MICAM: VISUALIZING FEATURE EXTRACTION OF NONNATURAL DATA

Randy Klepetko and Ram Krishnan

Department of Electrical and Computer Engineering
University of Texas at San Antonio, San Antonio, Texas, USA

ABSTRACT

Convolutional Neural Networks (CNN) continue to revolutionize image recognition technology and are being used in non-image related fields such as cybersecurity. They are known to work as feature extractors, identifying patterns within large data sets, but when dealing with nonnatural data, what these features represent is not understood. Several class activation map (CAM) visualization tools are available that assist with understanding the CNN decisions when used with images, but they are not intuitively comprehended when dealing with nonnatural security data. Understanding what the extracted features represent should enable the data analyst and model architect tailor a model to maximize the extracted features while minimizing the computational parameters. In this paper we offer a new tool Model integrated Class Activation Maps, (MiCAM) which allows the analyst the ability to visually compare extracted feature intensities at the individual layer detail. We explore using this new tool to analyse several datasets. First the MNIST handwriting data set to gain a baseline understanding. We then analyse two security data sets: computers process metrics from cloud based application servers that are infected with malware and the CIC-IDS-2017 IP data traffic set and identify how re-ordering nonnatural security related data affects feature extraction performance and identify how reordering the data affect feature extraction performance.

KEYWORDS

Convolutional Neural Networks, Security, Malware Detection, Visualizations, Deep Learning

1. INTRODUCTION

Improvements in CNN have achieved better than human performance in computer image recognition [1]. They also have applications in non-image related research. Other sources of data include text [2], sound [3], and in the medical diagnostics of DNA [4]. These are examples where the data is organized in a grid like fashion by nature. But what about cases where the data wasn't ordered by natural phenomena? Sensors on an automated vehicle [5] for example, does the order matter? In most "*nonnatural*" applications the grid order is defined by some man made structure, usually defined by an arbitrary specification. The term "*nonnatural*" is for those data ordering schemes not defined by nature, as opposed "unnatural" which infers the "supernatural".

Using CNN in detecting cyber-security issues has shown significant interest. Raw IP traffic [6, 7] computer process metrics [8], and industrial sensors [9] are all data sets where researchers are evaluating CNN use in security. CNN are successful as they identify patterns from large data sets to extract features. CNN are often applied as a feature extractor source which is supplied as the input stage to decision network such as a densely connected, recurrent, or another machine learning procedure. To maximize the patterns detected, order of the grid supplied to a CNN should be of concern when it was arbitrarily defined.

In our previous research we showed that using the structural order in detecting malware using computer process metrics is not preferred when training a shallow or deep CNN model if high accuracy and precision are desired. We found that using statistical relationships as a basis for order does improve performance. We showed that grouping our data points created *artificial objects* that most CNN models could better identify as malware features. Do these finding hold true when analysing raw IP data traffic?

CNN models consist of various layers each performing a specific task. Some run convolutions via a series of filters, some pool data points together, while others perform mathematical operations over either one or a pair of grids. Comprehending what could be going on within these "black boxes" is improved with *visualization* techniques that let the user by eyesight understand what the network is doing.

By providing transparency and an explanation [10] as to the network parameter intensities they assist the researcher in all stages of the network development life cycle. Early in model construction visualizations provide failure details letting the engineer to see how performance is affected by model changes. Visualizing the hidden layers enhance confidence that the model is identifying a proper set of features during network maturity. As the network exceeds human performance, the visualization tools provide a computer instructor, teaching novel ways of examining the data to the researcher.

A number of visualization tools have been created to assist in the engineering and development of CNN. Some image generating tools create graphs to provide a higher level understanding of the data flow within the model. Other visualization tools provide histograms of the parameters as they adjust over the training period. One important class of visualization tools are classification response graphs which are designed to show the how responsive a pixel is to that particular classification made on a tested sample. These include Salience and CAM graphs. Most of these latter tools apply well with image data, but are not as well suited for data that is not visual in nature like cyber-security. These novel cases is where this research is focused. To find the patterns that the CNN layers are extracting from non-natural security data as features, we built a better visualization tool.

The contributions of this paper are:

- Present a new visualization tool, Model integrated Class Activation Maps (MiCAM), a confluence of several visualization tools, and show how MiCAM assists in identifying feature extraction response.
- Test previous defined ordering algorithms with a new security data set, raw IP traffic from CIC-IDS-2017, showing again that statistical correlation provides a better than randomly ordered performance.

  The remainder of the paper is organized as follows: Section 2 discusses related work using CNN with nonnatural data and a background on visualization tools. Section 3 outlines the methodology including a description of MiCAM and data organization. Section 4 describes the analysis procedure and evaluation results. Section 5 summarizes and concludes this paper.

2. RELATED WORK

2.1. Convolutional Neural Networks and NonnaturalData

CNN have matured to where they have many applications, beyond the recognition of images. Their ability is to identify patterns in large data sets when that data can be arraigned in a grid. For instance, in the analysis of tire tread using the parameters measured during the manufacturing process. Lihao and Yanni [11] with eleven metrics sampled from four manufacturing levels, they arraigned a 4x11 matrix and were able to identify faulty tires with a 94% accuracy.

Golinko et al. in [12] used a one dimensional CNN as a feature extractor front for other machine learning algorithms (k-Nearest Neighbour with k=1, Support Vector Machine, and Random Forest), examining if the ordering of nonnatural ``Generic" source data for the CNN has a performance impact on the final classifying algorithm. They found that using statistical correlation as a method for identifying relationships of adjacent data performed well, but not pre-ordering the data for CNN feature extraction was detrimental. Using a correlation ordering scheme offered improvement in most cases, especially for kNN and SVN, improving accuracy from 76% with no feature extraction to 82% if the data points were ordered by correlation prior to CNN feature extraction.

In a collision detection systemPark, et. al. [13] used information from robotic sensors and actuators creating 66 data points. Testing both a one-dimensional CNN and a Support Vector Machine Regression they were able to show that the CNN would perform better if it trained with enough data, but the SVMR performed better with less training.

With cross-related sensor data (local speed, GPS location, and accelerometer) from automated vehicles, Van Wyk, et. al. [5] used an analyser to identify whenever any of the sensors behaved anomalously. The different analysers tested included a Kalman Filter, CNN, and a CNN-KF hybrid. Each had its unique benefits.

2.2. Convolutional Neural Networks and Security

CNNs have found value in cyber-security applications. Their ability to find patterns instead of statically looking for distinct signatures provide feature extraction from large data sets and using the algorithm's nonlinear space enables the dynamic/online detection of zero-day attacks. These data sources are usually nonnatural.

From hypervisors in a cloud environment Abdelsalem et al. [8] places process metrics as they are reported into a grid as they look for malware as it is injected into virtual machines. This produced a set of 35 metrics that were captured per time segment for every running process. They were supplied to a Lenet-5 [14] CNN. Using the order as found in the logs and specifications, they achieved an 89% accuracy. McDoleet. al. [15] follow up with research analysing deeper CNN architectures using the same data set and ordering scheme. Kimmellet. al. [16] includes using recurrent neural networks (RNN), by testing the validity of using long short term memories (LSTM) and Bi-Direction LSTMs. They also explore if the order has an effect on training and discover that it does affect performance for all models.

Arranging raw IP traffic packets in a grid after the physical layer was stripped, Zhang et. al. [7]analysed them using CNN, LSTM, and a hybrid of the two. They tested for both binary classification (benign/maleficent) and multi-classification (benign + 10 maleficent types). They show all systems achieve quite remarkable, near-perfect results. For binary classification from the best in precision was the hybrid which was better than CNN, followed by LSTM. With multi-classification, CNN had some minor advantage in precision over the hybrid, but LSTM was behind both.

2.3. Visualizing Convolutional Neural Networks

Visually revealing the hidden layers provides researchers comprehension behind neural network decisions. They are also evolving as the field matures. They aresome form of flow and layer diagrams, class activation maps [17] (CAM), gradient visualization [18] sensitivity to perturbations [19], or a confluence of these.

Flow and model diagrams were introduced since the very first deep learning models were published. They provide a visual representation of the mathematical processing objects that are coded into the software. They represent these abstracts as spheres or cubes, and as multiple mathematical objects are aligned in a layer, the graphical constructs are placed next to each other in a row. A line between objects represent communication or parameter passing pathways. For convolutional layers, a plane of objects is used, and stacks of planes are a symbol which includes the third filter dimension. For brevity when the interpretation is understood, sometimes a higher dimension abstract is represented by a lower level visual construct.

CAMs were initially generated using a weighted sum and up-sampling the class activation maps from the penultimate layer to generate activation regions of the original image. CAMs have evolved using different parameters as the weight values for the ratio in summing the class activation maps. Detailed by Selvarajuet. al in 2016, GradCAM [10] uses gradients in a back propagation step with a *relu* function. LayerCAM[20] published by Jiang, et. al. collects the GradCAM maps from all of the individual layers and then sums them together in a normalized total that includes higher amount of detail from the shallower layers within the network.

GradCAM++ [21] by Chattopadhyay et. al. modified GradCAM by adjusting a normalizing factor used to determine the weights for the individual gradients from the feature activation maps. Devised by Wang et. al. in 2020 ScoreCAM [22], goes further by dropping the gradients altogether and include a contribution value to measure the importance of each activation map. EigenCAM submitted by Muhammad et. al. [23] replaces the gradients with an eigenvector that is derived from a combinations of the weights from all of the layers.

All of these CAM systems have several things in common. They attempt to produce a two dimensional region that shows how the features on the penultimate layer are related to the objects within the sample image, and they do so with only a single degree of the resulting image, grey scale. This works fine with shallower networks since the features within the penultimate layer are closely related to the pixels within the source image, but what about CNN models that are deep, and the final feature set have no direct relationship to the initial image, e.g. a source image of 75x75 pixels (75 x 75 x 3) and the resulting DenseNet-121 penultimate layer (2 x 2 x 1028). A 2x2 grid does not distinctly map to points on a 75x75 grid. A better visualization tool is needed tounderstand these deeper models.

2.4. CNN Models

Many models have been derived as CNN technology matures. Each new model uses a novel technique to accomplish higher degree of computer image object identification and classification precision. We examine three in this research. LeNet model [24], ResNet[25], and DenseNet[26]. They were chosen for their distinct architecture and their place as milestones in CNN evolution.

In 1989, LeCunet. al. introduced the LeNet-5 model in [24]. The first to use back propagation in a practical application as it identifies and classifies black and white images of hand written numbers provided by the US postal system. The goal, a 1% error rate, was reached after 23 epoch of training. It was sequential in structure and consisted of three convolutional and two dense layers. The data set they used closely resembles one used in this paper, the MNIST [27] data set of handwritten numbers.

He et. al. in late 2105 [25], introduced ResNet which added a new feature in network topology, the residual connection. This is a new link from the input of a convolution stage directly to the output, using addition, which feeds the next stage's input. This reintroduces the input data to the following stages, greatly reducing vanishing gradient, a major issue when training deep networks. They were able to win first in the 2015 ImageNet competition taking the prize in all categories: classification, localization, and detection. They also won the 2015 COCO competition in the categories of detection and segmentation. This research uses the smallest published version, ResNet-18.

Revised in 2018, Huang et. al. [26] published DenseNet. Like residual links, they have connections around layers but instead of using addition as the function for combining the input source with the output, they used concatenation. Each stage increases in depth from the previous, creating adepthwise*denser* input cluster. This forwards all of the input information and details previously gathered from earlier stages to the latter stages. This reduces the data lost by the addition process used in residual links, maintaining input integrity, further mitigating the vanishing gradient. They use bottleneck stages to reduce parameter count in the latter layers. These include a depth separable convolution to reduce the depth and a pooling layer for a reduction in width and height. This study uses DenseNet-121.

Our previous research expands on the techniques discussed by Abdelsalem et al. [8] by exploring the relationship between ordering of the rows, columns, and various CNN models' performance analysing cyber-security computer process metric data. We identified several structural relationships on which to base our ordering scheme, we included the use of a statistical relationship as an option for ordering the metric columns, and we compared those against a background of random orderings. We showed that using structural relationships as an ordering appeared to have no more advantage than a random order and statistical relationships as a foundation for order offered some performance improvement. We also shared that although the visualization tools available showed some response, the plots were difficult to interpret.

In this research we test these statistical ordering techniques using a different cyber-security data set, raw IP traffic from CIC-IDS-2017 following the work done by Zhang et. al. [7], and compare it to the structural order used in Zhang's research. We share a new tool, the Model integrated Class Activation Map (MiCAM), a confluence of model diagrams with activation maps displayed per layer. We use with the MNIST data set to establish a baseline so we can understand the visual representations as they are constructed for features extracted from black and white images. We then use this tool to analyse the features generated for two cyber-security data sets, computer process metrics and raw IP traffic, and show how it better displays feature extraction.

3. METHODOLOGY

3.1. Model integrated Class Activation Maps MiCAM

To fully visualize feature extraction we built a tool that is a combination of a model diagram with class activation maps. A model diagram is a flow plot that has the network layers displayed with the data pathways identified so the engineer can visually see the related connections between layers. This flow diagram is rather trivial when working with sequential models, but can be quite complex when dealing with network like Inception Net, that have multiple interconnections between layers. A class activation map (CAM) is a combination via a weighted sum of all of the activation maps for the filters a single layer. The weights for this sum define the type of CAM. This tool takes the model diagram and instead of displaying an object (i.e. layer) as a graphical construct (sphere or rectangle) it displays the CAM for that layer. After the MiCAM diagram is complete the result is a map clearly showing the various features that each layer defines as important in identifying the class of a tested sample. A diagram of the process steps used to generate MiCAM plots is found in Figure 1.

The multiple steps to the process are identified in alphabetical order. In the beginning the researcher has the chosen model and the data seen in (A). The model is trained in step (B) while at the same time, the model layout is extracted from the model definition. From the result, the trained model in (C) and the layout the layers are pulled out and the activation model is defined (D). This model has the pre-trained layers from the trained model laid out with the filters' outputs exposed for sampling later.

With the activation model, we take a sample (E) and test it determine how it is classified in (F). Using the activation model post-test and the model layout, we now extract the outputs or activations (G) for all of the filters and the associated filters' weights in (H). In step (I), using an inverse Fourier transform, we take the inverted convolution between a filters' activation and its kernels' weights. We then take the result for each filter and use the weight for the particular filter to sum a single CAM plot for each layer. This CAM plot is then up-sampled to match the original input grids dimensions.

To enhance the details within the CAM plots, we use the full RGBA pallete, by associating different variations of the CAM data within the plotted pixels. We note that every plot has a maximum and minimum range that is scaled to 256 discrete intensities. These pixel values can be positive or negative, so we use a set of *relu* functions to display these variations in intensities by matching one of the 4 degrees to a specific range of values. For blue we use the full range of minimum to maximum for this plot, scaled to the 256 colour levels. For red we display the positive peaks using the *relu* of the values, scaling from zero to the maximum of this plot. For green we display the negative peaks using *relu* of the negative value or zero if the values are positive, scaling from zero to the minimum. For alpha and size, we use the full range for the plot, but scale the results to the minimum and maximum values for all of the CAM plots within the model. The results are very dynamic images that display a full range of the extracted features.

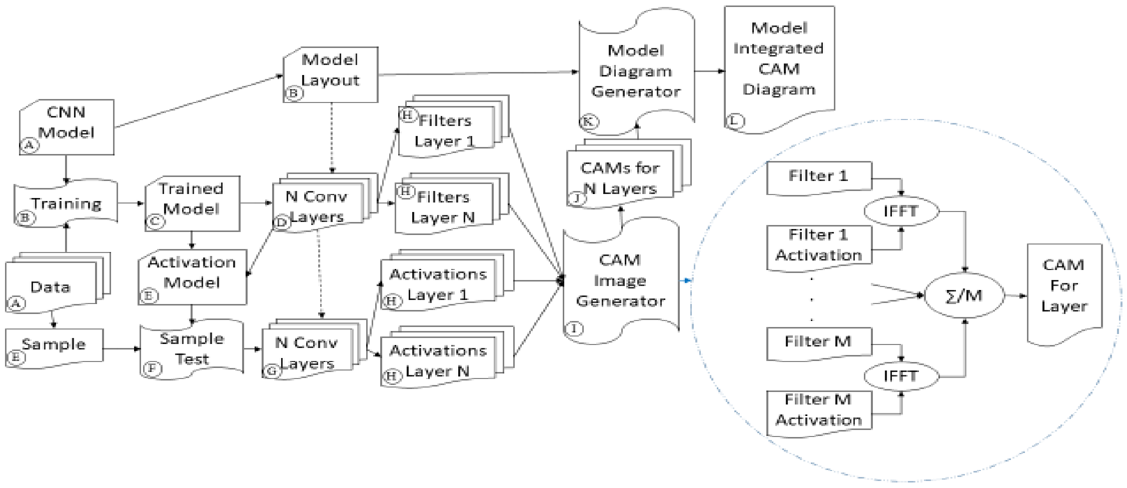

Figure 1. MiCAM Generation Process

After generating the images, we have a stack of CAM plots for all of the layers within the model (J). For layers that are not convolutional, we simply use a weighted sum of the outputs across the filter dimension, and then up-sample them to provide a graphic for each layer. For layers that are one-dimensional (flatten and dense) MiCAM fits the linear data within the input grid, scaling elements up if there are fewer data points within the layer than the width and height of the source data. The CAM plots are then integrated with the Model Layout in the Model Diagram Generator (K) which produces the final MiCAM diagram.

The code uses the "pydot/graphviz" graphical diagram module which has an interface for integrating images in place of objects. We added some slight modification for passing two list of parameters. One the list of layers than had CAM plot images, and the second was the list of the image files for the CAM plots. Both lists must be the same length, and for proper diagram generation the layer names in the first list should align with the filenames in the second list. The code is under open source license and found at https://github.com/rklepetko/MiCAM.gitfor easy access.

3.2. Dataset-1: MNIST Handwritten Numbers

The MNIST data set, compiled and released by Deng [27], consist of a library of images of hand written numerical text. The 10 image classes are from "0" to "9" and consist 60,000 samples from 250 census takers and 250 high school students. Another set of testing data was compiled from a separate group of 250 census and high school students, but comprised of only 10,000 samples. We join the two, shuffle them and use 20% of the data for testing, 20% in validation, or 14,000 of the samples per set, with the remaining used for training. Each sample was fitted in to a 20x20 grid, normalized for shading, and centered on a 28x28 image. For our analysis on deeper models, we further up-sampled the image to 75x75 pixels in size. Visual examples of our MNIST data are seen in Figure 2. We use several MNIST samples with the MiCAM diagrams to give us a base line on evaluating feature extraction.

Figure 2. MNIST Data Samples

3.3. Dataset-2: Malware Infected Computer Metric by Process Grids

The second data source is process metric samples taken from virtual machines in a cloud IaaS environment. They were application servers arrayed in a LAMP stack hosted web-site. The machines were injected with malware halfway through the experiment. There were 114 infections each from different malware packages. During the experiment, the server was polled for process log samples. Each sample is for a unique process running on the VM kernel and contains a set of ***M*** number of metrics per process during a section of time. Stacking ***P*** processes that are captured during a single time slice results in the matrix:

$$\mathbf{X}_{t=}\begin{bmatrix} & m_1 & m_2 & \dots & m_M \\ p_1 & x_{m1p1} & x_{m2p1} & \dots & x_{mMp1} \\ p_2 & x_{m1p2} & x_{m2p2} & \dots & x_{mMp2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ p_P & x_{m1pP} & x_{m2pP} & \dots & x_{mMpP} \end{bmatrix}$$

Our initial research was identifying how order of nonnatural data within the grid affects performance. We generated ten random rows and ten random columns for 100 options. We also identified several structural ordering methods and after examining the mathematical relationships within images derived several statistical relationships to see if they provide any improved performance. Since objects in images have pixels that are statistically correlated, we use the statistical functions used are detailed in Table 3 of Appendix A, at the end of this paper. The metric columns calculation were independent per sample, so we used correlation between two metrics (Eq. 1), absolute value of correlation (Eq. 3), and one minus the absolute value of the correlation, or what we called anticorrelation (Eq. 4) to test a counter hypothesis.

Unlike the independent metric columns, process rows calculations were dependent between samples, so the correlation function (Eq. 2) was derived per metric for a pair of processes. A sum of the correlation between two processes (Eq. 5) was used as the base process relationship function, from which we also derived an absolute correlation (Eq. 6) and anticorrelation (Eq. 7) relationship functions.

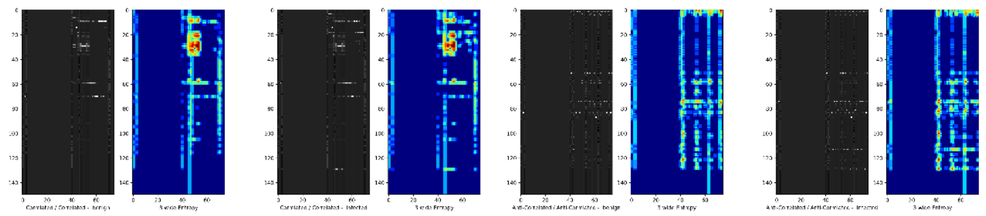

Figure 3. Correlated Rows & Columns (left) And Anticorrelated Rows & Columns(right) Benign & Infected Samples

These functions are then processed through the ordering algorithm, shown in Algorithm 1 found within the Appendix A which generates the ordering for each row or column along an axis. It can been seen in the samples ordered with correlation, Figure 3 left, and anticorrelation, Figure 3 right, our correlation functions generate artificial objects while the anticorrelation disperses them. The 35 metrics were expanded through one hot encoding to M = 75 metric columns and we made available room in the matrix for as many as P <= 150 process rows. The 29+ million process samples from 114 experiments (malware infections), and consisted of 31,064 grids, about half of which are considered infected. The experiments were split between 60% training, 20% validation, and 20% testing. The entire sample set for each experiment was included in the group it was assigned, so no experiment was split between training, validation, and testing. Every training and test set was reorganized among the 252 different ordering schemes we generated. We test all of our samples on several models, identified the best and worst ordering schemes (Table 6 in Appendix A) for each CNN model we trained, and then analysed the results of the best and worst ordering schemes with MiCAM.

3.4. Dataset-3: CIC-IDS-2017 Raw IP Data with Attack Vectors

The CIC-IDS-2017 data set has captured live, raw IP traffic that is intentionally subjected to various forms of attack vectors. There were 12 attack classes, ten of which were of a sizable sample. The sample count and break down by class is included with the results in Table 1 found in the next section. This traffic is compiled by session, with the sessions labeled benign or by attack class. Each packet in the session has the physical layer of the IP packet stripped, the first fourteen bytes, and only the following 160 bytes kept. If the original packet wasn't 174 bytes long, the remaining portion of the 160 bytes are supplied with zeros. The first ten packets of the session are then compiled in order of transmission, and if there aren't ten packets, the remaining are filled with zeros. The result is a 10x160 byte grid.

This is the basic single sample from the data set before it is reorganized into a 40x40 square. The current order of this gird is IP specification for the columns and transmission time for rows. Transmission time is a natural order, an instance in a sequence, but IP specification, human defined, is a nonnatural order. Is IP specification the best order? Will statistical correlation on the data be a high performing order? These are secondary questions this study is trying to resolve.

To test these hypothesis we first generated 100 random column ordering schemes to process and compare. Since the calculations between bytes are independent per sample we used the function Eq. 1 and the ordering algorithm shared in Algorithm 1, both found within the Appendix A. To diversify the number of ordering options available to analyse we used correlation relationships within different data subsets. The first data set was total of all samples. Next, we separate between the benign and maleficent and use the correlation of each of these data subsets. We then extract each of the attack types as subsets and generate correlated orderings from each of these. The idea is to see if it is possible to focus on a specific artificial objects by re-arraigning the order to match the correlation generated from that subset sample type. We also generate an absolute value of the correlation (Eq. 3) and anticorrelation (Eq. 4) orderings for each of the datasets.

This resulted in 146 ordering schemes to analyse. After reordering, the samples were then translated into a 40x40 grid by splitting the 160 bytes into four sections and stacking them on top of each other in order. We randomly reordered the samples and split them into 60% training, 20% validation, and 20% testing sets. We cover the evaluation in the next section.

4. EVALUATION

4.1. MiCAM and MNIST

The resulting MiCAM plots are large when compared to other CAM plots. They are usually vertically aligned following the model layout as the CNN is constructed. Since not only the convolutional layers, but the pooling, adding and concatenation layers, along with the final flatten and dense layers at the end of the convolutional stages are all plotted, the combined plot contains a visual representation of each layer. For example, DenseNet-121, with 121 convolutional layers has a total of 429 individual layers within the model. For brevity the diagrams are not all included but can be found on GitHub at: https://github.com/rklepetko/MiCAM.git. We do share snapshots of elements that illuminate the value of this visualization tool.

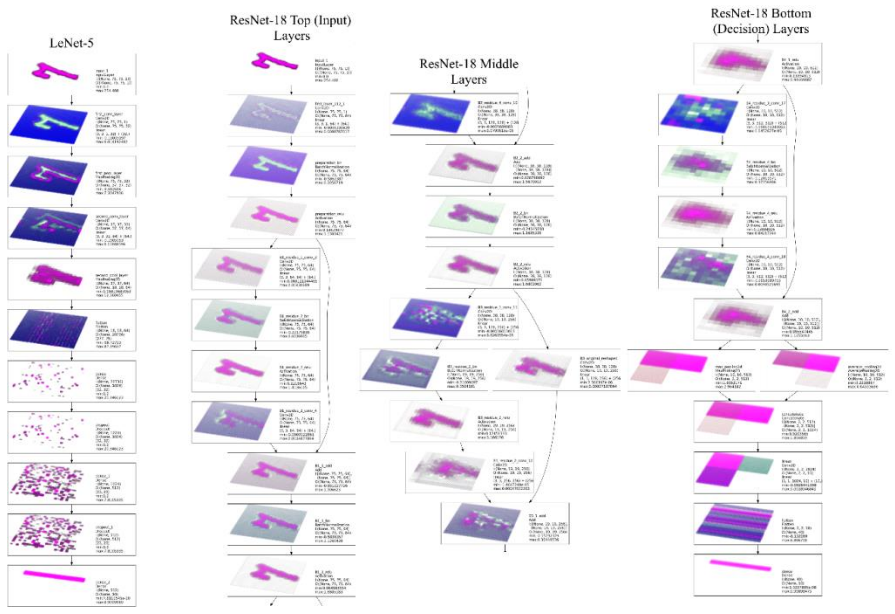
LeNet-5
ResNet-18 Top (Input) Layers
ResNet-18 Middle Layers
ResNet-18 Bottom (Decision) Layers

Figure 4. MiCAM Plots of LeNet-5 (left one) and MiCAM Plot Clips of ResNet-18(right three) analysing an MNIST sample”7”

To start we examine the LeNet-5 MiCAM plot (Left side of Figure 4) which clearly shows how the convolution layers build the identifying features. Examining the dense layers closely it can be seen the variation in the colour pixelintesities relate to specific features the network has identified.

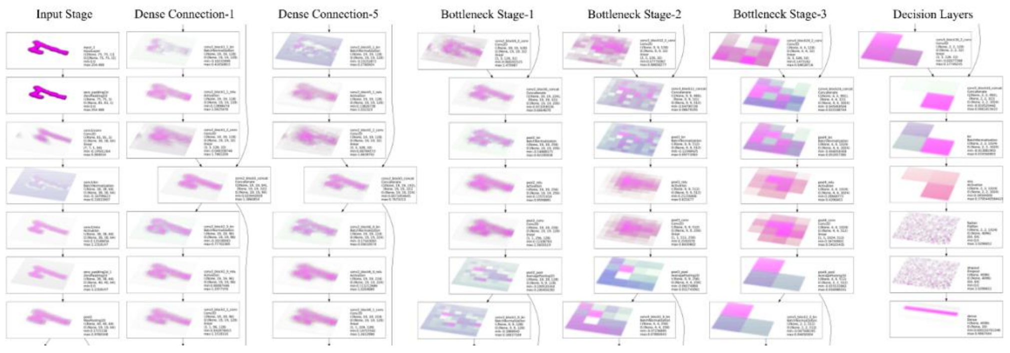

Figure 5. Clips of MiCAM Plot from a DenseNet-121 analysing anMNIST sample”7”

It is even clearer when examining ResNet-18 MiCAM plot (the right three plots of Figure-4) as we display the top, or input stages, the middle of the model, and the final bottom or decision layers. It's seen in these graphs how the residual links re-introduce features extracted from earlier layers. It can also be viewed within the final layers how the ResNet-18 network collapses the number of extracted features to relatively few, 40, as compared to LeNet-5 which was 20736.

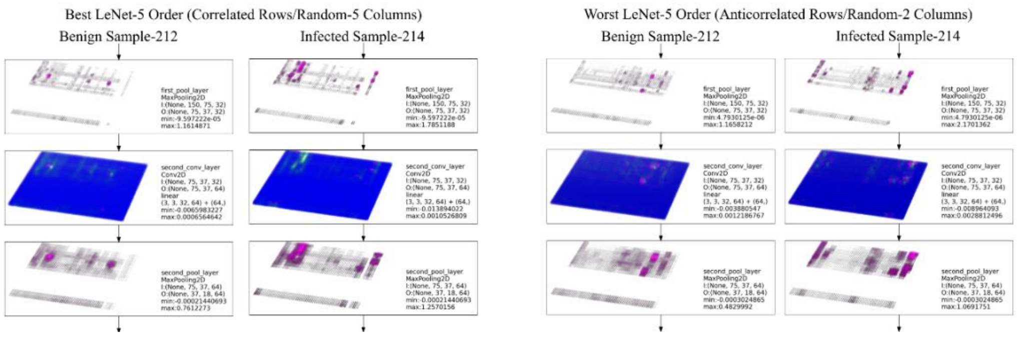

Figure 6. MiCAM Plots of the Lower Quarter for the LeNet-5 Best (left) andWorst (right) Orderingof Samples Benign #212 and Infected #214

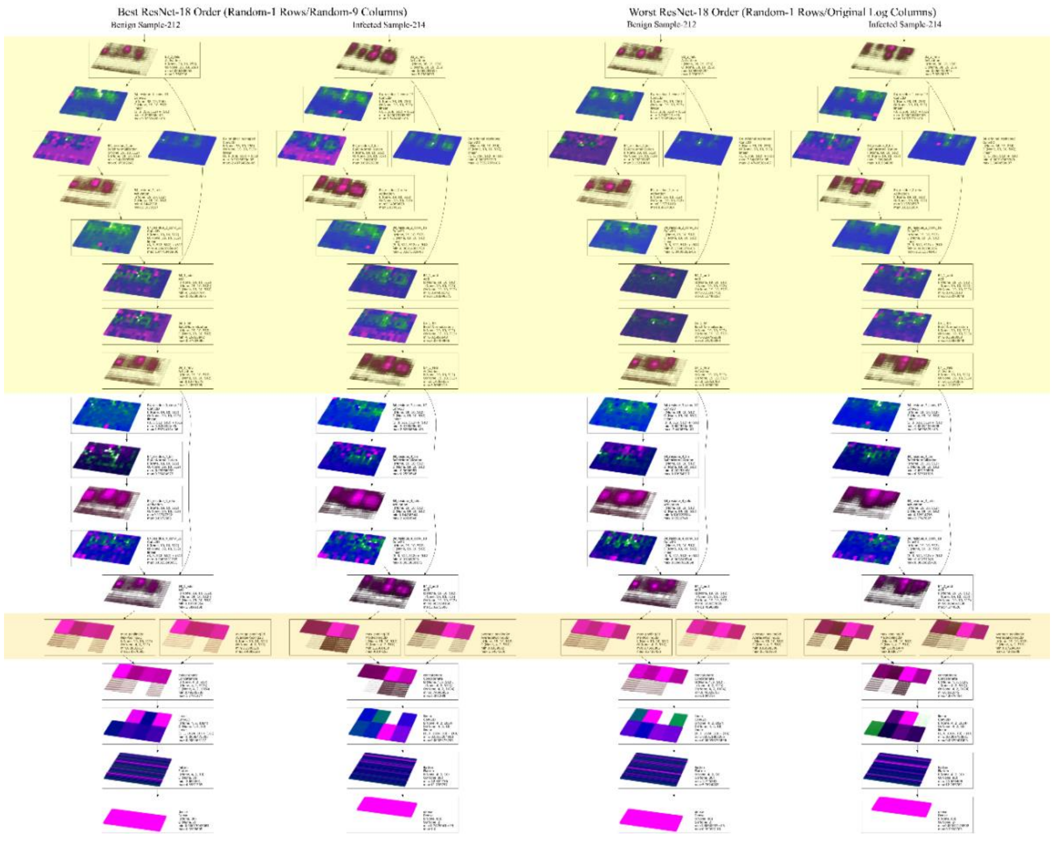

Figure 7. MiCAM Plots of Pooling and Last Convolution Layers for the ResNet-5Best (left) and Worst (right) Ordering of Samples Benign #212 and Infected #214

Examining the DenseNet-121 MiCAM plot of the same sample (Figure 5), we choose to share 49 of the 429 layers. From left to right we include the details of the input layers, the first and last dense connection before the first bottleneck, the three bottle neck stages, and the final decision layers. In the dense connection plots, the reintroduction of the input stages initial features (outline of a "7") is visible as the data cascades through all the way to the first bottleneck stage, maintaining a higher level of details for feature extraction precision. We can also see that it is these bottle neck layers that are compiling the features for discrimination later.

Table 1. Sample Counts by Class Set and Analysis Results by Order.

Sample			**Prec/Recal mAP**			**Improve/Degrade**		
Random Average			99.545%			0%		
Internet Protocol Specification			**99.703%**			**34.675%**		
Column Order Sample Set	Count	%	Corr	ABS	Anti	Corr	ABS	Anti
Bot	1228	0.151%	99.54%	99.58%	99.57%	-0.36%	6.98%	6.58%
DDoS	44918	5.539%	99.61%	99.65%	99.55%	14.19%	22.21%	1.70%
DoS Hulk	5952	0.734%	99.58%	99.57%	99.53%	7.86%	5.70%	-3.04%
DoS Slowhttptest	4216	0.520%	99.54%	99.59%	99.56%	-0.27%	9.00%	3.60%
DoS slowloris	3872	0.477%	99.46%	99.55%	99.66%	-18.97%	1.98%	25.29%
FTP - Patator	3974	0.490%	99.59%	99.55%	99.52%	8.93%	0.34%	-5.35%
Infiltration	6	0.001%	99.59%	99.61%	99.64%	9.57%	13.52%	21.20%
PortScan	158410	19.534%	99.57%	99.55%	99.57%	4.51%	0.13%	4.48%
SSH-Patator	2978	0.367%	99.59%	99.56%	99.54%	9.82%	3.71%	-0.19%
Web Attack - Brute Force	1363	0.168%	99.61%	99.57%	99.48%	14.67%	5.37%	-15.16%
Web Attack - Sql Injection	12	0.001%	99.61%	99.59%	99.55%	15.33%	10.09%	0.84%
Web Attack - XSS	625	0.077%	99.58%	99.48%	99.52%	8.80%	-13.74%	-6.56%
Malfecient	227554	28.060%	99.56%	99.57%	99.53%	2.98%	6.30%	-4.36%
Benign	583411	71.940%	99.52%	99.54%	99.51%	-6.38%	-0.55%	-6.81%
Total	810965	100%	99.59%	99.59%	99.57%	10.95%	9.63%	6.47%
Average Improvement	-	-	-	-	-	5.44%	5.38%	1.91%

4.2. MiCAM and Malware Infections

As mentioned in the previous section, we use MiCAm to analyse the difference between the best and worst ordering schemes (Table 6 in Appendix A) when searching for malware. Between the LeNet-5 MiCAM plots we found the pooling layers to have the most distinguishing characteristics. It is visible in Figure 6 which is divided by the best and worst ordering schemes. We can see how the features are better defined in the pooling layers with the stronger intensities, and the range on the infected sample of the best order is noticeably larger in the second pooling layer than the worst order.

Within the ResNet-18 plots we see a number of items to take notice of in Figure 7. Several of the CAM plots are identifying clusters of data points they have some significance on the decision. In particular the B4 residue convolution layers and associated additions and activation layers, highlighted in yellow, perhaps point to particular data points the CNN identifies as maleficent or benign. Also noticed is that the features from the best ordering are distinct in the final pooling layers for the benign and infected samples, highlighted orange, but the worst order displays those layers as having similar features by comparison.

To keep this report within the space limit, we are not displaying the DenseNet-121 graphs, but they are available at the Git site mentioned earlier. Things to note, the CAM plots most relatable to the source data are the last convolution stage before the first bottle neck stage. We see a number of highlighted pixels of interest for the different classifications. In particular we notice a highlighted row within the best ordering scheme for an infected sample, perhaps informing us that we have an infected process on that row.

4.3. MiCAMand IP Attacks

One of the unique details this study considers relevant is analysing the affect that order has on nonnatural data, and one data set, the CIC-IDS-2017 raw IP-traffic data, poses a scenario to tests our hypothesis. As described previously, we devised 146 different columns related ordering schemes, and compare them with the results when using the order devised using the IP-specification as a scheme. We trained a shallow LeNet-3 CNN model (2 convolution and one dense layer), matching previously published research and the results are found in Table 1. They include the PR curve mAP for every non-random ordering scheme we devised including a percentage of improvement over the average mAP for all of the randomly generated schemes. We include a breakdown of the results in our conclusion section.

Table 2. Best and Worst Ordering Schemes for Maleficent IP-Traffic.

CNN Architecture	Best Column Order	mAP Score	Worst Column Row Order	mAP Score
Lenet-3 (10 Epoch)	IP Specification	99.70%	Random-40	99.45%

To analyse the differences between the best and worst ordering schemes with the MiCAM diagrams, we identified them and include their details in Table2.

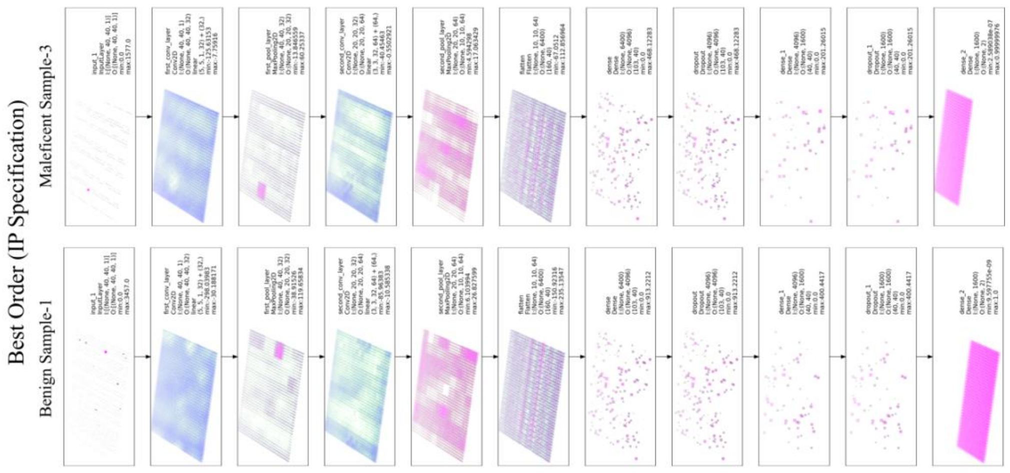

Figure 8. MiCAM Plots of LeNet-3 Analysing Best Order (IP Spec) IP Packets with Benign and Maleficent Packages

Examining the MiCAM plots, in Figures 8 and 9, we can see how the best order has a wider range, with the peak negative values showing very distinct regions within the convolutional layers. Also in both orders, in several layers it shows the first quarter of the sample is significant in finding the maleficent sample's attack vector, while several areas within the packet are identified significant in the benign.

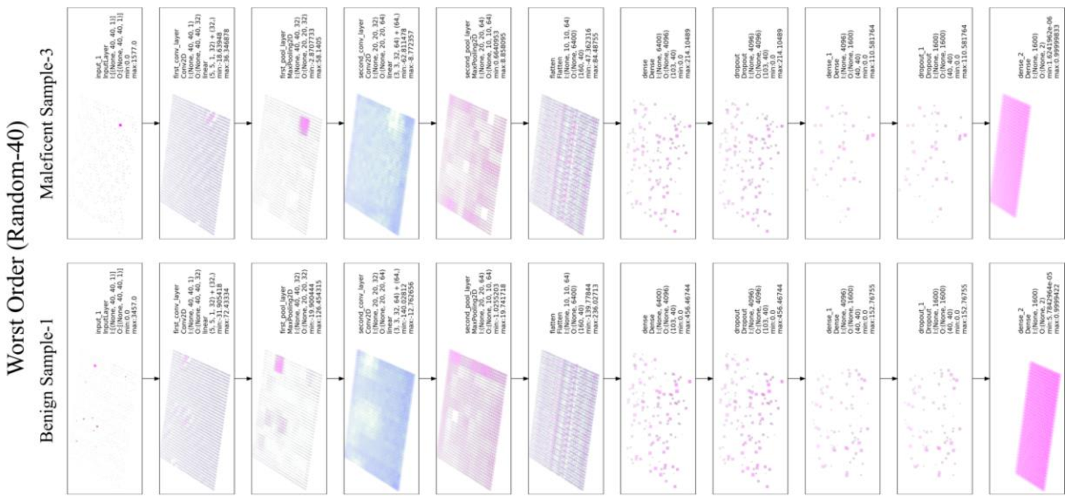

Figure 9. MiCAM Plots of LeNet-3 Analysing Worst Order (Randon-40) IP Packets with Benign and Maleficent Packages

5. Conclusions

The MiCAM diagrams offer more detail regarding feature extraction within the CNN models. They visually expose the layers allowing the user to further understand the intensities of the features extracted within the CNN structure. We've seen and identified several capabilities that allow us to further compare how minor variation in a model or process can affect feature extraction. This offers an additional tool for engineers as they tailor CNN models to non natural cybersecurty applications. We used CAM plots that normalized the sum of activation maps with the filters weights for the individual map, but one could enhance this tool to include other CAM variations, and better methods for displaying the one dimensional (flatten and dense) layers.

There is some processing cost related to generating the MiCAM plots. We went to some length to take advantage of the graphics engine by plotting all of the pixels within a single layer at one time which greatly improved the rendering speed.

When comparing the CIC-IDS-217 dataset ordering schemes, counter our hypothesis, the ordering scheme derived when following the IP specification exceeded expectations out performing all other ordering options. This shows the care to which IEEE specification was laid to logically organize the data packets as they relate to each other.

It is also interesting to note that the majority of the ordering schemes devised around a statistical relationship between data bytes within subsets of the data also performed better than average. The surprise regarding the subsets was the correlation of the benign samples. Only two other correlation subsets showed a major degradation in performance compared to the random average, and those sample sizes were less than one percent of the total samples. The benign correlation had 70% of the samples, but resulted in more than a 6% degradation. Focusing on benign samples to find maleficent actors proved detrimental. These findings support our hypothesis that statistical correlation does produce a better than average precision, as long as the data subset that the correlation is taken from has enough maleficent samples.

It's also notable that although anticorrelation ordering did have some significant improvement for some subsets, the majority of the subsets showed a poorer performance. Absolute value of correlation produced only one significantly detrimental ordering using a subset, which comprised of less than 1/10th of 1% of the total samples, so appears to be a relativity safe when using with a shallow network.

To further our understanding on how order affects CNN performance when analysing non-natural data, we plan on continuing our research by:

- UsingMiCAM to further analyse the differences in CNN model response when comparing ordering schemes.
- Identifying other security and nonsecurity datasets on which to test ordering hypothesis and techniques.
- Integrating the CNN feature extraction with other models to see if order can improve performance of ML hybrids.

ACKNOWLEDGEMENTS

This work is partially supported by NSF grants HRD-1736209 and CNS-1553696.

REFERENCES

[1] He, K., Zhang, X., Ren, S., Sun, J. (December 2015) Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: The IEEE International Conference on Computer Vision (ICCV).

[2] Lee, J.Y., Dernoncourt, F. (2016) Sequential short-text classification with recurrent and convolutional neural networks. CoRR abs/1603.03827, http://arxiv.org/abs/1603.03827.

[3] Deng, L., Hinton, G., Kingsbury, B. (May 2013) New types of deep neural network learning for speech recognition and related applications: an overview. In: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing. pp. 8599-8603 https://doi.org/10.1109/ICASSP.2013.6639344.

[4] Mobadersany, P., Yousefi, S., Amgad, M., Gutman, D.A., Barnholtz-Sloan, J.S., Velazquez Vega, J.E., Brat, D.J.,Cooper, L.A.D. (2018) Predicting cancer outcomes from histology and genomics using convolutional networks. Proceedings of the National Academy of Sciences 115(13), E2970-E2979. https://doi.org/10.1073/pnas.1717139115,https://www.pnas.org/content/115/13/E2970.

[5] van Wyk, F., Wang, Y., Khojandi, A., Masoud, N. (2020) Real-time sensor anomaly detection and identification in automated vehicles. IEEE Transactions on Intelligent Transportation Systems 21(3), 1264-1276. https://doi.org/10.1109/TITS.2019.2906038.

[6] Liu, C., Dai, L., Cui, W., Lin, T. (2019) A byte-level cnn method to detect dns tunnels. In: 2019 IEEE 38th International Performance Computing and Communications Conference (IPCCC). pp. 1-8. https://doi.org/10.1109/IPCCC47392.2019.8958714.

[7] Zhang, Y., Chen, X., Jin, L., Wang, X., Guo, D. (2019) Network intrusion detection: Based on deep hierarchical network and original flow data. IEEE Access 7, 37004-37016. https://doi.org/10.1109/ACCESS.2019.2905041.

[8] Abdelsalem, M., Krishnan, R., Huang, Y., Sandu, R. (2018) Malware detection in cloud infrastructure using convolutional neural networks. IEEE 11th International Conference on Cloud Computing.

[9] Hu, Y., Zhang, D., Cao, G., Pan, Q. (2019) Network data analysis and anomaly detection using CNN technique for industrial control systems security. In: 2019 IEEE International Conference on Systems, Man and Cybernetics (SMC). pp. 593-597. https://doi.org/10.1109/SMC.2019.8913895.

[10] Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D. (Oct 2019) Grad-cam: Visual explanations from deep networks via gradient-based localization. International Journal of Computer Vision 128(2), 336-359. https://doi.org/10.1007/s11263-019-01228-7, http://dx.doi.org/10.1007/s11263-019-01228-7.

[11] Lihao, W., Yanni, D. (Nov 2018) A fault diagnosis method of tread production line based on convolutional neural network. In: 2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS). pp. 987-990. https://doi.org/10.1109/ICSESS.2018.8663824.

[12] Golinko, E., Sonderman, T., Zhu, X. (Dec 2018) Learning convolutional neural networks from ordered features of generic data. In: 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA). pp. 897-900 https://doi.org/10.1109/ICMLA.2018.00145.
[13] Park, K.M., Kim, J., Park, J., Park, F.C. (2021) Learning-based realtime detection of robot collisions without joint torque sensors. IEEE Robotics and Automation Letters 6(1), 103–110. https://doi.org/10.1109/LRA.2020.3033269.
[14] Liu, G., Zhao, F. (2007) An efficient compression algorithm for hyperspectral images based on correlation coefficients adaptive three dimensional wavelet zerotree coding. In: 2007 IEEE International Conference on Image Processing. vol. 2, pp. II-341 -- II-344. https://doi.org/10.1109/ICIP.2007.4379162.
[15] McDole, A., Abdelsalam, M., Gupta, M., Mittal, S. (2020): Analyzing CNN based behavioural malware detection techniques on cloud IAAS. In: Zhang, Q.,Wang, Y., Zhang, L.J. (eds.) Cloud Computing - CLOUD 2020. pp. 64-79. Springer International Publishing, Cham.
[16] Kimmel, J.C., Mcdole, A.D., Abdelsalam, M., Gupta, M., Sandhu, R. (2021) Recurrent neural networks based online behavioural malware detection techniques for cloud infrastructure. IEEE Access 9, 68066-68080. https://doi.org/10.1109/ACCESS.2021.3077498.
[17] Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., Torralba, A. (2015) Learning deep features for discriminative localization.
[18] Erhan, D., Bengio, Y., Courville, A., Vincent, P. (2009) Visualizing higher layer features of a deep network. University of Montreal 1341(3), 1.
[19] Ribeiro, M.T., Singh, S., Guestrin, C. (2016) " why should i trust you?" explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. pp. 1135-1144
[20] Jiang, P.T., Zhang, C.B., Hou, Q., Cheng, M.M., Wei, Y. (June 2021) Layercam: Exploring hierarchical class activation maps. IEEE Transactions on Image Processing pp, 1-1. https://doi.org/10.1109/TIP.2021.3089943.
[21] Chattopadhay, A., Sarkar, A., Howlader, P., Balasubramanian, V.N. (Mar 2018) Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks. 2018 IEEE Winter Conference on Applications of Computer Vision (WACV). https://doi.org/10.1109/wacv.2018.00097, http://dx.doi.org/10.1109/WACV.2018.00097.
[22] Wang, H., Wang, Z., Du, M., Yang, F., Zhang, Z., Ding, S., Mardziel, P., Hu, X. (2020) Score-cam: Score-weighted visual explanations for convolutional neural networks
[23] Muhammad, M.B., Yeasin, M.(Jul 2020) Eigen-CAM: Class activation map using principal components. In: 2020 International Joint Conference on Neural Networks (IJCNN). IEEE. https://doi.org/10.1109/ijcnn48605.2020.9206626.
[24] LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., Jackel, L.D. (1989) Backpropagation applied to handwritten zip code recognition. Neural Computation 1(4), 541-551. https://doi.org/10.1162/neco.1989.1.4.541.
[25] He, K., Zhang, X., Ren, S., Sun, J. (2015) Deep residual learning for image recognition. CoRR abs/1512.03385,http://arxiv.org/abs/1512.03385.
[26] Huang, G., Liu, Z., Weinberger, K.Q. (2016) Densely connected convolutional networks. CoRR abs/1608.06993, http://arxiv.org/abs/1608.06993.
[27] Deng, L. (2012) Themnist database of handwritten digit images for machine learning research. IEEE Signal Processing Magazine 29(6),141-142.

APPENDIX-A

In Table 3on the next page are the equations we presented as ordering foundations in our previous research and test MiCAM analysis with in this study. They are used as parameters for ordering Algorithm 1 found on the following page.

Table 3. Detailed Set of Statistical Relationship Functions.

Equation 1: Metric/Byte Column Statistical Correlation Function $\rho_{mi\,mj} = \frac{E(x_{mi}x_{mj}) - E(x_{mi})E(x_{mj})}{\sqrt{E(x_{mi}^2) - E(x_{mi})^2} - \sqrt{E(x_{mj}^2) - E(x_{mj})^2}}$
Equation 2: Process Row Statistical Correlation Function $\rho_{mk\,pi\,pj} = \frac{E(x_{mk\,pi}\,x_{mk\,pj}) - E(x_{mk\,pi})E(x_{mk\,pj})}{\sqrt{E(x_{mk\,pi}^2) - E(x_{mk\,pi})^2} - \sqrt{E(x_{mk\,pj}^2) - E(x_{mk\,pj})^2}}$
Equation 3: Metric/Byte Column ABS-Correlation Function $\rho_{ABS\,mi\,mj} = \lvert\rho_{mi\,mj}\rvert$
Equation 4: Metric/Byte Column Anticorrelation Function $\rho_{ANTI\,mi\,mj} = (1 - \lvert\rho_{mi\,mj}\rvert)$
Equation 5: Process Row Correlation (Sum) for All Metrics Function $\rho_{SUM\,pi\,pj} = \sum_{k=1}^{M} \rho_{mk\,pi\,pj}$
Equation 6:Process Row ABS-Correlation for All Metrics Function $\rho_{ABS\,pi\,pj} = \sum_{k=1}^{M} \lvert\rho_{mk\,pi\,pj}\rvert$
Equation 7: Process Row Anticorrelation for All Metrics Function $\rho_{ANTI\,pi\,pj} = \sum_{k=1}^{M} (1 - \lvert\rho_{mk\,pi\,pj}\rvert)$
Equation 8: Metric/Byte Column Total Relationship Function $\rho_{TOT\,mi} = \sum_{j=1}^{M} (\rho_{mi\,mj})$
Equation 9: Process Row Total Relationship Function $\rho_{TOT\,pi} = \sum_{j=1}^{P} (\rho_{SUM\,pi\,pj})$

Algorithm 1: Derive Statistical Relationship Order.

```
For features along an axis, $f_i$, define a function, $\rho_{fi\,fj} \forall\, i, j$;
From $\rho_{fi\,fj}$ define $\rho_{TOT\,fi}\, \forall\, i$;
Create a selection pool of features $P \ni f_i$;
While $P \neq \emptyset$ do:
    Create and empty bidirectional queue $Q$ for features $f_i$;
    Find $max(\rho_{TOT\,fi})\ \forall f_i \in P$;
    Place corresponding feature $f_{max(\rho)}$ onto $Q$;
    Remove feature $f_{max(\rho)}$ from $P$;
    Create two pointers left, $L$, and right, $R$; $L, R \in Q$;
    Point $L$ and $R$ towards $f_{max(\rho)}$ in $Q$;
    While $P \neq \emptyset$ and not(STOP) do:
        If $\exists \rho_{fL\,fi}\ \forall f_i \in P$ or $\exists \rho_{fR\,fi}\ \forall f_i \in P$ then:
            Find $max(\rho_{fL\,fi}, \rho_{fR\,fi})\ \forall f_i \in P$;
            Place new feature $f_{max(\rho)}$ next to the appropriate $f_L$ or $f_R$ on $Q$;
            Remove new feature $f_{max(\rho)}$ from $P$;
            Move the appropriate pointer $L$ and $R$ towards the new $f_{max(\rho)}$ in $Q$;
        Else:
            Stack current queue $Q$ into final ordered axis $V$;
            STOP;
      End if else;
    End while;
End while;
```

Following that are two tables which are the results from our previously published VM Malware analysis but are now using with MiCAM. The first (Table 4) is the PR curve mAP results from the various ordering schemes. The second is (Table 5) the percentage of improvement (or degradation) over the observed average. The last (Table 6) shows the best and worst performing ordering schemes that we use to analyse with MiCAM.

Table 4. Mean AUC for Precision Recall Curves for Malware Analysis.

CNN Architecture	All Options	Correlated Rows	ABS-Corr Rows	Anti-Corr Rows	Correlated Columns	ABS-Corr Columns	Anti-Corr Columns
LENET-5 (20 epoch)	99.550%	99.680%	99.580%	99.090%	99.590%	99.600%	99.440%
ResNet-18	89.850%	87.020%	86.560%	94.530%	91.240%	89.230%	95.130%
DenseNet-121	99.530%	99.700%	99.430%	99.200%	99.600%	99.520%	99.560%

Table 5. Percentage Improvement over Average (Mean) Performance for Malware Analysis.

CNN Architecture	100% minus All Mean	Correlated Columns	ABS-Corr Columns	Anti-Corr Columns	Correlated Rows	ABS-Corr Rows	Anti-Corr Rows
LENET-5 (20 epoch)	0.450%	8.889%	11.111%	-24.444%	28.889%	6.667%	-102.222%
ResNet-18	10.150%	13.695%	-6.108%	52.020%	-27.882%	-32.414%	46.108%
DenseNet-121	0.470%	14.894%	-2.128%	6.383%	36.170%	-21.277%	-70.213%

Table 6. Best and Worst Ordering Schemes for Malware Analysis by CNN Model.

CNN	Best Combined		mAP	Worst Combined		mAP
Architecture	Row Order	Column Order	Score	Row Order	Column Order	Score
LENET-5 (20 epoch)	Correlated	Random-5	99.82%	Anticorrelated	Random-2	98.64%
ResNet-18	Random-1	Random-9	99.99%	Random-1	Original	50.31%
DenseNet-121	VMPID	Random-1	99.87%	ABS-Correlated	Random-5	96.36%

AUTHORS

Randy Klepetko graduated from Texas A&M University in May of 1990 with a Bachelors n Computer Science. Since, he has enjoyed a 30 year career in a broad range of engineering and digital fields. He continued his education with courses in electrical engineering and electronics at Texas A&M University, San Antonio College, and University at San Antonio during the 1990's and 2000's, saturating his knowledge in audio and video engineering protocols, methods and techniques. In 2017 he returned to academia at the University of Texas at San Antonio to enhance his competency regarding digital security where he was encouraged to join the University's Center for Security and Privacy Enhanced Cloud Computing (C-SPECC), receiving his Masters in May 2022 and scheduled PhD completion in December

Ram Krishnan is a Professor of Electrical and Computer Engineering at the University of Texas at San Antonio, where he holds Microsoft President's Endowed Professorship. His research focuses on (a) applying machine learning to strengthen cybersecurity of complex systems and (b) developing novel techniques to address security/privacy concerns in machine learning. He actively works on topics such as using deep learning techniques for runtime malware detection in cloud systems and automating identity and access control administration, security and privacy enhanced machine learning and defending against adversarial attacks in deep neural networks. He is a recipient of NSF CAREER award (2016), the University of Texas System Regents' Outstanding Teaching Award (2015) and the UTSA President's Distinguished Award for Research Achievement (2016). He received his PhD from George Mason University in 2010.

© 2023 By AIRCC Publishing Corporation. This article is published under the Creative Commons Attribution (CC BY) license.