International Journal of Cognitive Informatics and Natural Intelligence
Volume 14 • Issue 1 • January-March 2020

Detecting DDoS Attacks Using Polyscale Analysis and Deep Learning

Maryam Ghanbari, University of Manitoba, Winnipeg, Canada
Witold Kinsner, University of Manitoba, Winnipeg, Canada

ABSTRACT

Distributed denial-of-service (DDoS) attacks are serious threats to the availability of a smart grid infrastructure services because they can cause massive blackouts. This study describes an anomaly detection method for improving the detection rate of a DDoS attack in a smart grid. This improvement was achieved by increasing the classification of the training and testing phases in a convolutional neural network (CNN). A full version of the variance fractal dimension trajectory (VFDTv2) was used to extract inherent features from the stochastic fractal input data. A discrete wavelet transform (DWT) was applied to the input data and the VFDTv2 to extract significant distinguishing features during data pre-processing. A support vector machine (SVM) was used for data post-processing. The implementation detected the DDoS attack with 87.35% accuracy.

KEYWORDS

Anomaly Detection Method, Convolutional Neural Network, Cyber and Physical Network Security, Discrete Wavelet Transform, Distributed Denial of Service Attack, Variance Fractal Dimension

INTRODUCTION

A smart grid is an innovative electricity delivery system that uses a bidirectional communication network to connect the power providers' control systems and the consumers' smart meters (Yan, Qian, Sharif, & Tipper, 2013), (Beigi Mohammadi, Mišić, Mišić, & Khazaei, 2014). The purposes of the smart grid are (i) to increase the availability and the reliability of electricity, (ii) to control the system in real-time, (iii) to deliver power to users in a safe and secure infrastructure, (iv) to save energy, and (v) to reduce costs. However, hackers can attack the smart grid's cyber layer, which consequently can affect its physical domain. These attacks can disrupt the smart grid's benefits. A *distributed denial of service* (DDoS) attack is a common type of cyber-attack, which delays or blocks the communication in the smart grid, thus causing power outages (Asri & Pranggono, 2015). Cyber space infections can have serious impacts on the real world. A smart grid infrastructure and the *supervisory control and data acquisition* (SCADA) systems, used in power generation, water management and oil pipelines are examples of physical systems that are disrupted by cyber space infections (Nazir, Patel & Patel, 2017), (Asri & Pranggono, 2015). When the operation of physical devices is altered by the attack, the standard cybersecurity problem becomes a cyber-physical security problem. Since the impact of the alteration may also affect the society in a city, or a region, or even a country, the problem escalates to a cyber-physical-social security. Such security systems should be treated using cognitive informatics and cognitive computing (Wang, 2002), (Kinsner, 2012).

Identifying normal burst-data behaviors of a network and the abnormal burst-data behaviors caused by DDoS attacks is very challenging. Both classes of network traffic have similar intrinsic characteristics. They are both stochastic time-series signals, non-periodic, broadband, self-affine, and multi-fractal. Therefore, to differentiate the two classes of traffic, distinguishing features must be extracted. Furthermore, the attack patterns are almost always changing and the new attack patterns

DOI: 10.4018/IJCINI.2020010102

and behaviors must be detected in the smart grid, which is also a frequently-changing environment. Therefore, a learning method that can detect new attack patterns and behaviors in frequently changing environments must be used (Beqiri, 2009). A deep learning algorithm is a good candidate to learn and classify normal behaviors from anomalous behaviors in such an environment (Goodfellow, Bengio & Courville, 2016).

This paper reports on the improvement of an anomaly detection method that was developed previously by Ghanbari, Kinsner, & Ferens (2017). The original detection method had two steps: (i) the pre-processing step that used a *discrete wavelet transform* (DWT) and (ii) the processing step that used a *convolutional neural network* (CNN). To improve the detection rate of the original method, in this study we added a full version of *the variance fractal dimension trajectory* (VFDTv2) to extract features from the non-pure fractal data that rely on long-range dependence as proposed originally by Kinsner (2007, 2012, 2015). The VFDTv2 was adjusted to consider all points including the boundary points of a dataset not just the marginal points. Moreover, the variance equation of the data series was adjusted to consider all points. In addition, the pre-processing step was used to extract more distinguishing features. Also, we added a support vector machine (SVM) as a post-processing step.

Some algorithms have been proposed to detect anomalous behavior in computer networks and smart grid infrastructures. Barford, Kline, Plonka, and Ron (2002) proposed an anomaly detection algorithm based on signal processing. The algorithm derived and summed high frequency, mid frequency and low frequency part of signals. DDoS attacks were detected based on 1.5 and 2 thresholds. However, the length of window and weighted sum for detection attack was static. Hu, Pota, and Guo (2014) offered an anomaly detection based on phase angle, current and voltage in a frame. The algorithm detected DDoS attacks based on unavailability or delaying phase angle. Due to their architecture, there might be instances where the attackers attack the root node, and take advantages of the bottleneck problem. Mohammadi et al. (2014) proposed a hierarchical IDS by considering several rules to detect known and unknown attacks. Khan, Ferens, and Kinsner (2015) offered an anomaly detection method to extract features of the Internet traffic time series by an autonomous sliding windowed the *variance fractal dimension trajectory* (VFDT) to extract the bursts of data sample, accurately. However, the current study applied the machine learning tools to distinguish DDoS attack from normal burst data behaviors with a higher detection rate. The paper that was developed previously reports on anomaly detection method (Ghanbari et al., 2017). The fundamental idea of this method was to enhance the sensitivity of detection by using distinguishing features by the DWT to increase the sensitivity of the CNN. However, the detection rate of 80.77% was not high enough in a smart-grid infrastructure.

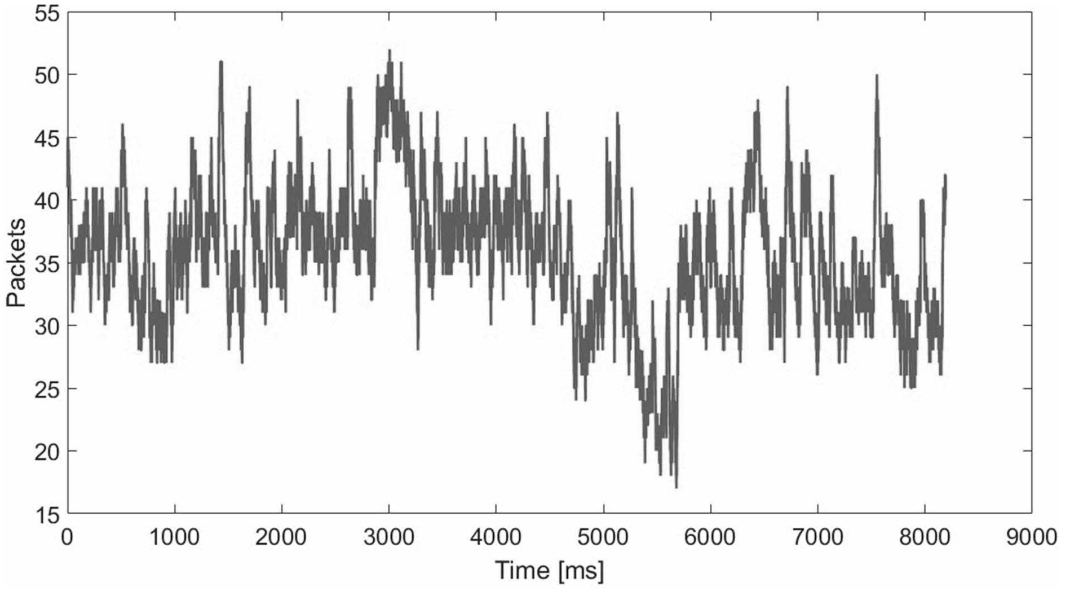
The organization of this paper is as follows. Section 2 presents the dataset. Section 3 presents the proposed an anomaly detection method algorithm. Section 4 presents the simulation. Section 5 presents the results and discussion, and section 6 offers some concluding remarks.

DATASET

The dataset that was used for training and testing in this research was downloaded from the *Center for Applied Internet Data Analysis* (CAIDA). CAIDA obtains different types of real time network traffic from around the world, and it has become one of the most reliable networks for downloading datasets. Commercial organizations, research sectors, and governments donate and collaborate with CAIDA while their privacy is protected (Center for Applied Internet Data Analysis, CAIDA, (2018). This center supports large-scale data collection for the scientific research community.

The dataset used in this research was chosen from CAIDA's 2007 DDoS-attack traffic, and it contained UDP Flood, TCP, ICMP (Ping) Flood, and SYN Flood packets. Each packet in the dataset contains a source IP address, a destination IP address, length of packet, protocol and packet-arrival time. The number of packets with 0.1ms duration within the stationary frame size is shown in Figure 1.

Figure 1. The number of packets within a stationary frame size



The Proposed Anomaly Detection Algorithm

The proposed anomaly detection algorithm consists of four steps: (i) preparing data, (ii) data pre-processing, (ii) data processing, and (iv) data post-processing. These four steps are shown in Figure 2.

The DDoS detection algorithm used in this study is described below.

Preparing Data for Processing

1. The raw data that contains a source IP address, a destination IP address, length of packet, protocol and packet- arrival time for each packet is fetched and tagged as normal or anomalous;
2. A new attribute called a packet arrival time-series signal (ATS) is calculated by using the difference between a packet- arrival time at *t* and the packet-arrival time at *t*-1.

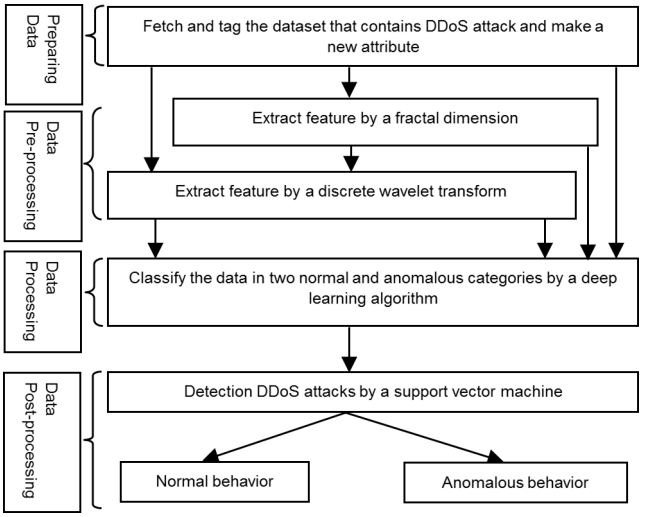
Pre-Processing

The ATS data is passed to the pre-processing step for feature extraction. The features are extracted in two different ways. Firstly, the ATS data is passed to the improved version of the variance fractal dimension (VFDv2) algorithm described in Part A below.

Secondly, the ATS data and the result of the VFDv2 algorithm are passed to the discrete wavelet transform (DWT). The wavelet coefficients of the ATS data and the output of the previous step are calculated by using the Daubechies wavelet transform. The wavelet coefficients can be obtained by convolution of the input signal and the Daubechies basis function within an adjustable window size. This can be achieved using Equation (1) (Gao & Yan, 2011), where *s* is the scaling parameter, τ is shifting parameter, $\psi(.)$ is a discrete basis function and the symbol * denotes the complex conjugate:

$$wt(s,\tau) = \frac{1}{\sqrt{s}} \sum_{n=-\infty}^{+\infty} x(n)\psi^*\left(\frac{n-\tau}{s}\right) \quad (1)$$

Figure 2. The algorithm of DDoS attack detection



In addition, the size of the window is defined in which the data are stationary. The z-score normalization within the window size is used to normalize the output coefficients of the DWT.

Processing

The processing step involves the CNN that was developed and described in detail in Ghanbari et al. (2017). This CNN consists of the following six layers: (i) the input layer, (ii) the convolution layer, (iii) the rectified linear unit (ReLU) layer, (iv) the normalization layer, (v) the pooling layer, and (vi) the fully-connected layer, as shown in Figure 3.

The processing step is as follows:

1. The raw data, the ATS data and the pre-processed data are sent simultaneously to the input layer of the CNN as one-dimensional time-series signals;
2. The CNN is then trained. The logistic regression function is used as a convex cost function. Furthermore, the stochastic gradient descent algorithm is used to train the CNN to update its weights and biases through a backward propagation algorithm;
3. Based on the learned parameters of the training phase, the output of the testing phase classifies the data into two classes: normal and anomalous behaviors.

This single-stage CNN comprises the processing step.

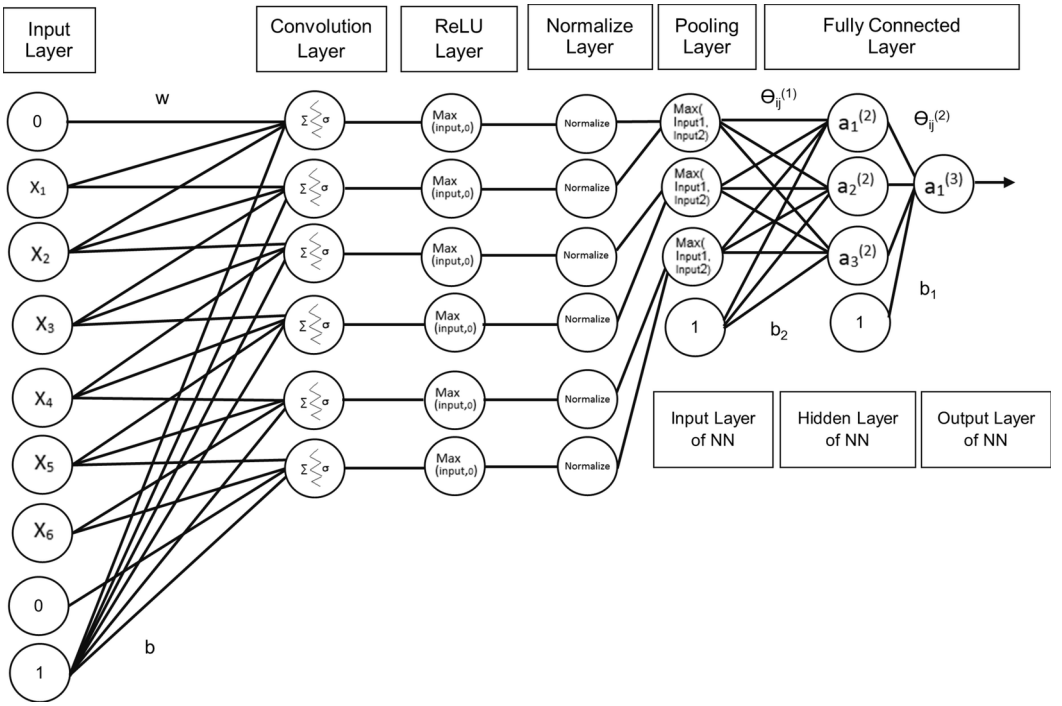
Post-Processing

The post-processing step is used to overcome the duration of training the CNN. In general, a CNN involves a large number of parameters, which decreases the training speed, so this study uses an SVM as the post-processing step.

The SVM algorithm obtains an optimal hyperplane to classify the data based on a large margin separation and a kernel function. The SVM tries to find the maximum distance between that hyperplane and the closest data point on either side of the hyperplane. The optimal hyperplane or the decision boundary is shown in Figure 4.

The SVM formulation is the equivalent of maximizing the margin while minimizing error. The cost function can be obtained using Equation (2) with the constraint shown by Equation (3) (Ferens, 2016):

Figure 3. A single-stage convolutional neural network architecture



$$C = \min_{w,b}(\frac{1}{2} \mid\mid w \mid\mid^2) \tag{2}$$

$$1 - y^{(i)}(w^T x^{(x)} + b) \leq 0 \quad i = 1,...,m \tag{3}$$

The gradient descent algorithm in Equation (4) is used to minimize the cost function *C*:

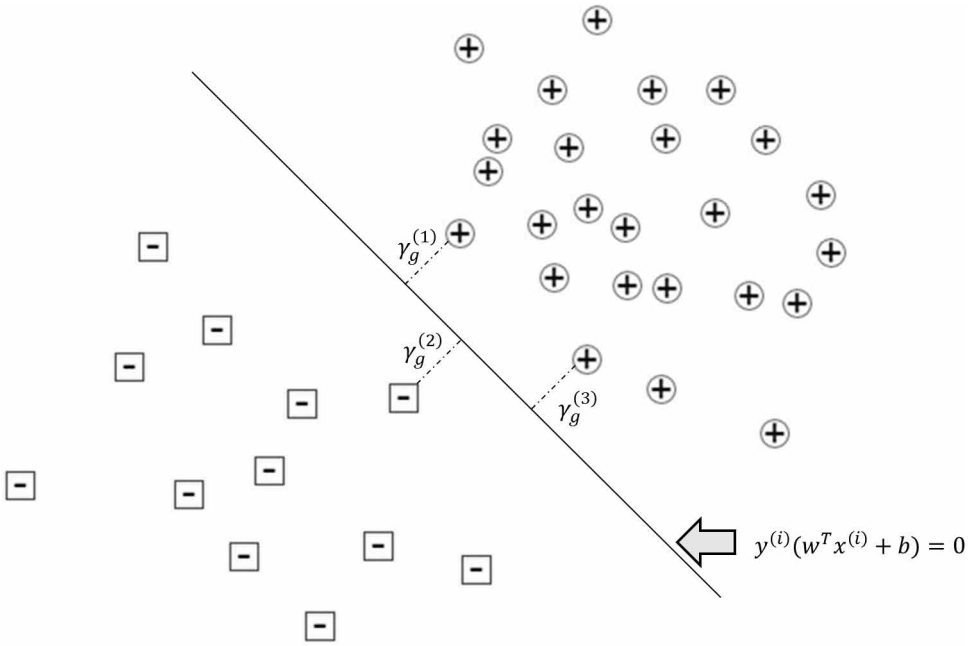
$$w_j \leftarrow w_{j-1} - \eta_t \nabla_{w,b} C(w,b) \tag{4}$$

The input of the SVM is the combination of the raw data, the ATS data, the pre-processed data, and the output of the processing step. The post-processing step increases the detection rate while reducing the detection processing time.

A FULL VERSION OF THE VARIANCE FRACTAL DIMENSION ALGORITHM

A fractal signal is a signal whose scaled-down version is similar to a part of itself (Kinsner, 2015). Also, a fractal signal is self-affine when different scales along different coordinates are used for a scale-down version of a signal. If the scales are the same, the signal is called self-similar. A network traffic signal is a nonstationary and a stochastic fractal signal. To analyze and measure a fractal

Figure 4. Optimal hyperplane to classify the positive and negative data with the support vector machine algorithm (After Ferens (2016))



signal complexity, a fractal dimension can be calculated. A fractal dimension interprets degree of meandering of a fractal signal (Kinsner, 2015). In this study, a polyscale analysis algorithm is used to measure the fractal dimension of the network traffic signal and extract distinguishing features of intrinsic characteristics of the signal (Kinsner, 2015). In contrast with multiscale analysis, where there is no correlation between outcomes, a polyscale analysis measures a signal with various scales, and its outcome requires all the scales be used simultaneously (Kinsner, 2015). In addition, this full version of the *variance fractal dimension* (VFD) is introduced because the input data is a stochastic fractal. For deterministic fractals, the simplified version can be used.

To analyze and measure the complexity of the signal using the VFD algorithm, the variance of its amplitude increments over a time increment has a power law relation with that time increment (Kinsner, 2015):

$$\mathrm{var}[A(t_2) - A(t_1)] \sim | t_2 - t_1 |^{2H} \tag{5}$$

where *var* is a variance function (the second moment), *A* is the signal, *H* is the Hurst exponent, and *t* is used to represent time in a continuous signal. After sampling the data and obtaining the discrete signal, the Hurst exponent can be calculated from a log-log plot through the following equation:

$$H = \lim_{\Delta n \to 0} \frac{1}{2} \left(\frac{\log_2 [\mathrm{var}(\Delta A)_{\Delta n}]}{\log_2 \Delta n} \right) \tag{6}$$

while *n* shows discrete time in a discrete signal.

The output of the VFD, which is a measure of signal complexity and shown by D_σ, can be obtained using Equation (7):

$$D_\sigma = E + 1 - H \tag{7}$$

where *E* is the Euclidean dimension, and it is equal to one for the time series data. The result of the VFD is a parameter, D_σ, which its value is limited between one and two that is shown by the following condition:

$$1 \le D_\sigma \le 2 \tag{8}$$

The real-time VFDv2 algorithm is characterized by a change in step 6 of the original VFD (Kinsner, 2015), as described below:

1. Find a frame size, *T*, in which the input data series is stationary, otherwise, the algorithm's result is invalid;
2. Segment the data series according to the length of the frame size;

Find the number of samples in each frame, N_T, in the current frame by Equation (9):

$$N_T = \left\lfloor \frac{T}{\delta_n} \right\rfloor \tag{9}$$

where δ_n is an interval between two successive sampled data.

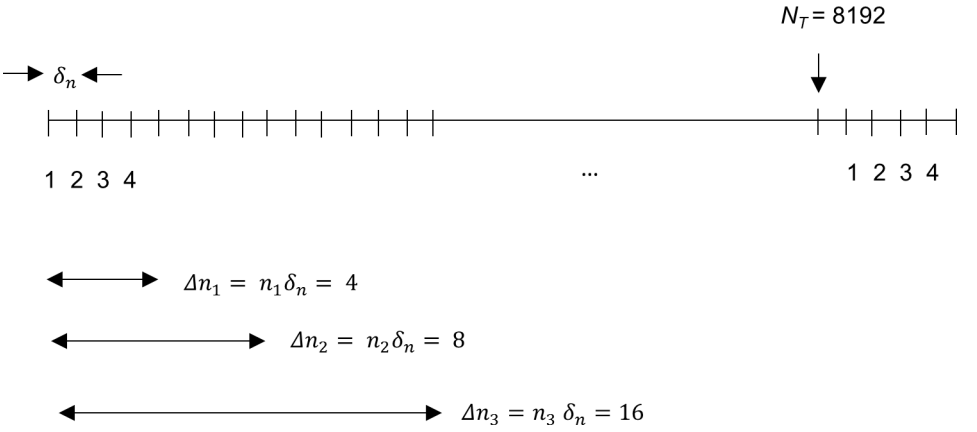
Find the maximum *volume element* (vel) size, $\Delta t_{k_{\max}}$, in the current frame using Equation (10):

$$\Delta t_{k_{\max}} = 2^{k_{\max}} \tag{10}$$

where $2^{k_{\max}}$ is the maximum number of samples in each interval, and $k_{\max}$ can be obtained from Equation (11). A relationship between parameters is shown in Figure 5. As an example, it is indicated that the number of vels within the frame size 8192 is equal to 2048, when the vel size is equal to four. If the vel size is equal to eight, then the number of vels within the frame is equal to 1024, etc.

5. To find the output of the VFDv2 for the data series (signal *A*) in each frame (N_T), parameters from Equations (11) to (14) are required. In the calculation program, a loop is designed to obtain the VFDv2 output within the lower bound, k_{low}, and the upper bound, k_{hi} values as shown by Equation (11) to Equation (14):

Figure 5. Relationship between parameters n_i, δ_n , N_T



$$K_{\max} = \left\lfloor \frac{\log_{10} N_T}{\log_{10} 2} \right\rfloor \tag{11}$$

$$K_{\text{hi}} = K_{\max} - K_{\text{buf}} \tag{12}$$

$$K_{buf} = \left\lfloor \frac{\log_{10} 30}{\log_{10} 2} \right\rfloor \tag{13}$$

$$K_{\text{low}} \geq 0 \tag{14}$$

6. To find the variance of the data series in the current frame, another nested loop is required. The parameters where described in Equations (15) and (16) are required to find the boundaries of the second loop:

$$n_k = 2^k \tag{15}$$

$$N_K = \left\lfloor N_T \,/\, n_k \right\rfloor \tag{16}$$

The loop is repeated N_K times, where N_K represents the number of vels in the frame. Also, N_K varies based on the values of n_k in each step, where n_k represents the size of vels in the frame. Therefore, by changing the vel size in each step, the concept of polyscale analysis is achieved using the VFDv2 algorithm. Equation (17) (Kinsner, 2015):

$$\mathrm{var}(\Delta A)_k = \frac{1}{N_K - 1} \sum_{j=1}^{N_k} [(\Delta A_{jk})^2 - \frac{1}{N_K} (\sum_{j=1}^{N_k} \Delta A_{jk})^2] \tag{17}$$

was adjusted to obtain the variance of the data series for stage *k*. Equation (18) is derived from Equation (17) for this study:

$$\mathrm{var}(\Delta A)_k = \frac{1}{N_T - n_k} \sum_{j=1}^{N_k} [(\Delta A_{jk})^2 - \frac{1}{N_T} (\sum_{j=1}^{N_k} \Delta A_{jk})^2] \tag{18}$$

where the amplitude increment ΔA can be obtained from Equation (19):

$$\Delta A_{jk} = A(j.n_k) - A((j-1).n_k) \text{ for } j = 1,...,N_k \tag{19}$$

An important issue regarding Equation (18) is that all points other than boundary points of the dataset, within the current frame, should be considered. Therefore, based on this concept, the variance is introduced through Equation (18). Figure 6 shows an example regarding a sliding vel with size of 16 that covers all points of the dataset within the current frame of data to calculate the VFDv2.

Figure 7 shows an example of a range of vel sizes from 1 to 4, and the number of vels in a frame size with 8192 data points.

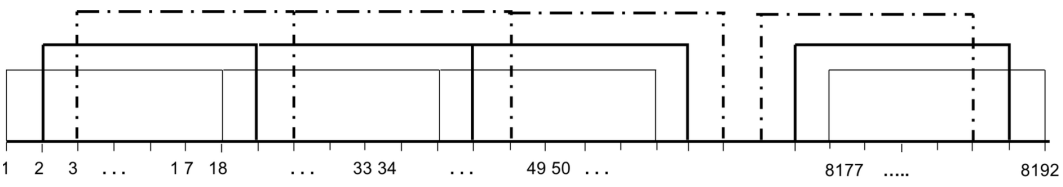
7. Find the log-log plot values for each coordinate axes from Equation (20) and Equation (21):

$$X_k = \log[n_k] \tag{20}$$

$$Y_k = \log[\mathrm{var}(\Delta A)_k] \tag{21}$$

8. Compute the slope, *S*, from the log-log plot based on the least squares fit by linear regression according to Equation (22):

Figure 6. A graphic illustration describing the concept of covering all points of data to calculate the VFDv2



$$S = \frac{K\sum_{i=1}^{K} X_i Y_i - \sum_{i=1}^{K} X_i \sum_{i=1}^{K} Y_i}{K\sum_{i=1}^{K} X_i^2 - (\sum_{i=1}^{K} X_i)^2} \tag{22}$$

9. Compute the Hurst exponent according to Equation (23) from the slope of Equation (22), which is extracted from Equation (6):

$$H = (\frac{1}{2})S \tag{23}$$

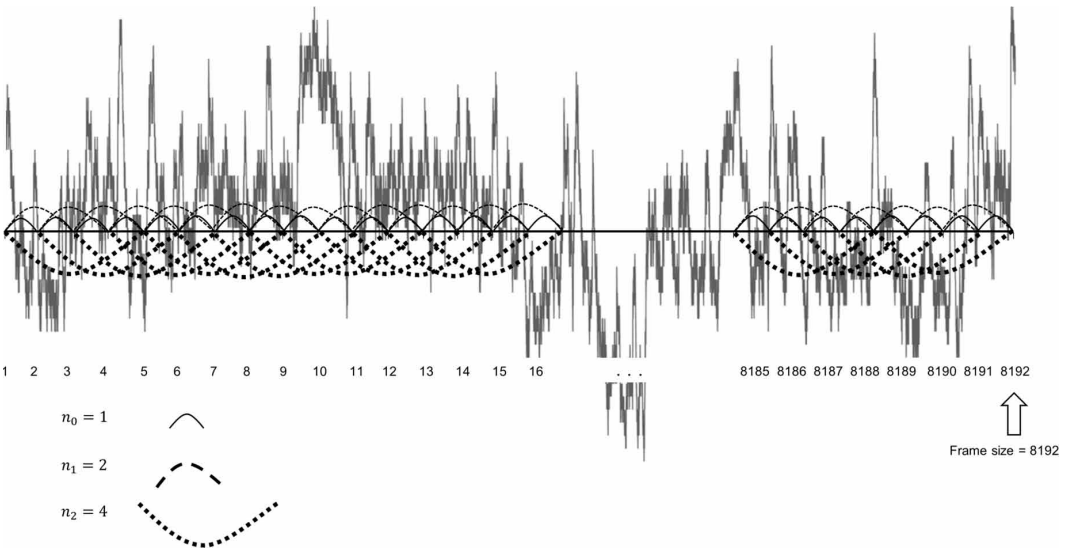
10. Compute the VFDv2 based on Equation (7);
11. Obtain the dimension for the next frame, until the end of the time series is reached. By a chain of the VFDv2 outputs, a VFDTv2 is calculated.

SIMULATION

In this section, the simulation of the proposed DDoS attack detection algorithm is described. The code was written with MATLAB R2017a, and the CC Compute Cluster at the University of Manitoba distributed the computationally intensive work. The simulation of the proposed algorithm followed five steps, which are described below.

First Step: The data output was tagged based on the number of packets within 0.1 *millisecond* (ms). If there were fewer than 40 packets within 0.1ms, the data was considered normal, so it was tagged

Figure 7. Relationship between a vel size and the number of vels in each frame to calculate the VFDv2 output



as zero. If there were more than 40 packets within 0.1ms, the data was considered anomalous, so it was tagged as one.

Second Step: The VFDTv2 was used to measure the complexity of the ATS data. To overcome the nonstationary problem, the stationary-frame size was selected by calculating the first and the second moment of the ATS data. In general, a frame size can be selected when the mean and variance of each frame are within two ranges, with about a 95% confidence level (Kinsner, 2015). In this simulation, a frame size of 8192 samples was chosen, and the lower bound in each frame, k_{low}, was assigned a zero value. At this point, the VFDv2 algorithm was performed to find a D_{σ} for each frame. In addition, an overlapping scheme of the VFDv2 algorithm was used to extract underlying features in real time, so the samples from each frame and its adjacent frame were overlapped. In this simulation, an overlapping size of 8191 samples was chosen to achieve a 99.98% overlap. Therefore, sensitivity to anomalous behaviors was increased in real time and any deviation from normal behaviors was declared. Finally, a VFDTv2, which was obtained by a chain of VFDv2s in each frame, was calculated as the output at this step. The output of this step was concatenated with the data.

Third Step: The *Daubechies wavelet transform 4* (db4) and the inverse Daubechies wavelet transform were calculated at each level. Different Daubechies db4 wavelet transform levels, ranging from level 1 to level 13, were applied to analyze the ATS data and the output of the VFDTv2 time series to find patterns. Different Daubechies db4 wavelet levels had different results and different sizes of output. Therefore, the output length of the db4 at each level was not the same as the input, so it was not possible to add these extracted features to the input, the ATS data and the output of the VFDTv2. To overcome this problem, the time series were reconstructed by the inverse Daubechies wavelet transform 4 at each of the 13 levels. As a result, the ATS data, the output of the VFDTv2, and the decomposed 13 distinct signals were concatenated with the raw data.

Fourth Step: The CNN parameters were trained. Based on the calculation described in step two, the stationary-frame size was calculated. Next, outputs of previous steps were normalized within the stationary-frame size. Accordingly, specific parameters and metadata to train the CNN were developed and were described in detail in (Ghanbari et al., 2017). More specifically, the learning rate (α) among a set of candidates: $\alpha = \{0, \ldots 0.1, \ldots, 0.5, \ldots, 1\}$ was found. In order to meet the convergence of error to 0 at early iterations, and to achieve the best detection rate, the learning rate, α, was set to 0.6. Using a part of the dataset, the CNN was trained to extract the features automatically, and obtain coefficients from the dataset. Next, using the remaining dataset, the CNN was tested with the obtained coefficients of the CNN training step.

Fifth Step: The data, the pre-processed data and the output of the training step were passed to the SVM for training the SVM parameters. The parameters were a vector of the weight coefficients, w, and a constant, b. Table 1 shows the parameters to train the SVM. The remaining raw dataset, the remaining pre-processed dataset, and the output of the CNN testing phase were passed to the SVM testing step, to classify normal and anomalous behaviors.

RESULTS AND DISCUSSION

In this section, the justification for each step and the results of the proposed anomaly detection algorithm are explained.

The VFDT was chosen for the pre-processing step because it is immune to noise, and it can be utilized in real time (Kinsner, 2015). The VFDv2 algorithm was introduced because the ATS data was not a pure fractal. Therefore, the average measure regarding all points of the ATS data with various scales should be considered. The result of the VFDv2 algorithm was between 1.1329 and 1.1828 for $k_{low} = 0$, and the result of the VFDv2 was between 2.0069 and 2.0163 for $k_{low} = 1$, which are shown in Figure 8 and Figure 9 respectively. In addition, the sliding window VFDTv2 was

Table 1. Specific parameters to train the SVM with one dimensional input

Parameters	Value
SVM structure	classification
Kernel function	sigmoid kernel
Penalizes misclassified cost	2
Vector of coefficients in SVM (w)	vector[60×1]
The constant in SVM (b)	vector[1×1]

used to increase sensitivity of the VFDv2 algorithm to extract features from a short duration of DDoS attack.

The DWT was chosen in pre-processing step because it is sensitive to irregularities in signals. Firstly, the DWT coefficients with anomalous events have larger magnitudes when compared to the DWT coefficients without anomalous events. Secondly, the DWT reacts to the change in the first derivative (slope) of a signal, not to the change in the amplitude of a signal. The db4 as a basis function for the DWT is used because it is suitable to detect an anomaly with a low amplitude, short duration, fast calculation of DWT coefficients, fast rate of decay, and rapid oscillation in a signal (Safavian, 2006).

Training of a CNN has a long duration. In addition, the classification accuracy of a CNN is improved by increasing the number of its stages, which decreases the speed of the training. To overcome this problem, the SVM was considered to increase the detection rate.

Figure 8. The trajectory regarding VFDv2 algorithm of the ATS data within the stationary frame and $k_{low} = 0$

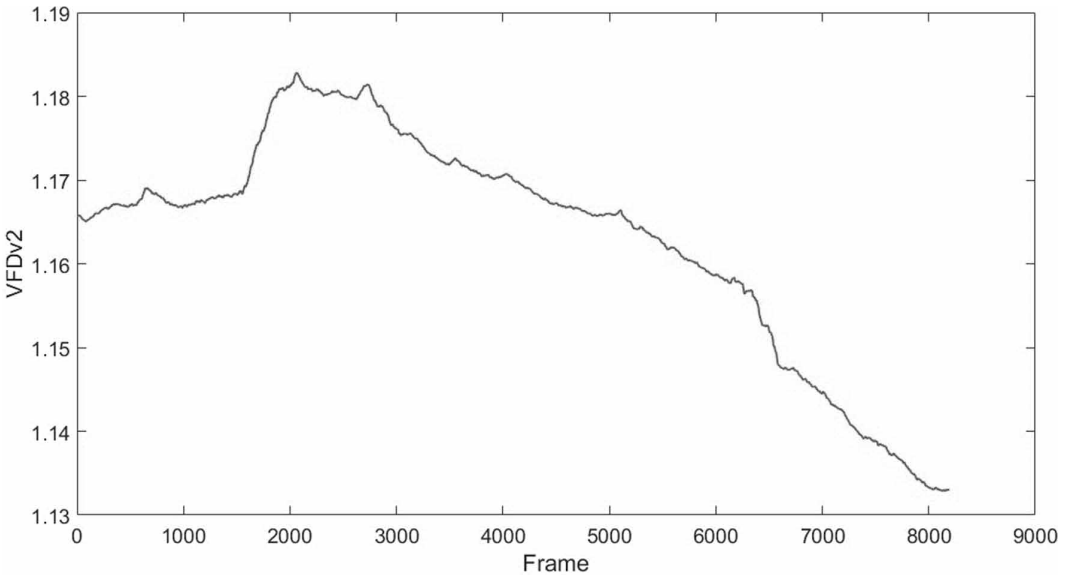
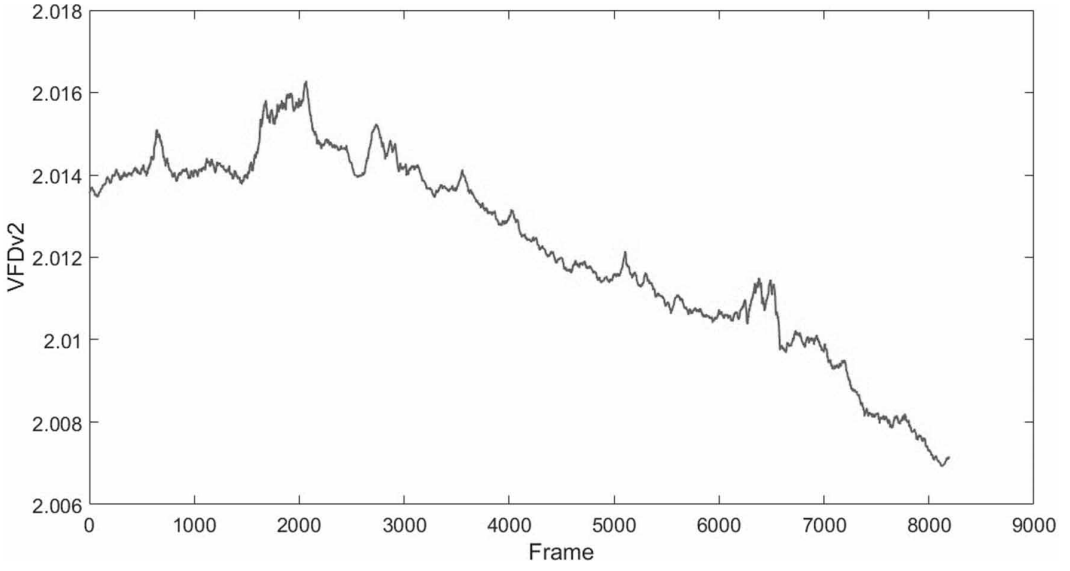


Figure 9. The trajectory regarding VFDv2 algorithm of the ATS data within the stationary frame and $k_{low} = 1$



In each proposed algorithm step, detection rates of DDoS attacks are evaluated using 50% of the dataset for training and 50% of the dataset for testing as shown in tables 2-7. When a single-stage CNN was considered, the detection rate was 56.1% (Ghanbari et al., 2017). However, when the db4 pre-processing step together with a single-stage CNN were considered, the detection rate was 80.77%, which was an increase of 24.6% from the previous result. The detection rate was further improved to 81.73% when the db4, the VFDTv2 and a single-stage CNN were considered. As a result, the distinguishing features in the pre-processing step were sensitive to anomalous behaviors in comparison with normal behaviors. When we considered the db4 and the VFDTv2 as pre-processing step, a single-stage CNN, and the SVM as post-processing, the detection rate was 87.35%. Comparing the proposed algorithm result in Table 7 with our previous work (Ghanbari et al., 2017), the results in Table 3 show that an increase of 6.58% true positive detection rate was achieved. Consequently, using the SVM as a post-processing step increased the sensitivity to detect anomalous behaviors. We can now observe that the pre-processing step contributed significantly to the detection rate compared to the post-processing step in the proposed algorithm.

Finally, using the db4 and the VFDTv2 in the pre-processing stage, the one-stage CNN as a processing step with $\alpha = 0.6$ using 80% of the dataset for training and 20% of the dataset for testing, and the SVM as post-processing, the detection rate was 68.19% as shown in Table 8.

DISCUSSION

Since the proposed detection DDoS attacks method is based on supervised learning, it needs tagged data as its input. To detect untagged data, an extension of the method is required such as one of the non-supervised learning methods. In addition, since this research was conducted to detect vulnerabilities and attacks in a smart grid infrastructure, we have extended the method to assist in preventing the attacks, or mitigating the effect of the attacks. Therefore, a severity score, which is a measure of the risk regarding a vulnerability can be assigned to make a priority to response a threat for mitigating the effect of the threat (Suh-Lee & Jo, 2015).

Moreover, one common method to generate data artificially to expand datasets for training deep learning algorithms, and increasing classification rate is data augmentation. One important condition

Table 2. Simulation result of the one-stage CNN as a processing step (Ghanbari et al., 2017)

Parameters	Value
Number of features as input to CNN	6
True positive rate	56.1%

Table 3. Simulation result of the pre-processed data by the db4 and one stage CNN (Ghanbari et al., 2017)

Parameters	Value
Number of features as input to CNN	32
True positive rate	80.77%

Table 4. Simulation result of the preprocessed data using the db4, the VFDv2 and the one-stage CNN

Parameters	Value
k_{low} (a parameter for the VFDv2 algorithm)	1
Number of features as input to CNN	59
True positive rate	81.73%

Table 5. Simulation result of using the SVM

Parameters	Value
Number of features as input to SVM	6
True positive rate	28.1%

is that the data augmentation should not affect constraint of the data. For example, flipping and stretching an image are data augmentation methods, so by applying the methods on a cat image, the result is still a cat. Therefore, data augmentation preserves constraint of data. However, the current study considers time-series signals that contain DDoS attacks. Data augmentation affects constraint of this data. For example, consider an *Electrocardiography* (ECG) signal. A healthy heart beats between 60 to 100 beats per minute (Hart, 2015), but with stretching, an ECG signal that shows one beat per minute does not belong to a healthy heart. As a result, the constraint of the signal is not preserved any more. Likewise, data augmentation cannot apply for time-series signals that contain DDoS attacks. (The time-series signals that contain DDoS attacks rely on long-range dependence

Table 6. Simulation result of the pre-processed data using the db4 and the VFDv2, the one-stage CNN as a processing step with a set of candidate α and the SVM as a post-processing step

Learning Rate (α)	True Positive
0.1	87.02%
0.4	87.30%
0.6	87.35%
0.9	86.42%

Table 7. Simulation result of the pre-processed data using the db4 and the VFDv2, the one-stage CNN as a processing step with α = 0.6 using 50% of the dataset for training and 50% of the dataset for testing and the SVM as a post-processing step

Parameters	Value
k_{low}	1
Number of features as input to CNN	59
Number of features as input to SVM	60
False positive rate	12.26%
False negative rate	0.37%
True positive rate	87.35%

Table 8. Simulation result of the pre-processed data, a single-stage CNN using 80% of the dataset for training and 20% of the dataset for testing and the post-processing step

Parameters	Value
k_{low}	1
Number of features as input to CNN	59
Number of features as input to SVM	60
False positive rate	31.80 %
False negative rate	0.01 %
True positive rate	68.19 %

that can be measured by the Hurst exponent. The closer to one the value of the Hurst exponent the greater the degree of long memory or long-range dependence (Zhou, Han, & Tang, 2011). In the current study, the time-series signals have the dimension between 2.0069 and 2.0163. According to Equation (23), the Hurst exponent has values between 1.00345 and 1.00815, which are close to one, so the data has a heavy tail. Internet traffic with this characteristic follows the Poisson distribution (Zhou et al., 2011)) and the Lévy distribution. (The ATS data, which is calculated as a new attribute in the current study, is based on the difference between packet-arrival times. Also, the data output was tagged as normal or anomalous based on the number of packets within 0.1 ms. Therefore, stretching, squeezing, cutting a chunk of data and other methods of data augmentation change the ATS data and the tagged output, so the constraints are not preserved. For example, one situation in which the constraints are not preserved is when data fails to follow the Poisson distribution.) Therefore, data augmentation does not apply to ECG and Internet traffic data. To increase anomaly detection, another CNN architecture should be applied.

EXTENSION OF THE WORK: MULTIFRACTAL DATA AUGMENTATION

Small data sets can lead to overfitting and poor generalization. Several techniques can be used to cope with the problem such as regularization, dropouts, and batch normalization (Xiang & Li, 2017). This can be applied even to generative adversarial networks (Goodfellow at al., 2014), such as the CycleGAN (Zhu, et al., 2017).

A small dataset can be expanded by simple or complex transformations of the original set. This is often done in 2D signals by simple techniques such as translation, flipping, rotation, scaling, zooming, cropping, interpolation, extrapolation, adding Gaussian noise. Data augmentation can also be done by transferring styles (e.g., texture/ambiance/appearance) from images in the dataset using neural networks (Pan & Yang, 2010). Generative adversarial networks (GANs) and conditional GANs (cGANs) can generate content from the existing data. The objective of a GAN is to train a parameterized function to map a source of noise to the input space of synthetic signals.

While most of the work has been done on images, one-dimensional signals (e.g., stock price index) are being now considered too (e.g., Lou, Qi, & Li, J., 2018). We are considering such transformations as they relate to time series. The major challenge of this data augmentation will be to preserve the multifractal nature of the Internet traffic and the long-range dependence. Some of the challenges may require adaptive multi-objective memetic optimization techniques such as reported in Dang & Kinsner (2016).

CONCLUSION

In this paper, an anomaly detection method was proposed to detect DDoS attacks. The aim of this paper was to achieve a higher accuracy detection rate. Therefore, by using distinguishing features of the data carrying anomalous behavior that causes to increase sensitivity of the CNN to classify the input data, enhanced sensitivity of detection was achieved. A higher accuracy detection rate was achieved by adding the full version of the variance fractal dimension trajectory, VFDTv2, as pre-processing step of the stochastic fractal input data and the SVM as post-processing step, and the proposed anomaly detection method detected the DDoS attack with 87.35% accuracy. This method of detecting DDoS attacks allows detection of anomalous behavior in real time, and adapts to various environments.

ACKNOWLEDGMENT

Support for CAIDA's Internet Traces is provided by the National Science Foundation, the US Department of Homeland Security, and CAIDA Members.

REFERENCES

Asri, S., & Pranggono, B. (2015). Impact of Distributed Denial-of-Service Attack on Advanced Metering Infrastructure. *Wireless Personal Communications*, *83*(3), 2211–2223. doi:10.1007/s11277-015-2510-3

Barford, P., Kline, J., Plonka, D., & Ron, A. (2002). A Signal Analysis of Network Traffic Anomalies. *Proc. ACM SIGCOMM Internet Measurment Workshop*, 1, 1–12. doi:10.1145/637201.637210

Beigi Mohammadi, N., Mišić, J., Mišić, V. B., & Khazaei, H. (2014). A framework for intrusion detection system in advanced metering infrastructure. *Security and Communication Networks*, *7*(1), 195–205. doi:10.1002/sec.690

Beqiri, E. (2009). *Neural Networks for Intrusion Detection Systems*. Berlin: Springer. doi:10.1007/978-3-642-04062-7_17

Center for Applied Internet Data Analysis. (2018, September 4). *The CAIDA 'DDoS Attack 2007' Dataset*. Retrieved from http://www.caida.org/data/passive/ddos-20070804_dataset.xml

Dang, H. V., & Kinsner, W. (2016). Adaptive multi-objective memetic optimization. *International Journal of Cognitive Informatics and Natural Intelligence*, *10*(4), 21–58. doi:10.4018/IJCINI.2016100102

Ferens, K. (2016). *ECE 7650: Applied Computational Intelligence* [Course notes]. Winnipeg, Canada: Faculty of Engineering, University of Manitoba.

Gao, R. X., & Yan, R. (2011). *Wavelets Theory and Applications for Manufacturing*. Springer.

Gao, R. X., & Yan, R. (2011). Wavelets Theory and Applications for Manufacturing (pp. 49–50). Springer. doi -1545-010.1007/978-1-4419

Ghanbari, M., Kinsner, W., & Ferens, K. (2017). Detecting a distributed denial of service attack using a pre-processed convolutional neural network. In 2017 IEEE Electrical Power and Energy Conference (EPEC) (pp. 1–6). IEEE. doi:10.1109/EPEC.2017.8286243

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. Cambridge, MA: MIT Press.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., . . . Bengio, Y. (2014). Generative adversarial networks. *Poster in Proc. Advances in Neural Information Processing Systems, 27*(3), 2672–2680. http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf

Hart, J. (2015). Normal resting pulse rate ranges. *Journal of Nursing Education and Practice*, *5*(8). doi:10.5430/jnep.v5n8p95

Hu, J., Pota, H. R., & Guo, S. (2014). Taxonomy of Attacks for Agent-Based Smart Grids. *IEEE Transactions on Parallel and Distributed Systems*, *25*(7), 1886–1895. doi:10.1109/TPDS.2013.301

Khan, M. S., Ferens, K., & Kinsner, W. (2015). *A Polyscale Autonomous Sliding Window for Cognitive Machine Classification of Malicious Internet Traffic*. Retrieved from https://www.semanticscholar.org/paper/A-Polyscale-Autonomous-Sliding-Window-for-Cognitive-Khan-Ferens/eb6fa8d9824f7d475e2efd2a0cd5b3b5d646169f

Kinsner, W. (2007). A unified approach to fractal dimensions. *International Journal of Cognitive Informatics and Natural Intelligence*, *1*(4), 26–46. doi:10.4018/jcini.2007100103

Kinsner, W. (2012) Towards cognitive security systems. In *Proc. of the 11th IEEE Intern. Conf. on Cognitive Informatics and Cognitive Computing*. Kyoto, Japan: IEEE.

Kinsner, W. (2012). Towards cognitive security systems. In Proc. of the 11th IEEE Intern. Conf. on Cognitive Informatics and Cognitive Computing (ICCI*CC) (pp. 22–24). Kyoto, Japan: IEEE (Keynote Speech). doi:10.1109/ICCI-CC.2012.6311207

Kinsner, W. (2015). *ECE 7210: Fractal and Chaos Engineering* [Course notes]. Winnipeg, Canada: Faculty of Engineering, University of Manitoba.

Kinsner, W. (2015). ECE 7210: Fractal and Chaos Engineering [Course notes], week 7, session 1, pp. ch. II-p4-13 – ch. II-p4-23. Faculty of Engineering, University of Manitoba, Winnipeg, Canada.

Lou, H., Qi, Z., & Li, J. (2018). One-dimensional data augmentation using a Wasserstein generative adversarial network with supervised signal. *Proc. 2018 Chinese Control And Decision Conference*. doi:10.1109/CCDC.2018.8407436

Nazir, S., Patel, S., & Patel, D. (2017). Autonomic Computing Architecture for SCADA Cyber Security. *International Journal of Cognitive Informatics and Natural Intelligence*, *11*(4), 66–79. doi:10.4018/IJCINI.2017100104

Pan, S. J., & Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, *22*(10), 1345–1359. doi:10.1109/TKDE.2009.191

Safavian, L. S. (2006). *Wavelet and multifractal analysis of transients in power systems*. University of Manitoba. Retrieved from https://mspace.lib.umanitoba.ca/handle/1993/20880

Suh-Lee, C., & Jo, J. (2015). Quantifying security risk by measuring network risk conditions. *2015 IEEE/ACIS 14th International Conference on Computer and Information Science, ICIS 2015 - Proceedings*, 9–14. doi:10.1109/ICIS.2015.7166562

Wang, Y. (2002). On cognitive informatics. *Proc. 1st IEEE Intern. Conf. Cognitive Informatic*, 34-42. doi:10.1109/COGINF.2002.1039280

Xiang, S., & Li, H. (2017). *On the effects of batch and weight normalization in generative adversarial networks*. Retrieved from https://arxiv.org/pdf/1704.03971.pdf

Yan, Y., Qian, Y., Sharif, H., & Tipper, D. (2013). A survey on smart grid communication infrastructures: Motivations, requirements and challenges. *IEEE Communications Surveys and Tutorials*, *15*(1), 5–20. doi:10.1109/SURV.2012.021312.00034

Zhou, S., Han, J., & Tang, H. (2011). Trust Measurement Model for Industrial Control Ethernet Network. *Advanced Materials Research, 268-270*, 1568-1573. Retrieved from www.scientific.net/amr.268-270.1568

Zhu, J. Y., Park, T., Isola, P., & Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. *Proc. IEEE International Conference on Computer Vision (ICCV17)*. Retrieved from https://junyanz.github.io/CycleGAN/

Maryam Ghanbari is a PhD candidate in the Department of Electrical and Computer Engineering at the University of Manitoba, working with her advisor Professor Witold Kinsner. She obtained her Master's degree in Computer Software Engineering and her Bachelor's degree in Computer Software Engineering both from the Islamic Azad University, Qazvin Branch, Iran. Her current research focuses on the detection of vulnerabilities in smart grids that are under cyber domain and physical domain attacks, using polyscale analysis and convolutional neural networks with deep learning.

Witold Kinsner is Professor in the Department of Electrical and Computer Engineering, University of Manitoba, Winnipeg, Canada. He obtained his Ph.D. degree in Electrical and Computer Engineering from McMaster University in 1974, and became Assistant Professor at McMaster University and then at McGill University. He was a co-founder and Director of Research of the Industrial Applications of Microelectronics Centre from 1979 to 1987. He is a Fellow of the Engineering Institute of Canada (FEIC), a Fellow of Engineers Canada (FEC), a Fellow of the Canadian Academy of Engineering (CAE), a Life Senior Member of the Institute of Electrical & Electronics Engineers (IEEE), a member of the Association of Computing Machinery (ACM), the American Association for the Advancement of Science (AAAS), the Engineers Geoscientists Manitoba (APEGM), and a member of other societies. For over 45 years, he has been very active all the IEEE levels: Region 7 (IEEE Canada), Council, Section, Chapter, and Student Branch. He was elected IEEE Canada President Elect 2014-2015, IEEE Director Elect (Region 7) 2014-2015, IEEE Canada President/Director 2016-2017, Past President 2018-19, and IEEE Educational Activities Vice President, 2018-19. He has been involved in research on cognitive systems, computational intelligence, robust real-time computing engines, and computer memories. Applications included biomedical, industrial monitoring and controls, aerospace, and space. He has authored and co-authored over 780 publications in the above areas, as well as supervised 73 Master's and Doctorate graduate students, over 200 undergraduate final-year thesis/capstone project students, and mentored 35 summer research students.