user defined number of iterations has been conducted.

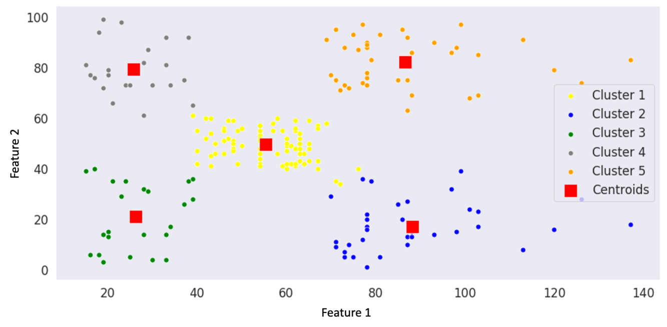


Fig. 2.   Example of k-means Clustering with five clusters [10]

2) *Gaussian Mixture Models*

Gaussian mixture models are similar to the previously discussed k-means method in that both require the user to define a variable "k" for the number of clusters within the dataset. The primary difference between the two is that GMM utilize a Gaussian density model to distinguish between clusters rather than a centroid value as can be seen in Figure 3 [10]. The probability density function of a Gaussian model is given by

$$f(x) = \frac{1}{\sigma_i\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu_i}{\sigma_i}\right)^2} \tag{1}$$

where i = 1, 2, 3. In the example, k = 3, which models the dataset to three different gaussian density functions and assigns each datapoint to the cluster accordingly.

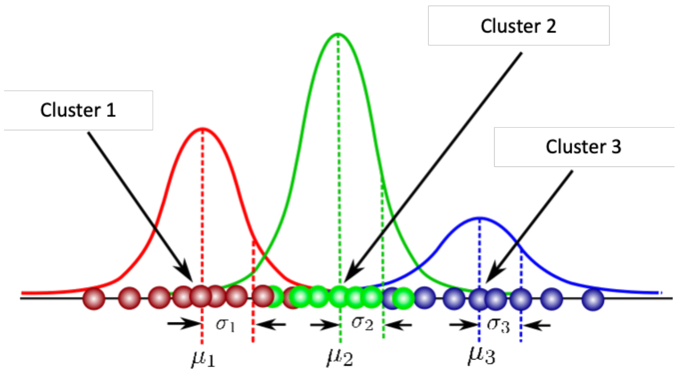


Fig. 3.   Example of GMM Clustering Methodology with Three Clusters [11]

C. *Classification*

k-nearest neighbors (KNN) is a method of machine learning that falls within the supervised learning category. The basic assumption of this algorithm is that similar datapoints will reside closer to each other within a dataset [11]. The model is fairly easy to implement as there is only a single parameter "k" that needs to be chosen. Choosing the correct value of k, the number of nearest neighbors, is important as it is the basis for making classification decisions. As an example, if a k value of five is chosen, the algorithm will take the five closest datapoints to the reference datapoint and make a classification decision based on the most frequently seen label within that set. It is then run iteratively until the entire dataset is explored. It is now evident that although this is an extremely simple means of classification, it is extremely sensitive to the size of data being presented to the model and will be much less efficient for larger datasets.

A neural network is another supervised learning technique that is primarily used for either classification or regression problems. It is modeled after the way the neurons of the human brain function in response to input from the five senses of the human body in order to understand its surroundings. The basic concept of a neural network is that given an input x_i, hidden layers composed of a user defined number of neurons N_n, will decompose the input into different feature sets that can be learned and recognized by a computer. The output value if each individual neuron is calculated by

$$z = \sum_{i=1}^{n} w_i x_i + b \qquad (2)$$

where w_i are the weights associated with the links between neurons as seen in Figure 4, and b is a bias factor that is added to the sum prior to the activation phase. In order to account for the non-linearity of the data an activation function is implemented to perform a non-linear transformation of the results as well as determining the accuracy of the model and ensuring the convergence of the model by finding optimal weights and biases values [12]. This entire process is known as forward propagation and continues throughout all layers in the model until the predicted output $\hat{y}$ is produced. The fine tuning of the weights and biases is then accomplished by a process called backward propagation. The initial values of w and b were arbitrarily chosen as input parameters to the model and to fine tune them; the total error between predicted output $\hat{y}$ and expected output y is calculated using a loss function is given by

$$\lambda(y,\hat{y}) = \sum_{i=1}^{n}(y-\hat{y})^2 \qquad (3)$$

where the goal is to increase the accuracy and minimize the loss associated with the model. This process is conducted iteratively based on the number of epochs, or number of total passes through the model, that are defined as another input parameter by the user.

When the desired accuracy and loss metrics are achieved through training, the model can then be implemented on unknown datasets in which it will receive an input, decompose the input into different feature sets, and make a classification decision based on the relative similarity to the training data. As compared to the KNN classification model this is much more resilient and can handle much larger datasets more efficiently. The model is also much more tailorable to specific problem sets as there are a multitude of parameters and hyperparameters that can be fine-tuned to increase accuracy based on the users desired outcome.

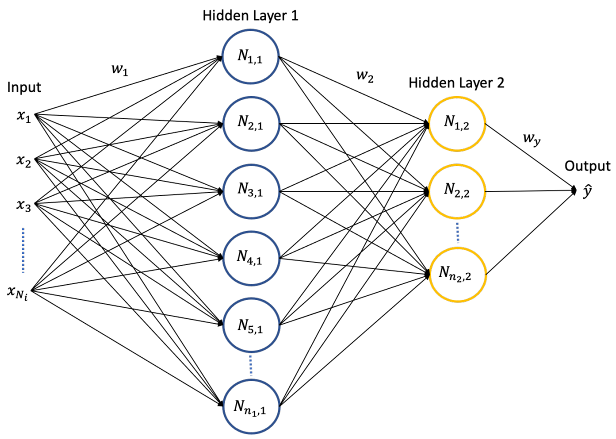


Fig. 4. Example of Fully Connected Neural Network with two hidden layers.

III. Methodology

The objective of section three is to present the proposed continuous training methodology that classifies computer network security status through the implementation of both clustering and deep neural network models. It discusses the critical data preprocessing phase in which network flow information is gathered and network identifiers are identified within the two datasets used for analysis. It also examines how different machine learning algorithms were used for the development of classes and network security classification.

A. Continuous Training Methodology

The proposed methodology takes advantage of a continuous stream of data and the ability for a machine learning algorithm to increase its accuracy based on the amount of data it receives. As seen in Figure 5, the initial dataset is captured and preprocessed into a format that can be input into the clustering machine learning model. Labels are created and appended to the preprocessed data where it is then input into the classification machine learning model to be trained on a desired feature set in order to better classify the output. Once a classification of the NetID is made, the accuracy metric is compared to a desired threshold value and a determination is made as to whether the model should be retrained. When the threshold is exceeded, the new data is then incorporated into the previously available dataset, retrained, and tested on a new unknown dataset in order to make a new classification of the network. This process will run continuously until the user determines it is no longer required.

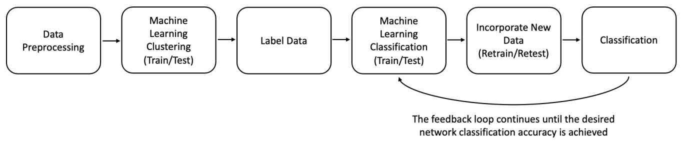

Data Preprocessing
Machine Learning Clustering (Train/Test)
Label Data
Machine Learning Classification (Train/Test)
Incorporate New Data (Retrain/Retest)
Classification
The feedback loop continues until the desired network classification accuracy is achieved

Fig. 5. Continuous Training Methodology Block Diagram

B. Data Preprocessing

Data preprocessing is a critical step in the process that ensures the dataset is void of any inconsistencies and is in an appropriate format that can be input into the model. The two datasets used in this work were both originally pcap files which are commonly used by Wireshark and contain packet data of computer network traffic. In order to simulate the limited access to network traffic information those pcap files were converted in to Netflow data using the nfpcapd tool on a Linux machine. Once converted to Netflow data, the files were transferred back to the Mac workstation and converted to csv files to be manipulated into the proper formats for the machine learning algorithms.

Netflow data when captured provides access to a variety of different metadata features and can be tailored to the specific needs of the user. For the purpose of this paper, the eleven different features seen in Table 3 were extracted for use in the machine learning algorithms. In order to analyze individual network identifiers, the most frequently used source IP addresses were identified and a WhoIs query was conducted to determine the network identifiers. A WhoIs query is a tool that allows a user to input a single IP address and determine a variety of different information associated with the given IP including the network identifier it belongs to. This provided a means to then separate flows by the most frequently used network identifiers for analysis.

TABLE III. NETFLOW FEATURES USED IN MACHINE LEARNING ALGORITHMS

Label	Feature Description
ts	Start Time – Time flow began
te	End Time – Time flow ended
td	Duration – Total time elapsed in milliseconds
sa	Source Address – IP address at source
da	Destination Address – IP address at destination
sp	Source Port
dp	Destination Port
pr	Protocol
flg	Flags – TCP flags associated with a single flow
ipkt	Input Packets – Total packets in a single flow
ibyt	Input Bytes – Total bytes in a single flow

There are two datasets used in this work. The first dataset is a five-minute network capture from the Naval Postgraduate School computer network. This computer network's primary function is to support the university's research efforts and, for the purposes of this work, is assumed to be an example of a secure network. There are two different network identifiers within the dataset and approximately 403,000 flows for analysis.

The MAWI dataset is part of an archive that is hosted by the MAWI working group of the WIDE project. The archive consists of network traffic traces that are intended to be used for the purpose of testing anomaly detection methods on computer network traffic. The database is updated daily in order to incorporate the most up to date applications and network traffic anomalies. In contrast to the NPS network dataset the MAWI dataset is composed of approximately 3.5 million flows with thousands of network identifiers all having varying degrees of security. In an effort to control the amount of information input into the model the top twenty most frequently used network identifiers were used for analysis.

C. *Machine Learning Algorithms*

As discussed previously, the clustering method was used for the development of class labels for each network traffic flow within the dataset. It is important to note that the Netflow data used as input to the model is unlabeled and therefor required an unsupervised learning method. Several different clustering methods were used in order to best determine the classes associated with the unknown dataset presented to the models. Following the clustering of the datasets, classes were determined, and all network flows were labeled accordingly. This labeled dataset then became the input into the classification model.

The development of classes in clustering provides the ability for a determination as to the security status of the network to be made. Two different classification techniques were explored in this work. The KNN model was used for its simplicity in classification and a fully connected neural network was used for its variety of configurable parameters and efficiency with respect to the larger datasets being used. The labeled datasets from the clustering were input into the classification models and a determination as to whether the network was secure, moderately secure, or unsecure was made associated with different levels of accuracy for each. The level of accuracy associated with a classification can then be used as a threshold that is established for a trigger to retrain the network to establish a new baseline.

IV. Simulation Results and Analysis

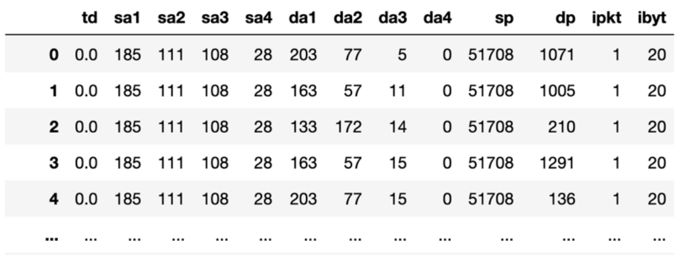
A. *Setup of the simulation environment*

Two different computers were used to complete the data preprocessing and simulations. The primary computer used was an iMac with MacOS Big Sur, a 3.6GHz 8-Core Intel Core i9 processor, and 32GB 2667 MHz DDR4 RAM. This computer was used to do part of the data pre-processing and all computer simulations. The secondary computer used was a Dell Precision T7610, operating on a Linux system which was used for conversion of pcap files to Netflow. The nfcapd tool is only compatible with Linux operating systems thus the requirement for an additional computer.

Several different software tools were used for both data preprocessing and computer simulations throughout this paper. The base platform used for coding was Jupyter Notebook with Python 3 as the programming language. The Keras software library with a TensorFlow 2.0 backbone was used for its artificial neural network tools as well as the Scikit-Learn library for its extensive set of classification, clustering, and regression tools.

B. *NPS and MAWI Data Preprocessing*

Once the pcap files were converted to csv files with Netflow data the files were further preprocessed specific to the machine learning algorithm being used. For the purposes of clustering a comparison between the source port and destination port traffic was used in order to reduce the dimensionality of the data to two dimensions and provide an accurate representation of traffic flow in to and out of the network. Each flow was then reduced to two features and used as input to the clustering algorithm. The classification algorithms are more robust to higher dimensionality and therefor a separate dataset was used with a larger feature set and higher dimensionality. Both the KNN and Neural Net models used seven different features for classification as seen in Figure 6. The source and destination IP address features were then further expanded into octets creating a thirteen-dimension dataset for each flow. This increase in dimensionality provided for a more robust and accurate representation of the network traffic metadata incorporating multiple features for classification.



	td	sa1	sa2	sa3	sa4	da1	da2	da3	da4	sp	dp	ipkt	ibyt
0	0.0	185	111	108	28	203	77	5	0	51708	1071	1	20
1	0.0	185	111	108	28	163	57	11	0	51708	1005	1	20
2	0.0	185	111	108	28	133	172	14	0	51708	210	1	20
3	0.0	185	111	108	28	163	57	15	0	51708	1291	1	20
4	0.0	185	111	108	28	203	77	15	0	51708	136	1	20
...	...	...	...	...	...	...	...	...	...	...	...	...	...

Fig. 6. Sample of classification dataset input with 13 Dimensions

In order to use the dataset for classification purposes it was necessary to split the composite dataset into several different sections in order to train, validate, and test the model. The training data is a significantly larger portion of the dataset and is the mechanism for learning the desired model parameters. A smaller validation set is further segmented from the training set in order to avoid overfitting the model which provides a false sense of accuracy. If the model is overfit, it is only learning the patterns of the provided training set and will not perform accurately when presented unknown data outside of the training set. The test set is the last section of data utilized by the model. It is critical to note that at no point in the training process should the testing data be introduced into the model. The test set is used to determine the accuracy of the model on unknown data and if the data has already been introduced into the algorithm, the testing results will be inaccurate.

The next step in data preprocessing was to standardize the data by using the StandardScalar tool in sklearn. The tool normalizes all data to a standard normal distribution with mean of zero and unit variance. Machine learning algorithms are extremely sensitive to large deviations in data and normalizing the data ensures the algorithm performs as expected. The final step is to then transform the labels into a useable format by the neural network model. The Netflow feature data and the labels were separated into two different variables and the labels were converted via on-hot encoding as seen in Table 4.

TABLE IV. ONE-HOT ENCODING FOR NETFLOW DATA LABELS

Label	Label Description	One-hot Encoded Label
0	Secure	[1,0,0]
1	Moderately Secure	[0,1,0]
2	Unsecure	[0,0,1]

In an effort to both further reduce the dimensionality of the larger datasets and more accurately model the computer networks within the dataset, network flows were combined based on the NetID they belonged to. The source addresses were used as the distinguishing feature and were sorted in order to determine the most frequently used source addresses. Those source addresses were then inputted in to a Who Is query which provided the NetID associated with the IP address. From this analysis, a list of the most frequently used NetIDs was created and the two NetIDs from the NPS dataset as well as the top twenty NetIDs within the MAWI dataset were used for analysis.

C. *Clustering Results*

The results of k-means and GMM clustering methods on both the NPS and MAWI datasets will be discussed in the following subsections. The analysis of both and a discussion of how the results led to the selection of the optimum clustering method is presented followed by how the development of labels for implementation in classification was conducted.

1) NPS Dataset

As k-Means clustering is the simplest model to implement it was the first model used for analysis. The first step was to determine the value of "k" clusters to be used in the algorithm. To do this the elbow method was implemented. The elbow method is a calculation that varies the number of k-clusters from 1-10 and for each value of k, calculates the Within-Cluster Sum of Square value (WCSS) [8]. The WCSS is the sum of squared distance between each data point and the centroid in its respective cluster. As the number of clusters increases, the WCSS value decreases and as can be seen in Figure 7, the point at which the curve most drastically changes direction (the elbow) is the optimum number of clusters. From analysis Figure 7 cluster values of k=3,4, and 5 were chosen for this dataset.

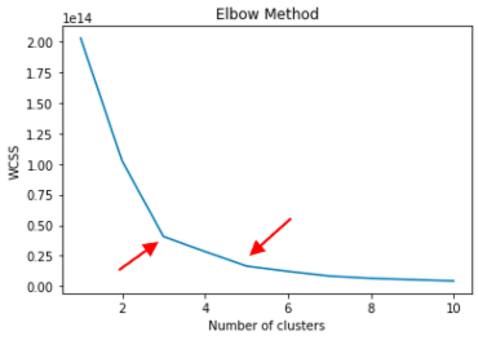


Fig. 7. Elbow Method for Determining Number of K-Clusters

The NPS dataset being the presumed secure network was chosen for analysis first. It can be seen in Figure 8 that the clusters are clearly concentrated below a threshold port value of approximately 10,000 for both the source and destination port features. It is also worth noting that the traffic from lower numbered source ports predominantly flows to lower numbered destination ports and vice versus indicating relatively secure practices. There is minimal traffic in the NPS dataset from higher port numbers in both the source and destination port which is indicative of unsecure practices, and these ports are unregistered and not controlled as previously mentioned. As the number of clusters increases, the only observed effect is the segmentation of the bottom cluster into smaller pieces. Therefore, based on the k-means clustering analysis, a value of k=3 was chosen as optimum, and the number of clusters led to the development of three different classes for input into the classification models.

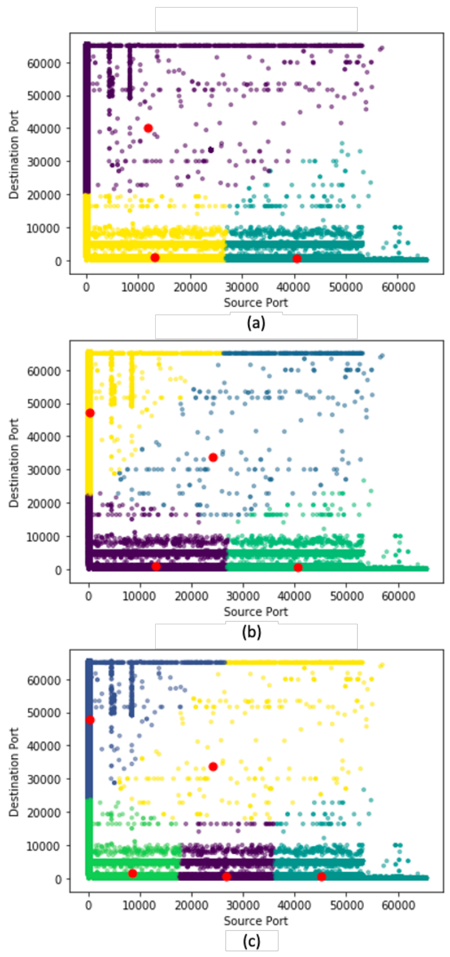


Fig. 8. NPS K-Means Model with n_clusters = 3,4,5

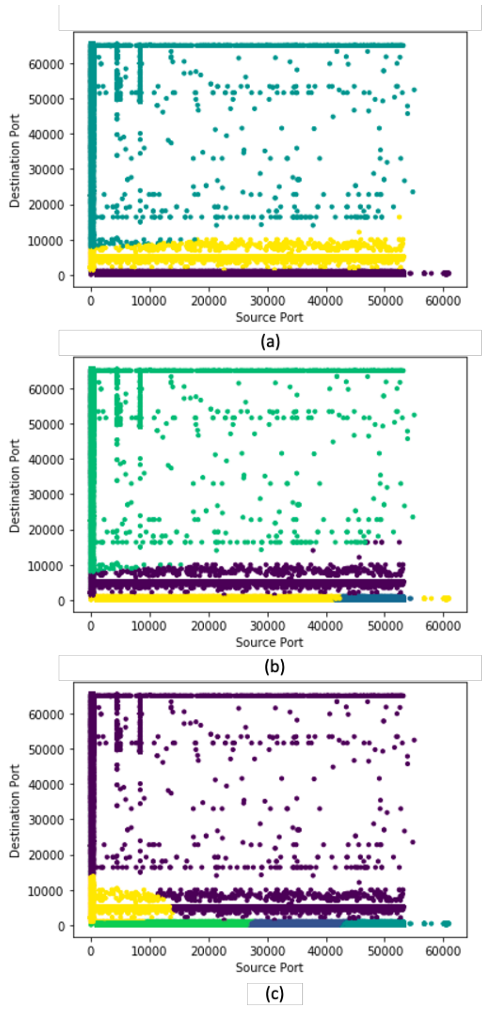


Fig. 9. NPS GMM with N_components = 3,4,5

As a means of comparison several other clustering methods were performed on the NPS dataset in order to confirm the optimum number of classes. As can be seen in Figure 9, when using Gaussian mixture models the data still trends below the 10,000-port value threshold for both the source and destination ports. The lower cluster is again further segmented into smaller pieces as the number of clusters is increased indicating the increase in number of clusters provides no benefit for analysis.

2) *MAWI Dataset*

Based on the analysis of the NPS dataset and the identified 10,000 port value threshold, the K-means clustering algorithm was used for initial analysis of the first four most frequently used network identifiers within the MAWI dataset. As can be seen in Figure 10 there is a varying degree of secure practices as compared to the NPS networks however, the network traffic does appear to be concentrated among certain port values within a given NetID which aided in the classification of the network.

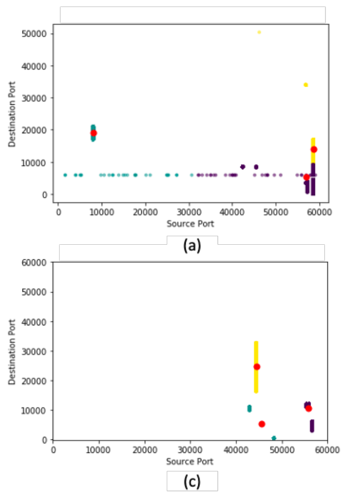


Fig. 10. Four MAWI NetIDs K-Means with n_clusters = 3

The results from the NPS dataset clustering methods determined that K-Means was the most relevant method for clustering the data. It was then determined that further clustering analysis on the MAWI dataset was not required. Based on the results from the k-means analysis on both datasets, a threshold port value of 10,000 was chosen to determine labels to be used in the follow-on neural network classification algorithms. These three labels would be associated with secure, moderately secure, and unsecure classifications.

D. *KNN and Deep Neural Network results*

The results of both KNN and DNN will be discussed in the following subsections. The analysis of both and a discussion of how the results led to the selection of the optimum classification method is presented. This is followed by a detailed discussion on the accuracy metrics associated with each model and how the algorithm was fine tuned to increase accuracy and minimize loss associated with its performance.

1) *Classes of Secure Networks*

In order to successfully classify the security of the network the Netflow data needed to be labeled. Based on the chosen threshold port value of 10,000 from the k-means clustering model, the data was filtered in to three different classes and labeled accordingly. Port values that fall within the eleven different secure ports notated in Table 2 were all labeled "Secure", any port values that were between 0-10,000, exclusive of the secure ports, were labeled "Moderately Secure", and all other port values greater than 10,000 were labeled "Unsecure".

2) *The First KNN Model*

The KNN model, being the simplest classifier used, only required the tuning of one hyperparameter. The number of nearest neighbors was modeled for n_neighbors values of 3,4, and 5. The three models where then built, trained, and tested on the NPS NetID 204 dataset with an accuracy deviation of approximately 1%. The number of nearest neighbors' value of three performed the best with an accuracy score of 93% and was chosen as the model to be used with the expansion of training data and tested on unknown datasets.

3) *The First Neural Network Model*

There is an important distinguishing characteristic between parameters of a machine learning model and the hyperparameters of the model. Model parameters are the aspects that are learned by the algorithm whereas hyperparameters are the user provided inputs to the algorithm that have an effect on the model output and are not learned by the model. The choice of these various hyperparameters will certainly have varying effects on the performance of the model and maximizing their performance is key.

The neural network model utilized several different hyperparameters to take advantage of the flexibility of the model. The initial hyperparameters chosen and their values can be seen in Table 5.

TABLE V. INITIAL HYPERPARAMETERS FOR THE NEURAL NETWORK MODEL

Hyperparameter	Value
Batch Size	128
Epochs	20
Activation Function	relu
Hidden Layers	3,4,5
Hidden units	256
Loss Function	categorical cross entropy
Optimizers	adam

With the initial hyperparameters chosen three different models were created with varying hidden layers and first tested on NPS NetID_204 dataset in order to determine the most favorable number of hidden layers with respect to model performance. The accuracy score associated with the different number of hidden layers varies only slightly by approximately 2%.

The three models were then tested with results seen in Table 6. Without changing the hyperparameters, and with an accuracy score of 78.44%, a value of three hidden layers was chosen for the model.

TABLE VI. TESTING RESULTS OF THE FIRST MODEL

Accuracy Score of the model with 1-Hidden Layer	76.41%
Accuracy Score of the model with 2-Hidden Layer	76.57%
Accuracy Score of the model with 3-Hidden Layer	78.44%

The three hidden layer model was then tested on two additional datasets that were not used during training and completely unknown to the model. The NPS NetID 205 dataset was used for comparison as the datasets are independent but fairly similar and the MAWI NetID 1 dataset was used for a completely independent evaluation of the model on unfamiliar data. As can be seen in Table 7 the performance of the model on NPS NetID 205 dropped by approximately 10% while still performing within an acceptable range of accuracy. The performance on the MAWI dataset however was extremely poor and led to the future expansion of training data to increase accuracy in testing and implementation.

TABLE VII. TESTING RESULTS OF 3 HIDDEN LAYER MODEL ON UNKNOWN DATA

Accuracy Score of 3-Hidden Layer Model on NPS NetID 205	66.28%
Accuracy Score of 3-Hidden Layer Model on MAWI NetID 1	6.44%

4) Expanded Dataset Results

With the results of both the KNN and neural network models performances having been trained on the NPS NetID 204, the next step was to increase the dataset available for training in an effort to increase the accuracy of the model. As a result, the NPS NetID 204 dataset was concatenated with four other NetIDs expanding the total number of flows from 181k to approximately 1.4million. Additionally, a larger unknown dataset was created for testing purposes by concatenating the next six more frequently used NetIDs from the MAWI dataset.

The KNN model was first to be tested with an established number of nearest neighbors' value of three. As expected, the model accuracy score increased in testing to 98.5% when a larger dataset was incorporated into training which is a slight increase as compared to the first model. The model was then tested on the larger unknown dataset and although accuracy increased initially, it significantly decreased to 50.12% when introduced to data from unknown NetIDs. The DNN model with three hidden layers was then trained and tested on the expanded dataset and performed with an accuracy score of 94.89% which is an increase of approximately 16% from the first model.

5) Grid Search Model Results

With an increase in accuracy observed in both models as a result of increasing the training dataset, the next step to increase accuracy was to fine tune the hyperparameters of the model. As the KNN model only requires one hyperparameter this method only applied to the DNN model. The grid search tool from sklearn was used to calculate an accuracy score associated with every possible combination over a specific set of hyperparameters in order to maximize performance. This grid search method was conducted with a single hidden layer DNN model and searched over the values of the batch size, epochs, and activation function hyperparameters seen in Table 8. Over eighty different iterations of possible combinations of hyperparameters the best performing combination was determined to be batch size of 32, 100 epochs, and the relu activation function with an accuracy score of 99.87%.

TABLE VIII. GRID SEARCH HYPERPARAMETER VALUES

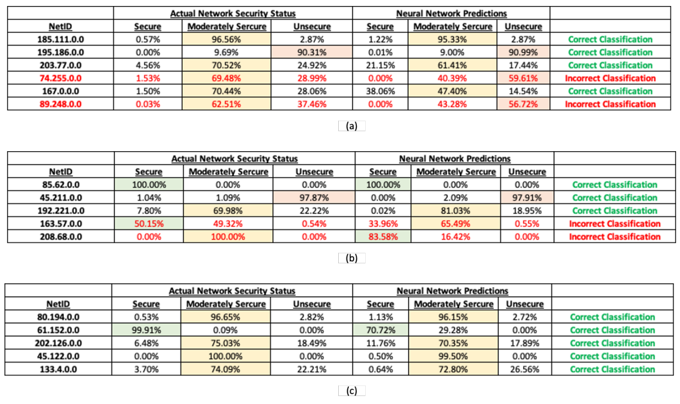
Hyperparameter	Values to be Searched
Batch Size	16, 32, 64, 128, 256
Epoch	10, 20, 50, 100
Activation Function	relu, tanh, sigmoid, linear

The grid search model was then retrained with the new hyperparameters on the expanded dataset to confirm its performance and the model accuracy metric significantly increases while the loss metric quickly approaches zero. The grid search model was then tested on the unknown expanded dataset with MAWI NetIDs 5-10 and performed with an accuracy score of 79.56%. This is approximately a 20% increase in accuracy over the KNN model and resulted in the DNN model being selected as the best method for network security classification.

6) Continuous Training Method Results

With the deep neural network model selected as the best model for classification the next step in the process was to simulate a dynamic flow of network traffic for analysis. This was accomplished by incrementally presenting three rounds of unknown network flows associated with the most frequently used NetIDs within the datasets. Each round of data presented to the model was then further analyzed to determine which of the NetIDs the model was classifying correctly as compared to the labeled data based on the percentage of flows associated with each security class.

The first round of testing on unknown data resulted in an accuracy score of 79.56%. As seen in Figure 11a, the neural network model correctly classified four out of the six unknown NetIDs. The two NetIDs that were incorrectly classified corresponded to networks that were moderately secure and misclassified as unsecure by approximately 20%. By implementing the continuous training methodology to further increase accuracy, the unknown NetIDs were then compiled with the previously known dataset, the model was retrained, and reimplemented for a second round on five additional unknown NetIDs. The second round of testing resulted in an accuracy score of 81.96%, increasing on the previous round by approximately 2.5%. As seen in Figure 11b three of the five NetIDs were correctly classified. The unknown NetIDs were then again compiled with the previously known dataset, the model was retrained, and reimplemented for a third round on another set of five unknown NetIDs. The third round of testing resulted in an accuracy score of 88.64% which is an increase of approximately 6.5% over the second round and a 9% increase overall. As seen in Figure 11c all five NetIDs were correctly classified. By implementing the proposed continuous training methodology there is a noticeable increase in accuracy as seen in Table 9.



	Actual Network Security Status			Neural Network Predictions			
NetID	Secure	Moderately Sercure	Unsecure	Secure	Moderately Sercure	Unsecure	
185.111.0.0	0.57%	96.56%	2.87%	1.22%	95.33%	2.87%	Correct Classification
195.186.0.0	0.00%	9.69%	90.31%	0.01%	9.00%	90.99%	Correct Classification
203.77.0.0	4.56%	70.52%	24.92%	21.15%	61.41%	17.44%	Correct Classification
74.255.0.0	1.53%	69.48%	28.99%	0.00%	40.39%	59.61%	Incorrect Classification
167.0.0.0	1.50%	70.44%	28.06%	38.06%	47.40%	14.54%	Correct Classification
89.248.0.0	0.03%	62.51%	37.46%	0.00%	43.28%	56.72%	Incorrect Classification

(a)

	Actual Network Security Status			Neural Network Predictions			
NetID	Secure	Moderately Sercure	Unsecure	Secure	Moderately Sercure	Unsecure	
85.62.0.0	100.00%	0.00%	0.00%	100.00%	0.00%	0.00%	Correct Classification
45.211.0.0	1.04%	1.09%	97.87%	0.00%	2.09%	97.91%	Correct Classification
192.221.0.0	7.80%	69.98%	22.22%	0.02%	81.03%	18.95%	Correct Classification
163.57.0.0	50.15%	49.32%	0.54%	33.96%	65.49%	0.55%	Incorrect Classification
208.68.0.0	0.00%	100.00%	0.00%	83.58%	16.42%	0.00%	Incorrect Classification

(b)

	Actual Network Security Status			Neural Network Predictions			
NetID	Secure	Moderately Sercure	Unsecure	Secure	Moderately Sercure	Unsecure	
80.194.0.0	0.53%	96.65%	2.82%	1.13%	96.15%	2.72%	Correct Classification
61.152.0.0	99.91%	0.09%	0.00%	70.72%	29.28%	0.00%	Correct Classification
202.126.0.0	6.48%	75.03%	18.49%	11.76%	70.35%	17.89%	Correct Classification
45.122.0.0	0.00%	100.00%	0.00%	0.50%	99.50%	0.00%	Correct Classification
133.4.0.0	3.70%	74.09%	22.21%	0.64%	72.80%	26.56%	Correct Classification

(c)

a. First Round of Results, b. Second Round of Results, c. Third Round of Results

Fig. 11. Classification results for rounds 1,2, and 3 of the simulated dynamic flow of unknown network traffic by percentage of flows associated with each security class

TABLE IX. ACCURACY OF OPTIMIZED NEURAL NETWORK MODEL TESTED ON A SIMULATED DYNAMIC FLOW OF UNKNOWN NETWORK DATA

	Accuracy Score
Round 1	79.56%
Round 2	81.96%
Round 3	88.64%

After an observed increase in accuracy of classification with the previously discussed model, an additional test was conducted to further prove the continuous training methodology results. This consisted of an additional three rounds of simulations with the NetIDs being presented to the model in reverse order. This simulated the ability for the model to learn from the data being presented in any order as long as the designated feature sets were available. As can be seen in Table 10, there is still a noticeable increase in accuracy between all three rounds with an overall increase in accuracy of approximately 23%.

TABLE X. ACCURACY OF OPTIMIZED NEURAL NETWORK MODEL TESTED ON A SIMULATED DYNAMIC FLOW OF UNKNOWN NETWORK DATA IN REVERSE ORDER

	Accuracy Score
Round 1	69.15%
Round 2	80.77%
Round 3	92.36%

V. Conclusions and Recommendations

This effort focused on the development of a machine learning model to accurately classify the security status of computer networks based on Netflow data. By focusing heavily on the preprocessing phase, eleven different feature sets were identified and extracted from each flow for analysis. This provided a means for feature space reduction while still maintaining a diverse set of features available for analysis. The dataset was first segmented by NetID and further sorted by the most frequently used NetIDs for classification. The k-means clustering method proved to be the most useful in identifying the three clusters associated with the three labels that would be used for classification. These labels where then appended to the network flows and became the basis for input into the neural network model which was fine-tuned and implemented in several different iterations resulting in the accurate classification of computer networks with the highest observed accuracy score of 92.3%.

All previously discussed results have demonstrated the effectiveness of the continuous training methodology with an observed increase in accuracy of approximately 23% over successive iterations of testing. By fine tuning the hyperparameters with the grid search method and continuously expanding the training set while still exposing the model to unknown network traffic during testing, the proposed methodology proves to be significantly more effective in implementation. The use of both supervised and unsupervised machine learning models provides the ability to implement the model on real world computer network traffic without a priori knowledge of its structure or the activities that take place within. The ability to identify the level and degree of secure practices within a computer network using machine learning, based solely on metadata, is a significant advantage to anyone interested in this area of research.

The potential for future work exists in several different areas. Most notably the implementation of the model on a real-world open network for true dynamic network security classification. Although several different machine learning models were tested as part of this effort the list was certainly not exhaustive. The evaluation of additional machine learning techniques would significantly benefit the research and contribute to a more comprehensive analysis of performance

References

[1] V. Tiwari, C. Pandey, A. Dwivedi, and V. Yadav, Image classification using deep neural network, in *2020 2nd International Conference on Advances in Computing, Communication Control and Networking (ICACCCN)*, Greater Noida, India, Dec. 2020, pp. 730–733. [Online]. Available: https://doi.org/10.1109/ICACCCN51052.2020.9362804

[2] L. Mohammadpour, T. C. Ling, C. S. Liew, and C. Y. Chong, A convolutional neural network for network intrusion detection system, in *Proceedings of the Asia-Pacific Advanced Network*, Auckland, Australia, Aug. 2018, pp. 50–55.

[3] J. Kim, J. Kim, H. Kim, M. Shim, and E. Choi, CNN-Based Network Intrusion Detection against Denial-of-Service Attacks, Electronics, vol. 9, no. 6, p. 916, Jun. 2020, doi: 10.3390/electronics9060916.

[4] R. I. Farhan, A. T. Maolood, and N. F. Hassan, Performance analysis of flow-based attacks detection on CSE-CIC-IDS2018 dataset using deep learning, Indones. J. Electr. Eng. Comput. Sci., vol. 20, no. 3, p. 1413, Dec. 2020, doi: 10.11591/ijeecs.v20.i3.pp1413-1418.

[5] Netflow - What is it, a Definition & How to Collect & Analyze Flow Data (sFlow, Ipfix, jFlow, etc), Software Portal, Feb. 16, 2019. https://softwareportal.com/netflow/ (accessed Jul. 19, 2021).

[6] Internet Assigned Numbers Authority. https://www.iana.org/ (accessed Jul. 19, 2021).

[7] Registered Port - an overview | ScienceDirect Topics. https://www.sciencedirect.com/topics/computer-science/registered-port (accessed Jul. 19, 2021).

[8] D. Soni, Supervised vs. Unsupervised Learning, Medium, Jul. 21, 2020. https://towardsdatascience.com/supervised-vs-unsupervised-learning-14f68e32ea8d (accessed Jul. 19, 2021).

[9] Step by Step KMeans Explained in Detail. https://kaggle.com/shrutimechlearn/step-by-step-kmeans-explained-in-detail (accessed Jul. 19, 2021).

[10] O. C. Carrasco, Gaussian Mixture Models Explained, Medium, Feb. 21, 2020. https://towardsdatascience.com/gaussian-mixture-models-explained-6986aaf5a95 (accessed Jul. 19, 2021).

[11] O. Harrison, Machine Learning Basics with the K-Nearest Neighbors Algorithm, Medium, Jul. 14, 2019. https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761 (accessed Jul. 19, 2021).

[12] G. S, An Introduction To Mathematics Behind Neural Networks, Medium, Aug. 04, 2020. https://medium.com/analytics-vidhya/an-introduction-to-mathematics-behind-neural-networks-135df0b85fa1 (accessed Jul. 21, 2021).