applied sciences

MDPI

Article

Image-Based Feature Representation for Insider Threat Classification

R. G. Gayathri, Atul Sajjanhar * and Yong Xiang

School of Information Technology, Deakin University, Geelong, VIC 3217, Australia;
gradhabaigopina@deakin.edu.au (R.G.G.); yong.xiang@deakin.edu.au (Y.X.)
* Correspondence: atuls@deakin.edu.au

Received: 20 May 2020; Accepted: 15 July 2020; Published: 18 July 2020

Abstract: Cybersecurity attacks can arise from internal and external sources. The attacks perpetrated by internal sources are also referred to as insider threats. These are a cause of serious concern to organizations because of the significant damage that can be inflicted by malicious insiders. In this paper, we propose an approach for insider threat classification which is motivated by the effectiveness of pre-trained deep convolutional neural networks (DCNNs) for image classification. In the proposed approach, we extract features from usage patterns of insiders and represent these features as images. Hence, images are used to represent the resource access patterns of the employees within an organization. After construction of images, we use pre-trained DCNNs for anomaly detection, with the aim to identify malicious insiders. Random under sampling is used for reducing the class imbalance issue. The proposed approach is evaluated using the MobileNetV2, VGG19, and ResNet50 pre-trained models, and a benchmark dataset. Experimental results show that the proposed method is effective and outperforms other state-of-the-art methods.

Keywords: cybersecurity; deep learning; insider threat; transfer learning; machine learning; image classification

1. Introduction

Insider threat is one of the most prevalent cybersecurity threats. It refers to potential attacks perpetrated by trusted employees associated with that organization. These employees may have access to sensitive information and resources, which makes it easier for them to orchestrate an attack. There can be many reasons for insider attacks like unintentional human error, conflict with co-workers or managers, and coercion by competing organizations. Insider threats are mainly targeted at violation and sabotage of computer systems and data exfiltration activities. Recent literature vividly prove the significance of this threat and its prevalence. However, there is still no concrete solution to efficiently characterize insider attacks and to facilitate an understanding of the problem and its various components.

The Verizon 2019 Data Breach Investigations Report (DBIR) [1] revealed that there has been a considerable rise in insider incidents. In 2018, the percentage of internal actors involved in overall breaches was 28%, whereas it increased to 34% in 2019. A study on the cost of Cybercrime jointly performed by Accenture and the Ponemon Institute in 2019 [2] revealed that average cost of an insider attack increased by 15% from 2018 to 2019. The Cost of a Data Breach Study in 2019 [3] claimed that the lifecycle of a breach takes 206 days (over 6 months) to first identify it and another 73 days to contain it.

Researchers use data analysis techniques to detect and mitigate security challenges. The use of statistical learning, machine learning, artificial intelligence, and natural language processing techniques has been broadened to solve cybersecurity use cases like detecting malwares and intrusions, phishing, denial of service (DoS) attacks, etc. Security analytics is an approach of data processing which combines acquisition, aggregation and analysis techniques of data for security monitoring and threat detection. Security analytics solutions can be applied on large and diverse datasets using machine learning, deep learning, and artificial intelligence frameworks. Deep Convolutional Neural Networks (DCNNs), one of the most popular deep learning models, proved to be highly efficient in handling visual data, such as images and videos [4].

The challenge of insider actions is that they might only leave a small footprint in the digital audit data because attackers know precisely how and where sensitive data resides and are aware of the security solutions implemented in the organization. This is why certain insider incidents are not revealed for a prolonged period. Effective and efficient data analysis platforms for insider attacks remains an open challenge.

Different insider threat patterns, such as unintentional human errors, misuse of privilege by unsuspecting and malicious users, and cyber espionage, aggravate the complexity of insider threat detection. Inexpensive data storage and increased computing power motivates data scientists to acquire and analyze huge volume of resource access log data. The data used in any learning algorithm for insider threat attack needs to be transformed into a structured format and used for discrepancy or user behavior analysis. In this paper, we propose an image-based feature representation method to depict the behavioral pattern of the employees in an organization. These images are used to detect the anomalous patterns using deep learning models, thereby detecting the insiders.

Our contributions in the work are summarized as follows. First, we represent the resource access patterns of the employees, from a benchmark dataset, as 1D feature vectors and grayscale images. Second, anomaly detection is applied on the 1D feature vectors using deep neural networks and on the images, using deep convolutional neural network to detect unusual usage patterns and to identify the malicious insiders. Third, we compare the effectiveness of our approach with the state-of-the-art deep learning methods available in recent literature.

The remainder of the paper is organized as follows. Section 2 gives an overview of approaches for insider threat detection in recent literature. In Section 3, we explain our proposed approach for insider threat detection. Section 4 discusses the implementation of the proposed approach. Finally, in Section 5, we conclude the paper by presenting a discussion and outlining the scope for future work.

2. Related Work

Researchers have worked on a plethora of solutions for insider threat detection. Ivan Homoliak et al. [5] proposed a novel categorization of the types of insider threat data available with references to existing works and frameworks available in the area of insider detection and analysis. Sanzgiri and Dasgupta [6] classified insider threat detection approaches into nine classes based on the techniques and features utilized in the detection, namely anomaly-based, role-based access control, scenario-based using decoys and honeypots, risk analysis using psychological factors, risk analysis using workflow, improving defense of the network, improving defense by access control, and process control to dissuade insiders. Out of these classes, anomaly-based detection and user behavior analysis are popularly used. User profiling used in behavior analysis helps in bringing out the abnormal user behavior. It is well established as a primary approach in insider threat detection.

Zeadally et al. [7] present a review of solutions for insider threat detection with their advantages and shortcomings. According to the paper, the range of existing techniques are categorized into Intrusion-detection-based approaches, System-call-based approaches, Data-centric approaches, Honeypot approaches, Dynamical-system-theory-based approaches, Anti-indirect-exfiltration approaches, and Visualization approaches.

One of the most popular approaches for insider threat problem is framing the problem as an anomaly detection problem [8]. Chandola et al. [9] contributed a well-studied overview of anomaly detection. In this paper, they state that anomaly detection methods for sequences with multivariate data is still in the budding stage. Machine learning methods can efficiently tackle the anomaly detection problem, and a lot of work has been accomplished in this direction.

Gavai et al. [10] compared supervised classifier approach with an Isolation Forest-based unsupervised approach for detecting insider threat using network logs. This work aggregates details of those features that contribute most to the isolation of a data sample in the tree to clearly understand why a user was tagged as anomalous.

Due to the scarcity of the real-world data, researchers use either data collected through their own data collection processes or synthetic data. The Carnegie Mellon University (CMU) Computer Emergency Response Team (CERT) generated a synthetic dataset mimicking an organization structure and the log files similar to the activities performed by the employees [11].

Liu et al. [12] proposed a method that could unravel the non-linear relationships in log data. An ensemble of deep autoencoders is used to detect malicious insider activities by calculating a score from the error resulting from the original data and the reconstructed data. A model is built from each autoencoder for the input features extracted from each log file. Finally, all the models from the individual autoencoder are combined to build a single model that is used to vividly identify the user's behavior pattern. However, this process is not effective for a dataset from a different source. One of the limitations to this approach is that the feature extraction based on frequency does not always give the expected outcome. Further, the one-hour interval considered for user behavior study is not enough to identify the usage patterns. The experiments were performed using the Carnegie Mellon University (CMU) CERT Insider Threat dataset [11].

Noever [13] tried different families of learning algorithms and concluded that random forest gives the best results with an accuracy of 94%. The experiments were performed using the CMU CERT Insider Threat dataset, and the risk factors were extracted from the data to create the feature vector. They incorporated the sentiment analysis factors from the email and the website content and file access details.

Meng et al. [14] used an approach using Long Short-Term Memory Recurrent Neural Network (LSTM-RNN) and Kernel PCA for the analysis of insiders. The model was built and tested using the CMU CERT Insider Threat dataset v6.2. Performance comparison of the proposed technique was done against popular algorithms, such as SVM and Isolation Forest, but it was not compared with deep learning models.

Lin et al. [15] formulated a hybrid method using Deep Belief Networks (DBN) for feature reconstruction, and One Class SVM (OCSVM) for insider threat detection. In the first step, the features were learned using the DBN model. Then, the multi-domain features were re-learned, and the hidden features were extracted and trained by the first layer that is visible in DBN, followed by each layer of Restricted Boltzmann Machine (RBM). The last layer was set up with the back-propagation network which receives the feature vector generated from RBM and optimizes the parameters of the entire network. Finally, the last layer of the network is replaced with the trained multilayer structure as a feature extractor. These features were fed to OCSVM to train the insider threat detection model. Each day was divided into intervals of fixed time to calculate the activity frequency from the resource log files. The evaluation of the method using CMU CERT Insider Threat dataset achieved an accuracy of 87.79%.

Yuan et al. [16] presented a user behavior anomaly detection-based insider threat detection technique using Deep Neural Network (DNN). User actions sequences were fed to a Long Short-Term Memory (LSTM), which extracts user behavior features and predicts the next user action. A sequence of hidden states of the LSTM model were used to create a feature matrix of fixed-size, which is given to the Convolutional Neural Network (CNN), that it normal or anomaly. The hidden units of the LSTM efficiently captures the temporal behavior patterns. Hence, the temporal dependencies on user action sequence in a long-term were recorded by LSTM. CMU CERT Insider Threat dataset V4.2 was used for experimentation and the results are promising with an area under the curve (AUC) of 0.9449.

Zhang et al. [17] focused on the unsupervised deep learning model DBN. The pre-processing stage included the collection and analysis of the insider behavior logs to extract the behavior feature in the format of a tuple. It included the time of the occurrence of the behaviors, the behavior subject, the host that produces the behaviors, and the specific behaviors. All kinds of logs had the first three items common in it. The fourth item behavior depended on the behavior types and was more difficult to integrate. 1/N code discretization was used on the extracted features for data normalization. The deep learning network model DBN used these features for threat detection. More than one RBM hidden layer with the sigma activation function was used. Back propagation was used to fine tune the network parameters to get an optimized DBN model. CMU CERT Insider Threat dataset was used for method validation.

Chattopadhyay [18] proposed an approach for insider threat detection based on classification of time-series user activities. Features of each day and related statistics were computed to construct the time-series features. Since the dataset is highly imbalanced, a cost-sensitive technique for data adjustment was used to randomly undersample the instances belonging to non-malicious class. A deep autoencoder with two layers was used for the classification. The observations showed that random forest and deep autoencoder classified the time-series feature vectors with high precision, recall and f-score. Using Multilayer perceptron gave a higher recall, but it resulted in a low f-score and precision than the other classifiers.

A behavioral analysis framework (BAIT) was proposed by Azaria et al. [19] for insider threat detection using a semi-supervised classification method that learns from highly imbalanced data. In this work, a one-person game was designed, and 795 players were recruited to play the game on Amazon Mechanical Turk. A few subjects were added to behave maliciously, thereby introducing imbalance. Maliciousness was predicted from the way a player plays the game. The paper explains several variations of the BAIT algorithm, amongst which the best approach gave a precision of only 0.07 and a recall of 0.7. Many works, like Reference [19], exist, in which they propose approaches for insider threat detection based on the behavioral analysis of people in an organization.

Several surveys [20,21] have been reported in the area of insider attacks. Salem et al. [20] mentions about different kinds of insider attacks, including, traitors, and masqueraders. The paper focused on the challenges of insider threat and categorized most popular techniques into Host-based User Profiling, Network-Based Sensors, and Integrated Approaches. One of the main challenges with the research in this direction is the lack of real-world data to study common solutions and general models.

Ferreira et al. [22] proposed a sliding window based approach to the insider threat problem using CERT CMU dataset and the Random Forest algorithm. They used 40% of the users amongst the 2000 users and applied the statistical feature normalization. Random Forest was used for classification, and the results were marked based on precision and F1-score. The main challenge in insider threat analysis is the need for algorithms and approaches that can efficiently identify malicious activities with reduced false negatives and false positives in optimal time [8,23,24].

There are existing works that have effectively applied image classification approaches and deep learning models for malware detection in the domain of cybersecurity. Transfer learning has been applied to malware classification [25] with an accuracy of 98.65%. Kancherla and Mukkamala [26] proposed malware detection by representing the machine code as grayscale images and extracting intensity-based and texture-based features. This approach achieved an accuracy of 95% on a dataset containing 12,000 benign samples and 25,000 malware. Tobiyama et al. [27] developed a malware detection method that uses RNN to perform feature extraction. The extracted features are represented as images, and a CNN is used for classification to malicious or non-malicious, and achieved 96% accuracy. Recently, Bhodia et al. [28] used the image-based transfer learning for malware analysis. There are papers, like Reference [29–31], that proved the effectiveness of image-based deep learning approaches for malware analysis. However, image classification has not been used for insider threat analysis. In the next section, we discuss our proposed method.

3. Proposed Method

In Section 2, the existing deep learning based insider threat detection methods are discussed. We use the resource access patterns of the employees in an organization to identify the major behavioral indicators and use them as features to perform anomaly detection. Usage patterns of employees are represented as 1D feature vectors. These feature vectors are used to create grayscale images. Anomaly detection is performed separately on the 1D feature vectors and on the grayscale images.

We described image classification approaches which are state-of-the-art for malware detection [25–28]. This motivated us to adopt image classification for insider threat detection. It is a novel contribution of this paper. The construction of user-behavior feature vectors, and their transformation into images are underlying steps in the proposed method. It is significantly different from image-based malware detection, in which the binaries are transformed into images with minimal preprocessing. Another novel contribution of the proposed method is the use of transfer learning (TL) for insider threat detection. We used popular TL frameworks to ease the process of designing the DCNN and finding suitable weights. Below, we elaborate the contributions of the paper.

We used the 1D feature vectors to represent the behavior of employees on a daily basis. These feature vectors are used for behavioral analysis. Further, we performed transformation of feature vectors to images which is a significant step in the process of user-behavior representation. It bridges the gap between insider threat analysis and computer vision. Currently, no computer vision techniques are designed for the task of insider threat detection. Hence, to apply image-based analysis, modification to the existing methods or redefinition of the original problem is required. In this paper, we formulate the insider threat analysis problem as a computer vision problem, by creating images from user-behavior feature vectors. The usage logs of an organization contain all the information required to articulate user-behavior. Feature vectors that represent user-behavior are constructed by extracting relevant attributes from the usage logs. The feature vectors constructed from the usage logs are transformed into images. Once the images are created, deep neural network approach for image classification is applied to perform anomaly detection. Any deviation of the behavior from the model is treated as an anomaly.

Transfer Learning (TL) is used in the proposed method to reduce the model complexity. TL is also referred to as inductive learning or domain adaptation. TL has been widely used in image recognition to transfer an already learned model into new tasks and domains. TL allows classifiers to retain their performance on incoming data with new distributions and/or classes by learning a new feature space [32]. TL was introduced to use knowledge from the source domain with sufficient labeled data to help build accurate models in a related but different domain, with small labeled data. This is well-suited for cybersecurity applications because of the limited number of malicious instances, as described in Section 4.2. This capacity of TL has been adopted in cybersecurity systems [33–35]. Juan et al. [34] proposed a feature-based TL approach that improved the classifier performance on NSL-KDD dataset of TCP traffic, illustrating the merit of TL for the cyber security domain. Specifically, we applied feature-based transfer learning, as explained in Section 3.4. In the TL scheme, we borrow knowledge from computer vision and apply to the target domain of insider threat detection. As a result, training time of deep neural networks is reduced, while achieving high classification performance. Figure 1 shows the proposed process flow.

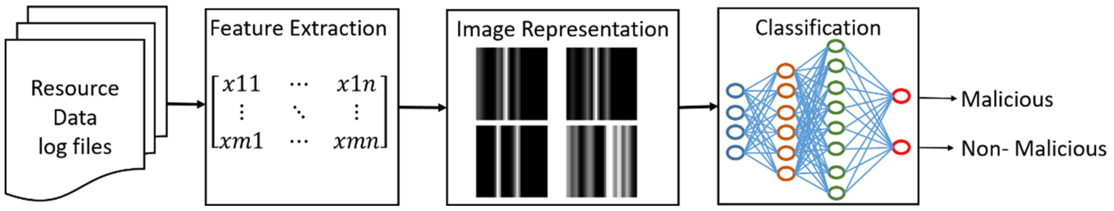

Resource
Data
log files
Feature Extraction
$\begin{bmatrix} x11 & \cdots & x1n \\ \vdots & \ddots & \vdots \\ xm1 & \cdots & xmn \end{bmatrix}$
Image Representation
Classification
Malicious
Non- Malicious

Figure 1. Overall process flow.

First, we extract features from resource usage log files. Second, we present an image-based feature representation approach to map the user behavior. Third, we use deep learning based anomaly detection to detect malicious insiders. Each step is explained in the following sub-sections.

3.1. Feature Vector Construction

In the first step of our proposed approach, user access log pre-processing and feature extraction is performed. In this process, the log files that contain the raw resource access information of employees in the form of rows and columns are extracted to create a feature vector with useful and meaningful features. We used the popular CMU CERT Insider Threat dataset V4.2 [11] for simulation. The dataset comprises of various log files with information regarding the user's logon/logoff details (logon.csv), details of the user browsing history (http.csv), file access patterns (files.csv), external device usage within the organization (device.csv), and email communications sent/received by the user (email.csv). The feature vector is constructed using a frequency-based approach as in Reference [12–15,18]. Figure 2 depicts how the normalized feature vector is created from log files.

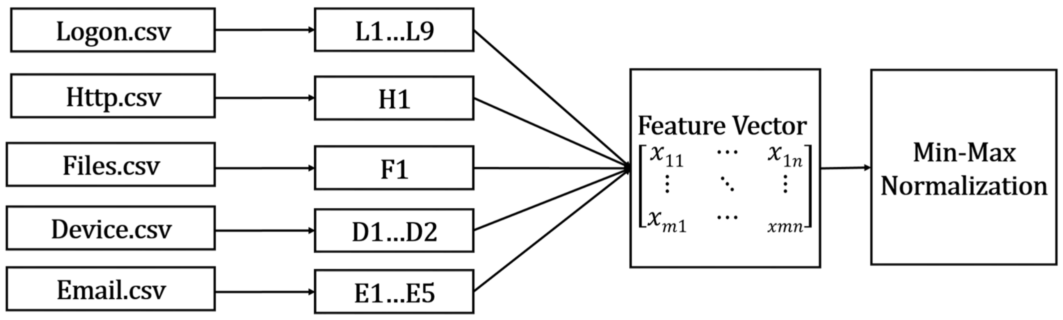

Logon.csv
Http.csv
Files.csv
Device.csv
Email.csv
L1...L9
H1
F1
D1...D2
E1...E5
Feature Vector
$\begin{bmatrix} x_{11} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{m1} & \cdots & xmn \end{bmatrix}$
Min-Max Normalization

Figure 2. Feature extraction from log files.

The logon.csv is used to extract the information about computer usage, such as the session duration, the number of sessions within and outside of office hours, etc. This information provides insights to the computer access patterns of the user. The file.csv gives details about the files copied/read/written by the user on each day. This information is used to analyze the file access patterns of the user. Device.csv log file contains the external device access details of the user on each day. It indicates a user's access to any device within and outside of office hours. Usage of devices outside of office hours may be considered a suspicious activity.

The email.csv file contains information about emails sent/received by the employee within and outside of office hours. It also indicates if the recipient/sender was an external/internal entity. The http.csv provides information about the user's browsing history. The log files described above are used to construct feature vectors. Each feature vector is comprised of 18 features, as shown in Table 1.

In this paper, we use the features used in the paper [18]. The CMU CERT dataset provides the ground truth to label the users as malicious or non-malicious. The ground truth contains the details of user ID, the malicious events that have occurred, and the date/time of the event. There is a list of malicious users with their user IDs and the period in which their activity is marked as suspicious. Separate files are provided for each malicious user with the details of the activity that marked the user as a malicious insider, the timestamp of the event, and the log file.

3.2. Image-Based Feature Vector Representation

The log files from the CMU CERT dataset are pre-processed to obtain the feature vector of each user for each day, as explained in Section 3.1. This represents the resource usage pattern of each user on a daily basis. The features contain numeric values, and need to be normalized to avoid skewing the results. We use Min-Max method to normalize the features in the range 0 to 1. The extracted features are represented as 1D feature vectors. The feature vectors are used for feature representation for user-per-day granularity. Further, the features extracted from the log files are represented as grayscale images. These 1D feature vectors and images are provided as input to the learning model to predict malicious and non-malicious users. The feature vector is represented as a grayscale image, where each pixel is a feature normalized in the range from 0 to 255. The images fall under two categories, namely malicious users and non-malicious users. All the images are of fixed length with same size and maintain the spatial properties which are seen in other greyscale images. Figure 3 depicts the image creation process from CMU CERT data.

Table 1. Feature set used in the proposed approach.

Log file	Features	Description
Login	L1	Difference between office start time and first login time
	L2	Difference between last login time and office end time
	L3	Average difference in time between office start time and number of logins before office hours
	L4	Average difference in time between office end time and number of logins after office hours
	L5	Total number of logins
	L6	Number of logins outside office hours
	L7	Number of systems accessed
	L8	Number of systems used outside office hours
	L9	Average session length outside office hours
Email	E1	Count of emails sent outside the domain of organization
	E2	Count of emails sent within the domain from supervisor's account
	E3	No. of attachments
	E4	Average email size
	E5	Number of recipients
Device	D1	Count of thumb drive usage outside office
	D2	Count of external device usage
File	F1	Number of .exe files downloaded
Http	H1	Count of usage of wikileaks.org

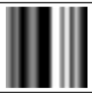
Extracted Features	L1…L9	E1…E5	H1	D1…D2	F1
Normalized Feature Values	0.5, 0.25, 0.394736842, 0, 0.137931034, 0.398118336, 0.389830508, 0, 0, 0.5, 0.4, 0.059496405, 0.807304366, 0.831472448, 0.407746032, 0.80, 0.25, 0				
Normalisation in the range 0-255	154, 74, 121, 3, 38, 123, 118, 0, 1, 154, 74, 16, 247, 255, 123, 246, 78, 0				
Grayscale image					

Figure 3. Image representation of the numerical features extracted from log files.

3.3. Classification

Section 3.2 explained the construction of images which represents the resource usage behavior. In this section, we address the classification of 1D feature vectors and synthesized images as malicious or non-malicious. Deep learning models are applied to perform anomaly detection. Any deviation from the normal user behavior is considered as an anomaly. It is considered as a binary classification and hence the instances are labelled as malicious and non-malicious. Two sets of experiments are performed. First, a deep neural network (DNN) is used for the classification of the feature vectors.

Second, we use the Deep Convolutional Neural Network (DCNN) for the classification of the image data. DCNNs take the unprocessed data as input and transforms the data by processing it through a series of basic computational units to get the representations that have useful values for classification in the higher layers. Figure 4 shows sample images generated using the features given in Table 1.

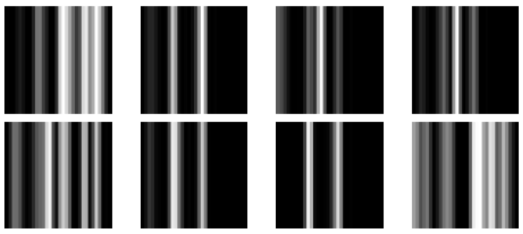


Figure 4. Sample grayscale images.

Deep Convolutional Neural Networks (DCNN) are comprised of neurons with weights and biases that are learnable. DCNN is a sequence of layers, and each layer of a network transforms one volume of activation to another applying a differentiable function. Figure 5 shows the input propagation through various layers of DCNN and how it finally gives a score for each class.

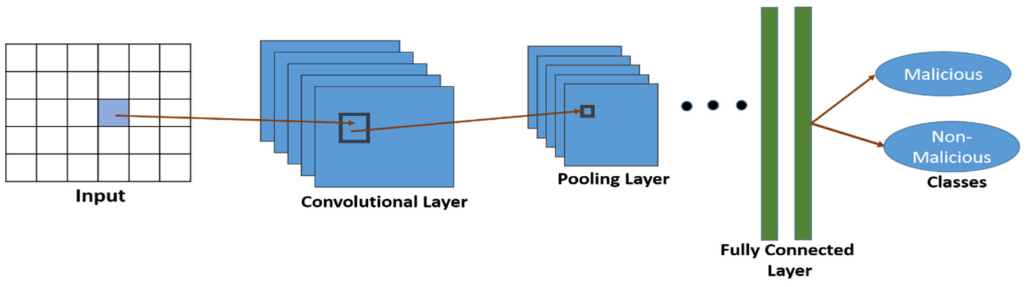


Figure 5. The Convolutional Neural Network (CNN) architecture comprises of a stack of Convolutional layer, Nonlinear layer rectified linear activation unit (ReLU), Pooling layer, and a Loss function (like Softmax) on the last fully connected layer.

The grayscale images are passed to the DCNN to compute the probability of a user being an insider or not. The network does the prediction by computing the probability for each class label. The error in the prediction is calculated using the loss function. Deep Convolutional Neural Network [36] models are popularly used in image classification and recognition. In order to train and test the images, each input image is passed through a series of convolution layers with filters, pooling, and fully connected layers.

Training a deep learning model from scratch requires a very large dataset to avoid overfitting. Moreover, the training time is significant. A pre-trained model converges quickly because the weights are already optimized. Therefore, we propose to use pre-trained models. This is the main motivation for representing the features as images. The pre-trained models available for image classification are more robust than training a new network and can be re-trained to accommodate new classes. The following section presents the details of transfer learning for insider threat detection.

3.4. *Transfer Learning*

Transfer learning [32] is a research direction in machine learning that focuses on retaining knowledge obtained during problem solving and re-applying it to another similar problem.

Transfer learning models can be broadly categorized into four types: instance-based, parameter, feature-representation, and relational-knowledge transfer learning [32]. Amongst these categories, feature-representation transfer learning is most suitable because this approach aims to identify good feature representations that can be utilized across domains. In the proposed approach, pre-trained

CNNs are used to extract image features that are domain-independent features, thereby improving the transferability from source to target domain. A CNN pre-trained on ImageNet is used, where the top layers are replaced with the classification head. There is remarkable reduction in the training time; hence, pre-trained models are preferred where applicable. Transfer learning is mainly used to eliminate the overhead in training. The pre-trained models use two methods to build new models: feature extraction and fine-tuning. In fine-tuning, we can add more layers to accommodate the new classes. We use the fine-tuning method here.

Fine-tuning is the approach followed on a pre-trained model to extend the model so that it suits the need of the new dataset. There is no need to train the model from scratch. This helps in reducing the training time. Here, a few other layers will be trained by updating the weights while training. The training process will tune the weights to match the features in our dataset. Usually the higher layers of the convolutional networks are more specialized. The lower layers learn generic and simple features that can be generalized to most of the images. Moving up the network, the features become more specific to the dataset on which the model is being trained. The main goal of fine-tuning is to use the specialized features with the desired dataset, rather than overwriting the generic learning. Fine-tuning can be applied to increase the performance.

Transfer learning is commonly used with problems on predictive modeling where the input is in the form of images. Each model uses a different image size as input tensor. We scale the images to 32×32 and modified the pre-trained models to accept input images of these dimensions. If the input dimensions are too large, the network might fail to achieve reasonable accuracy as there are not enough layers in the network to learn the features. If the dimensions of the input image is too small, then the neural network naturally reduces volume of dimensions during the forward propagation and then effectively runs out of data.

Our image dataset is dissimilar to the data that is used for pre-training of the original model. Therefore, we perform customization of the network and re-train the model. The proposed approach has two parts: the convolutional base and the classifier head. The convolutional base contains the pre-trained base model. In this stage, the pixels of the input image are converted into features. Then, these features are passed onto the classifier head. The convolutional base is responsible for getting the features from the input image. Once the features are extracted, it is passed to the fully connected layer, and the last layer with the classifier does the prediction in terms of the probability.

The fully connected layers connect all the neurons with each other and combines all the features to get the best prediction. In the fully connected layer, every neuron in the previous layer connects to every neuron in the next layer. The purpose of this layer is to utilize the features from the output of the previous layer to perform classification on the input image-based on the training data. In a fully connected layer, neurons have full connections to all activation in the previous layer similar to the normal neural networks. Fully-connected layer calculates the class probability scores, producing the volume of size $[1 \times 1 \times 2]$, in which both the numbers map to a class score. This class score is used for the classification.

In the proposed problem, there are two classes, Malicious and Non-Malicious. The number of neurons in the last layer of the network should be same as the number of classes to be identified in the dataset. In our last layer, we have two neurons as there are two classes (Malicious and Non-Malicious). The Softmax activation function is used in the final layer to predict the class probabilities.

We used the MobileNetV2 [37], VGG19 [38], and ResNet50 [39] to demonstrate the proposed approach. The base models are instantiated and loaded with the initial weights. Then, the fully connected model is appended to this loaded model and train the network. The details for each model are explained in detail in the following sections. We only fine-tuned the last convolutional block instead of the whole network so that it helps in preventing overfitting. The low-level convolutional blocks learn features that are less abstract and more general than the ones found from higher layers, so it is sensible to keep the first few blocks fixed (more general features) and only fine-tune the last one (more specialized features). Fine-tuning is done with a very slow learning rate to ensure that the updates stay very small, such that it does not affect the previously learned features.

The more layers are frozen, the less effective the network capacity becomes. So, we started from freezing all the layers and only learning the final classifier. Then, we considered unfreezing more layers progressively from the top to achieve a better result. Networks trained on relatively small datasets can overfit the training data. Dropout layers drop out a random set of activations in that layer by setting them to zero. This layer ensures that the network does not get too much fitted to the training data, thereby helping decrease the issue of overfitting. Dropout layer is only used while training. The outputs of a layer under dropout undergo random subsampling. This has the effect of thinning the network or reducing the capacity during training.

Fine tuning the models included addition of a global average pooling layer followed by fully connected layers as the classification layers. In all the three models, we used a Global Average Pooling after the base mode. Unlike the ordinary pooling layer that makes small changes to the images to reduce the size, the global average pooling layer is used to fully or partially replace the fully connected layers used as the highest ones in the convolutional neural networks. The fine-tuning includes the addition of global average pooling layer, which helps to reduce the size of the tensor and speeds up the calculations. The global average pooling layer does a more intense dimensionality reduction, such that a tensor of dimensions h x w x d is reduced to dimensions 1 x 1 x d. It is a simple operation that has no trainable parameters, hence speeding up the training process.

In a classification task, Softmax function is applied to classify an image with probabilistic values ranging between 0 and 1. The summed-up values of a feature map from the global average pooling layer is fed into the Softmax layer. The activation values are added up and divided by the sum produced to give the probability values. Figure 6 gives the details of these models.

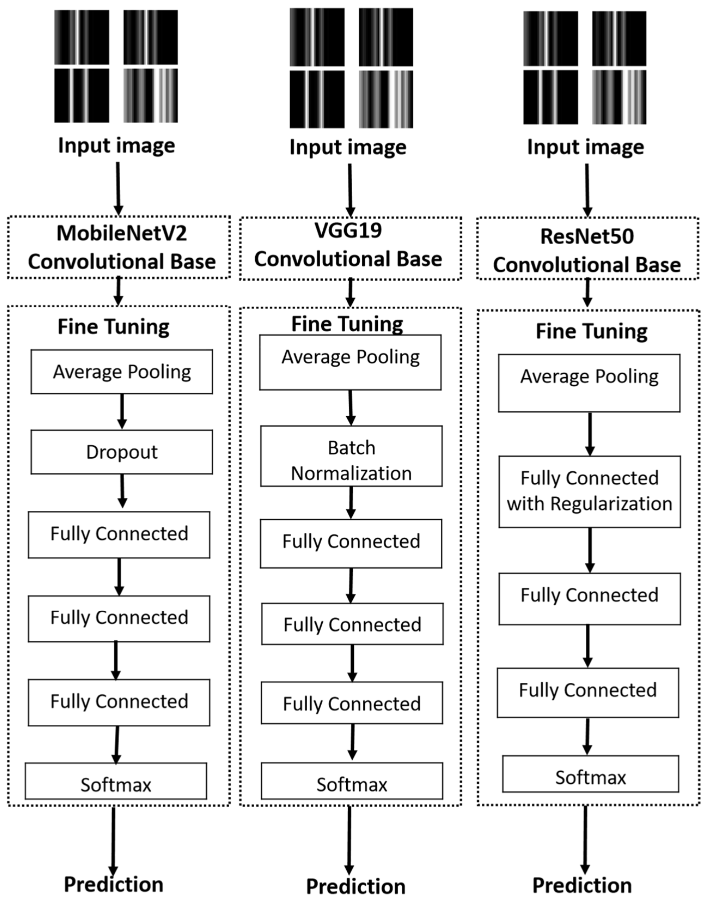


Figure 6. Transfer learning using MobilenetV2, VGG16, and ResNet50.

For the MobilenetV2 [37] model, weights are initialized with the weights used in the training of the original model. The global average pooling layer is followed by a dropout layer. There are three fully connected layers with the number of neurons 512, 256, and 128. When using the VGG19 [38] pre-trained model, we fine-tuned the model by adding an average pooling layer followed by batch normalization. The output features undergo a batch normalization that helps to bring all the activation values to the same scale such that the training speed is improved. In this model, we used the SGD optimizer. The fully connected layers are formed of 1280, 512, and 256 neurons and, finally, the classifier layer.

ResNet50 [39] is also used to train the insider threat dataset. The architecture is similar to VGG, but the size of the model is substantially smaller. This model is fine-tuned by freezing the top layers of the model and adding the fully connected layers followed by classifier layer. Increasing the depth of the network can improve the results as long as overfitting is taken care of. L2 Regularization is applied to avoid overfitting. The fully connected layers are formed of 1024, 512 and 256 neurons and finally the classifier layer.

The weight is initialized with the ImageNet weight used in the training of the original model. Since there are only two class variables, we used the binary cross-entropy logarithmic loss function. In a classification task, Softmax function is applied to classify an image with probabilistic values ranging between 0 and 1. The details of the hyper-parameters are given in Table 2.

Table 2. Hyper-parameters used in the MobilenetV2, VGG19, and ResNet50-based model.

Parameter	MobileNetV2	VGG19	ResNet50
Input shape	(32,32,3)	(32,32,3)	(32,32,3)
Weight	Initialized to ImageNet	Initialized to ImageNet	Initialized to ImageNet
Optimizer	RMSProp	SGD	Adamax
Loss function	Binary cross entropy	Binary cross entropy	Binary cross entropy
Classifier	Softmax	Softmax	Softmax
Epochs	15	15	15
Batch size	64	128	128
Dropout rate	0.3	Nil	Nil
Regularization	Nil	BatchNormalization	L2 Regularization

The proposed models are demonstrated by implementing them as deep learning models and evaluated them using a benchmark dataset. The following section explains the implementation and validation details of the models.

4. Implementation

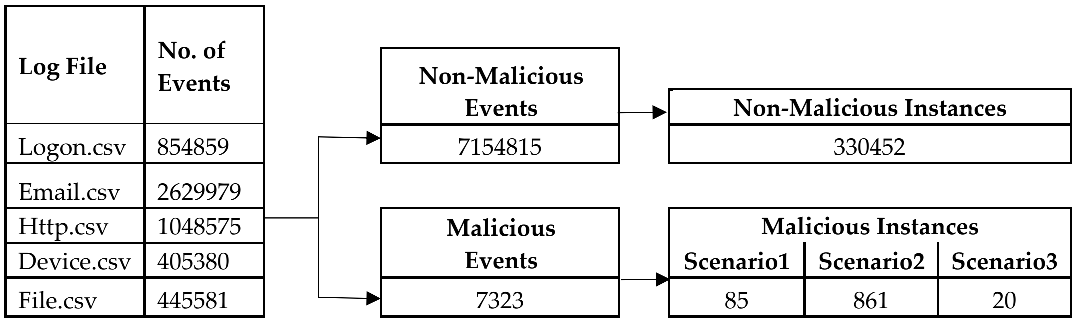
In this section, we present the details about the experiments performed to evaluate the proposed approach. First, the dataset description is given. Second, the class-imbalance problem inherent in the dataset is addressed. Third, we provide experimental results and comparisons with existing approaches. The implementation is done in Keras [40] with Tensorflow [41] backend, an open source machine learning platform from Google.

4.1. Dataset

We used CMU CERT insider threat dataset v4.2 [11] to test the proposed method. Availability of the data is indeed a challenge for insider threat detection because the data from the organizations are sensitive. These data are managed by the data owner and seamless data sharing remains an open challenge. CMU CERT data is a benchmark dataset for insider threat detection which has been widely used by researchers to evaluate their proposed methods [11]. This synthetic dataset generation uses various scenarios to define the malicious activities. The log files contain logon-logoff data, web browsing history, file access logs, emails, device usage, psychometric information, and LDAP data. Each dataset produced contains a small number of insider threat incidents, or scenarios. The malicious activities are categorized into three scenarios in this version. These scenarios were developed in consultation with counterintelligence experts. Each scenario is instantiated as synthetic data in the same form and scope as the normal background data generated for all users in the dataset. The dataset also consists of ground truth information needed to validate the approaches. The CMU CERT dataset is the most popular dataset being used in almost all the recent works in insider threat analysis [42–45].

We used the five event files: logon/logoff activity, http data, email communications, file operations, and usage of external storage device. The events from these log files are fed as input to derive the malicious and non-malicious instances used to create the feature vector and the representative images. Our feature extraction process extracts the various features and summarizes it per day per user to a feature vector of malicious and non-malicious instances. Table 3 gives an overview of the dataset.

Table 3. Dataset details.



Log File	No. of Events
Logon.csv	854859
Email.csv	2629979
Http.csv	1048575
Device.csv	405380
File.csv	445581

Non-Malicious Events
7154815

Non-Malicious Instances
330452

Malicious Events
7323

Malicious Instances		
Scenario1	Scenario2	Scenario3
85	861	20

As seen in Table 3, the number of malicious instances is very small compared with the non-malicious ones. In the next section, we address the data imbalance issue.

4.2. Imbalanced Data Handling

The class imbalance problem [46] is a critical issue in machine learning as there is lack of data in various domains and it has significant impact on the performance evaluation of learning methods modeled with an assumption of a balanced class distribution. Zhou and Liu [46] studied the effect of undersampling, oversampling, and threshold-moving in the training phase of cost-sensitive neural networks. Soft-ensemble and Hard-ensemble, i.e., the combination of oversampling, undersampling, and threshold-moving using voting mechanisms are also experimented. There are recent research works like [47,48] that focus on the anomaly detection in images with skewed class distribution. The main point to be noted is that none of these approaches require any kind of algorithm modification of the neural networks. Table 3 shows that the CMU CERT data is highly imbalanced in terms of the class distribution.

Imbalanced datasets can be dealt with various strategies like adapting the classification algorithms or balancing classes in the training data before providing the data as input to the learning algorithm. The sampling technique is applied to either add more samples to the minority class referred to as oversampling or remove samples from the majority class referred to as undersampling. The main idea is to achieve a better balance in the data sample for all the classes.

Undersampling means randomly selecting a subset of samples from the instances with majority class to avoid its influence from dominating the algorithm learning process. The most common method for doing this is resampling without replacement. In the proposed method, we used undersampling to alleviate the data imbalance. The original data has an imbalance ratio of 1:340 for non-malicious and malicious classes. We used various sampling ratios to undersample the majority class instances. Table 4 shows the data with varying sampling ratios.

Table 4. Sampling ratio.

Non-Malicious Instances	Malicious Instances	Undersampling Ratio
24,150	966	25
19,320	966	20
14,490	966	15
9660	966	10
4830	966	5

4.3. Performance Metrics

The proposed work is tested on the CERT CMU Dataset. Since it is a highly skewed data, we applied random undersampling. The following section shows how each sampling ratio behaves as the number of samples increase in the majority class. This phenomenon is referred to as accuracy paradox, where the majority class tends to be learnt by the model explicitly because of the outnumbered data samples. Hence the classifier gives a very high accuracy. Therefore, we use the precision, recall, and f1-score as the performance metrics. Majority of the existing works use accuracy as the performance metric. Since we compare the proposed approach with other existing approaches, we use accuracy in Section 4.4 to explain the evaluation.

Precision is the ability of a classifier not to label a negative sample ad positive, whereas recall is its ability to find all the positive instances. F1-scores, the harmonic mean of precision, and recall are used to compare the classifiers. These metrics are computed from the classification summary that gives the true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN).

4.4. Experimental Results

The evaluation is performed using the CMU CERT v4.2 data. The features are represented as 1D feature vectors and images, as described in Section 3. Random sampling is performed on the data to overcome the class imbalance problem. Various sampling ratios are used to perform the experiments. We tested on various train-test split ratios. The dataset is split to training and testing before feeding it to the deep learning model. Accuracy, precision, recall, and F-score are used to compare the results.

DNN is used for classification of 1D feature vectors. The DNN consists of 3 hidden layers and a final a classification layer which uses the sigmoid function. A dropout layer is added before the classification layer to reduce the effect of overfitting. Table 5 gives the overall performance of the DNN model for different train-test split of the dataset on the feature vectors. In the table, A, P, F, and R represent accuracy, precision, F1-score, and recall, respectively.

Table 5. Performance evaluation on the benchmark dataset using 1D feature vectors.

Model	**Sample Ratio**	**Training % (70)**				**Training % (80)**			
		A	**P**	**F**	**R**	**A**	**P**	**F**	**R**
DNN	5	90.86	76.04	73.37	70.88	91.72	79.62	72.25	66.12
	10	93.46	64.32	64.81	65.31	93.56	70.24	63.27	57.57
	20	95.57	53.65	52.76	59.87	95.88	67.34	57.39	45.49
	25	96.24	55.10	22.69	14.29	**96.34**	**62.96**	**36.30**	**25.50**

DCNN is used for classification of grayscale images. The weights used in the original pre-trained models are applied but with varying hyper-parameters. Table 6 gives the overall performance of the model for different train-test split of the dataset on the grayscale images. The results show that VGG19 performs better than the other two models.

Table 6. Performance evaluation on the benchmark dataset using grayscale images.

Model	**Sample Ratio**	**Training % (70)**				**Training % (80)**			
		A	**P**	**F**	**R**	**A**	**P**	**F**	**R**
Mobile Net V2	5	75.84	83.83	87.06	90.54	78.60	83.48	87.83	92.65
	10	85.32	89.98	92.83	95.84	87.81	90.86	93.49	96.27
	20	93.38	95.24	96.13	97.09	94.59	95.32	96.21	97.18
	25	94.61	96.21	97.54	98.09	**95.28**	**96.21**	**97.58**	**98.18**
VGG19	5	90.16	94.07	94.10	94.13	90.94	94.80	94.55	94.31
	10	91.21	96.90	95.08	93.34	92.24	96.37	95.65	94.93
	20	95.67	96.60	97.75	98.93	95.17	97.56	96.46	96.36
	25	94.16	96.16	98.78	98.03	**96.34**	**96.80**	**97.12**	**98.59**
ResNet50	5	78.94	80.20	83.66	94.75	79.26	83.99	87.96	92.33
	10	87.45	91.13	93.25	95.47	87.98	91.17	96.16	93.60
	20	89.95	95.29	90.43	85.27	92.28	95.29	96.18	97.15
	25	93.41	95.13	95.54	96.68	**95.31**	**96.12**	**97.43**	**98.09**

Tables 5 and 6 show the performance achieved for the 1D feature vectors and the grayscale images. Though the accuracy of the DNN on the 1D feature vectors is almost same as that of DCNN on image-based analysis, the precision and recall deteriorate for DNN. DCNN gives much improved precision and recall. Hence, the image-based analysis proved to be more promising. The proposed approach is compared with state-of-the-art approaches. Random forest method proposed by Noever [13] gave 94% accuracy with randomization and 90% accuracy with normal Random Forest. Several works use LSTM [14,16] for feature learning, followed by classification algorithms, like CNN. Another approach used deep learning for feature learning and then applied one class SVM [15] for classification. Our proposed approach gave an improved AUC = 97.38 when compared to the method using LSTM [16] gave an AUC = 94.49. Since various papers use different metrics for performance evaluation, we chose the papers with accuracy for the comparison, and those use the CERT CMU v4.2. The difference in the number of features and the representation used, methods applied for class imbalance handling, performance metrics applied, etc., affect the results of each method. Still, we tried to compare the proposed approach with the existing deep learning methods and the CMU CERT dataset. Table 7 shows the performance of the proposed method compared to existing approaches.

Table 7. Performance comparison with existing approaches using the CMU CERT dataset v4.2.

Method	**Accuracy (%)**
Random Forest with Randomization [13]	94.00
Random Forest [13]	90.00
LSTM-RNN [14]	93.85
DBN-OCSVM [15]	87.79
Graph Convolutional Networks [42]	94.50
Proposed Method	96.34

In the next section, we provide the discussion of results and conclude the paper with possible directions for future work.

5. Discussion and Conclusion

The results in the Section 4.4 show that the image-based representation can be effectively applied to the insider threat detection. The dataset has high class imbalance with a ratio of 1:340 for non-malicious and malicious instances. Therefore, we applied random undersampling. In image-based representation, the optimal imbalance ratios are identified as those with high precision, recall, and f1-score. The imbalance ratios from 5 to 25 gave promising results. The precision, recall, and f1-score remain consistent for these ratios. Increasing the sampling ratio beyond 25 causes the precision and recall to drop. This can be handled effectively by using techniques, like random oversampling of malicious instances. Data augmentation approaches can be applied to images to handle the issue of class imbalance [49].

The proposed approach is motivated by the effectiveness of deep learning to learn from very complex mappings between input and output data. Deep learning has proven to be successful in numerous fields like speech recognition, image recognition, and Natural Language Processing. However, the huge number of hidden neurons and layers used in Deep Neural Networks end up in computationally-intensive vector and matrix operations involving millions of parameters that need high performance computing resources. Moreover, it is impractical to collect labeled data samples in many real-world domains to train a network from scratch. In such cases, a pre-trained deep learning model can be used by fine-tuning it for new labels. This motivated us to formulate insider threat detection as a transfer learning problem.

We proposed grayscale images to represent the behavioral features of users which were extracted from resource log files. In order to detect anomalous behavior, these images were used to train and test a pre-trained deep learning model for classification. We used the CMU CERT Insider Threat dataset to evaluate the proposed method. The results presented in Section 4 show the effectiveness of the proposed approach. The temporal aspects of the usage access patterns can be added to the images to achieve an incremental solution approach.

There is vast scope for future work on image-based security analysis. Various other attacks can be formulated as pattern recognition problems to identify anomalies. Additionally, more experiments are needed to better portray the role of image-based analytics in cybersecurity.

Author Contributions: Conceptualization, R.G.G. and A.S.; methodology, R.G.G. and A.S.; software, R.G.G. and A.S.; validation, R.G.G. and A.S.; writing—original draft preparation, R.G.G.; writing—review and editing, A.S. and Y.X; supervision, A.S. and Y.X. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Verizon: 2019 Data Breach Investigations Report. In *Computer Fraud & Security*; Elsevier BV: Oxfordshire, UK, 2019; Volume 2019, p. 4. [CrossRef]
2. Accenture/Ponemon Institute. *The Cost of Cybercrime, Network Security*; Elsevier BV: Amsterdam, The Netherlands, 2019; Volume 2019, p. 4. [CrossRef]
3. IBM. Cost of a Data Breach Report 2019. In *Computer Fraud & Security*; Elsevier BV: Oxfordshire, UK, 2019; Volume 2019, p. 4. [CrossRef]
4. Garcia, A.; Orts-Escolano, S.; Oprea, S.; VillenaMartinez, V.; Martinez-Gonzalez, P.; Garcia-Rodriguez, J. A survey on deep learning techniques for image and video semantic segmentation. *Appl. Soft Comput.* **2018**, *70*, 41–65. [CrossRef]
5. Homoliak, I.; Toffalini, F.; Guarnizo, J.; Elovici, Y.; Ochoa, M. Insight into insiders and it: A survey of insider threat taxonomies, analysis, modeling, and countermeasures. *ACM Comput. Surv. (CSUR)* **2019**, *52*, 30. [CrossRef]
6. Sanzgiri, A.; Dasgupta, D. *Classification of Insider Threat Detection Techniques*; ACM: New York, NY, USA, 2016; Volume 25.
7. Zeadally, S.; Yu, B.; Jeong, D.H.; Liang, L. Detecting insider threats: Solutions and trends. *Inform. Secur. J. Glob. Perspect.* **2012**, *21*, 183–192. [CrossRef]
8. Berman, D.S.; Buczak, A.L.; Chavis, J.S.; Corbett, C.L. A survey of deep learning methods for cyber security. *Information* **2012**, *10*, 122. [CrossRef]

9. Chandola, V.; Banerjee, A.; Kumar, V. Anomaly detection: A survey. *ACM Comput. Surv. (CSUR)* **2009**, *41*, 15. [CrossRef]
10. Gavai, G.; Sricharan, K.; Gunning, D.; Hanley, J.; Singhal, M.; Rolleston, R. Supervised and Unsupervised methods to detect Insider Threat from Enterprise Social and Online Activity Data. *JoWUA* **2015**, *6*, 47–63.
11. Glasser, J.; Lindauer, B. Bridging the gap: A pragmatic approach to generating insider threat data. In *Security and Privacy Workshops*; IEEE: Piscataway, NJ, USA, 2013; pp. 98–104.
12. Liu, L.; De Vel, O.; Chen, C.; Zhang, J.; Xiang, Y. Anomaly-Based Insider Threat Detection Using Deep Autoencoders. In Proceedings of the 2018 IEEE International Conference on Data Mining Workshops (ICDMW) 2018, Singapore, 17–20 November 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 39–48.
13. Noever, D. Classifier Suites for Insider Threat Detection. *arXiv* **2019**, arXiv:1901.10948.
14. Meng, F.; Lou, F.; Fu, Y.; Tian, Z. Deep learning based attribute classification insider threat detection for data security. In Proceedings of the 2018 IEEE Third International Conference on Data Science in Cyberspace (DSC), Guangzhou, China, 18–21 June 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 576–581.
15. Lin, L.; Zhong, S.; Jia, C.; Chen, K. Insider threat detection based on deep belief network feature representation. In Proceedings of the 2017 International Conference on Green Informatics (ICGI), Fuzhou, China, 15–17 August 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 54–59.
16. Yuan, F.; Cao, Y.; Shang, Y.; Liu, Y.; Tan, J.; Fang, B. Insider threat detection with deep neural network. In Proceedings of the 2018 International Conference on Computational Science, Wuxi, China, 11–13 June 2018; Springer: Cham, Swizerland, 2018; pp. 43–54.
17. Zhang, J.; Chen, Y.; Ju, A. Insider threat detection of adaptive optimization DBN for behavior logs. *Turkish J. Electr. Eng. Comput. Sci.* **2018**, *26*, 792–802. [CrossRef]
18. Chattopadhyay, P.; Wang, L.; Tan, Y.P. Scenario-based insider threat detection from cyber activities. *IEEE Trans. Comput. Soc. Syst.* **2018**, *5*, 660–675. [CrossRef]
19. Azaria, A.; Richardson, A.; Kraus, S.; Subrahmanian, V.S. Behavioral analysis of insider threat: A survey and bootstrapped prediction in imbalanced data. *IEEE Trans. Comput. Soc. Syst.* **2014**, *1*, 135–155. [CrossRef]
20. Salem, M.B.; Hershkop, S.; Stolfo, S.J. A survey of insider attack detection research. In *Insider Attack and Cyber Security*; Springer: Boston, MA, USA, 2008; pp. 69–90.
21. Liu, L.; De Vel, O.; Han, Q.L.; Zhang, J.; Xiang, Y. Detecting and preventing cyber insider threats: A survey. *IEEE Commun. Surv. Tutor.* **2018**, *20*, 1397–1417. [CrossRef]
22. Ferreira, P.; Le, D.C.; Zincir-Heywood, N. Exploring Feature Normalization and Temporal Information for Machine Learning Based Insider Threat Detection. In Proceedings of the 2019 15th International Conference on Network and Service Management (CNSM), Halifax, NS, Canada, 21–25 October 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 1–7.
23. Xin, Y.; Kong, L.; Liu, Z.; Chen, Y.; Li, Y.; Zhu, H.; Gao, M.; Hou, H.; Wang, C. Machine learning and deep learning methods for cybersecurity. *IEEE Trans. Knowl. Data Eng.* **2018**, *6*, 35365–35381. [CrossRef]
24. Li, J.H. Cyber security meets artificial intelligence: A survey. *Front. Inform. Technol. Electron. Eng.* **2018**, *19*, 1462–1474. [CrossRef]
25. Rezende, E.; Ruppert, G.; Carvalho, T.; Ramos, F.; De Geus, P. Malicious software classification using transfer learning of resnet-50 deep neural network. In Proceedings of the 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA), Cancun, Mexico, 18–21 December 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 1011–1014.
26. Kancherla, K.; Mukkamala, S. Image visualization based malware detection. In Proceedings of the 2013 IEEE Symposium on Computational Intelligence in Cyber Security (CICS), Singapore, 16–19 April 2013; IEEE: Piscataway, NJ, USA, 2013; pp. 40–44.
27. Tobiyama, S.; Yamaguchi, Y.; Shimada, H.; Ikuse, T.; Yagi, T. Malware detection with deep neural network using process behavior. In Proceedings of the 2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC), Atlanta, GA, USA, 10–14 June 2016; IEEE: Piscataway, NJ, USA, 2016; Volume 2, pp. 577–658.
28. Bhodia, N.; Prajapati, P.; Di Troia, F.; Stamp, M. Transfer Learning for Image-Based Malware Classification. *arXiv* **2019**, arXiv:1903.11551.
29. Lison, P.; Mavroeidis, V. Automatic detection of malware-generated domains with recurrent neural models. *arXiv* **2017**, arXiv:1709.07102.

30. Feng, Z.; Shuo, C.; Xiaochuan, W. Classification for DGA-based malicious domain names with deep learning architectures. In Proceedings of the 2017 Second International Conference on Applied Mathematics and Information Technology, Vellore, India, 26 December 2017; p. 5.
31. Dai, Y.; Li, H.; Qian, Y.; Lu, X. A malware classification method based on memory dump grayscale image. *Digit. Investig.* **2018**, *27*, 30–37. [CrossRef]
32. Pan, S.J.; Yang, Q. A survey on transfer learning. *IEEE Trans. Knowl. Data Eng.* **2009**, *22*, 1345–1359. [CrossRef]
33. Zhao, J.; Shetty, S.; Pan, J.W.; Kamhoua, C.; Kwiat, K. Transfer learning for detecting unknown network attacks. *EURASIP J. Inf. Secur.* **2019**, *2019*, 1. [CrossRef]
34. Zhao, J.; Shetty, S.; Pan, J. Feature-based transfer learning for network security. Proceeding of MILCOM 2017—2017 IEEE Military Communications Conference (MILCOM), Baltimore, MD, USA, 11 December 2017; pp. 17–22.
35. Tan, Z.; Jamdagni, A.; He, X.; Nanda, P.; Liu, R.P.; Hu, J. Detection of denial-of-service attacks based on computer vision techniques. *IEEE Trans. Comput.* **2014**, *64*, 2519–2533. [CrossRef]
36. Yamashita, R.; Nishio, M.; Do, R.K.G.; Togashi, K. Convolutional neural networks: An overview and application in radiology. *Insights Imaging* **2018**, *9*, 611–629. [CrossRef]
37. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.C. Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 4510–4520.
38. Simonyan, K.; Zisserman, A. Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv* **2014**, arXiv:1409.1556.
39. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
40. Chollet, F. Keras: Deep learning library for theano and tensorflow. 2015. Available online: https://github.com/fchollet/keras (accessed on 18 July 2020).
41. Abadi, M.; Barham, P.; Chen, J.; Chen, Z.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Irving, G.; Isard, M.; et al. Tensorflow: A system for large-scale machine learning. In Proceedings of the 12th Symposium on Operating Systems Design and Implementation, Savannah, GA, USA, 2–4 November 2016; pp. 265–283.
42. Zhou, Z.H.; Liu, X.Y. Training cost-sensitive neural networks with methods addressing the class imbalance problem. *IEEE Trans. Knowl. Data Eng.* **2005**, *18*, 63–77. [CrossRef]
43. Piciarelli, C.; Mishra, P.; Foresti, G.L. Image anomaly detection with capsule networks and imbalanced datasets. In Proceedings of the International Conference on Image Analysis and Processing, Trento, Italy, 9–13 September 2019; Springer: Cham, Swizerlands, 2019; pp. 257–267.
44. Jiang, J.; Chen, J.; Gu, T.; Choo, K.K.R.; Liu, C.; Yu, M.; Huang, W.; Mohapatra, P. Anomaly Detection with Graph Convolutional Networks for Insider Threat and Fraud Detection. In Proceedings of the MILCOM 2019–2019 IEEE Military Communications Conference (MILCOM), Norfolk, VA, USA, 12–14 November 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 109–114.
45. Aldairi, M.; Karimi, L.; Joshi, J. A Trust Aware Unsupervised Learning Approach for Insider Threat Detection. In Proceedings of the 2019 IEEE 20th International Conference on Information Reuse and Integration for Data Science (IRI), Los Angeles, CA, USA, 30 July–1 August 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 89–98.
46. Khan, A.Y.; Latif, R.; Latif, S.; Tahir, S.; Batool, G.; Saba, T. *Malicious Insider Attack Detection in IoTs Using Data Analytics*; IEEE: Piscataway, NJ, USA, 2019; Volume 8, pp. 11743–11753.
47. Le, D.C.; Zincir-Heywood, N.; Heywood, M.I. Analyzing data granularity levels for insider threat detection using machine learning. *IEEE Trans. Netw. Serv. Manag.* **2020**, *17*, 30–44. [CrossRef]
48. Perera, P.; Patel, V.M. Learning deep features for one-class classification. *IEEE Trans. Image Process.* **2019**, *28*, 5450–5463. [CrossRef] [PubMed]
49. Vasan, D.; Alazab, M.; Wassan, S.; Naeem, H.; Safaei, B.; Zheng, Q. IMCFN: Image-based malware classification using fine-tuned convolutional neural network architecture. *Comput. Netw.* **2020**, *171*, 107138. [CrossRef]