24th International Congress on Modelling and Simulation, Sydney, NSW, Australia, 5 to 10 December 2021
mssanz.org.au/modsim2021

Data preprocessing for non-image data: using convolution neural networks for network intrusion detection systems

G. Glukhov [a] **and T. Lynar** [b]

[a]*School of Engineering and Information Technology, University of New South Wales, Canberra, ACT 2600, Australia*

[b]*School of Engineering and Information Technology, University of New South Wales, Canberra, ACT 2600, Australia*

Email: g.glukhov@unsw.edu.au

Abstract: Application of machine learning models in network intrusion detection systems has been the subject of extensive investigation and testing. Modern day networks produce data in quantities that put more emphasis on the accuracy and precision of the intrusion detection systems. This produces the drive for more accurate and time-efficient intrusion detection systems, and machine learning was investigated as a viable solution. Research into machine learning models in other fields has yielded several different algorithms and approaches, highly specialised to those particular data types. Testing for intrusion detection has found that the models that process the network data best tend to yield higher accuracy and lower false-positive rates, whereas those models that perform best on their original data have struggled. One such model that under performed in intrusion detection when compared to the original field is convolution neural network. This paper aims to investigate preprocessing methods for network data to increase the effectiveness of using a convolution neural network model as part of a network intrusion detection system. Specifically, the paper will analyse the use of the DeepInsight architecture,using a modified t-distributed stochastic neighbour embedding technique, the positioning of features in isolation and a control class of simple reshaping data from vector to matrix form.

Keywords: *Artificial intelligence, neural network, data preprocessing, feature mapping, network intrusion detection*

1 INTRODUCTION

A Network Intrusion Detection System (NIDS) monitors the activity of an entire protected network as a whole, analysing packet metadata and other communications. Originally, Rules-based methods Denning (1987) were used to identify malicious activity, with a later shift to anomaly detection methods as those became more feasible Axelsson (2000), which were focused on statistical learning processes Manikopoulos & Papavassiliou (2002) as well as early unsupervised machine learning models such as ZANERO Zanero & Savaresi (2004). Contemporary NIDS utilise a misuse detection approach, categorising abnormal activity first and treating uncategorised data as normal operational behaviour. Misuse detection is not a novel technique, as it was formally introduced in 1998 as an alternative to the rules-based IDS at the time Cannady (1998).

Misuse detection utilises information of normal traffic as well as hostile attacks to classify traffic, which typically results in a low false alarm rate when compared to its anomaly detection analogues. Supervised Learning is the dominant model used within misuse detection systems, with an extensive amount of research in the past being placed on shallow supervised learning models Chauhan et al. (2019), with Support Vector Machine Kim & Park (2003) and Decision tree Boukhris et al. (2017) systems being investigated amongst others. Recently the focus has shifted to deep supervised learning models. These algorithms tend to perform better when tested against new or unseen data, and when compared to previous, more shallow methods. The issue is that current development of deep learning research has been targeted towards image and text processing, but work is being done to integrate more deep learning methods into IDS Diro & Chilamkurti (2018). Convolution Neural Network (CNN) is one such deep learning approach that was developed as one of the most effective algorithms for image classification. Otherwise known as space invariant artificial neural networks, and have been demonstrated to be one of the most efficient and accurate classifiers in that domain Chauhan et al. (2019). Work has begun on integrating this algorithm into IDS Zheng (2020), but the issue of adapting this architecture to network data processing has been a consistent issue in getting the desired performance of this algorithm when compared to other alternatives, such as SVM or Decision Trees Maseer et al. (2021).

The main findings of the current research indicate that CNN models are less accurate and efficient when compared to more generalised models in IDS. However, the area of data preprocessing is lacking when compared to research into models. A tremendous amount of research has been put in to comparing the effectiveness of different models and algorithms in an IDS, with current benchmark being set by the other deep learning models Maseer et al. (2021). However, this puts the provably more effective models like the CNN at a disadvantage because these models are tailored for a different environment. The effectiveness of these models would hypothetically rise if the data they are input is in a similar format to what they are designed for, such as the CNN being fed image-based data. A novel transformation method DeepInsight Sharma et al. (2019) uses dimension reduction techniques such as t-SNE Van der Maaten & Hinton (2008) and kPCA Schölkopf et al. (1998) to transform non-image data into optimal image format. This method was tested on 5 datasets comprising of 5 different data types - gene expression dataset, speech dataset, text dataset and two artificial datasets. This method has not been tried on network data, which is what this paper aims to explore.

The CICIDS2017 dataset consists of network data for a system resembling modern network systems over five days. This network simulated typical activity for the industry, implementing different systems and protocols to resemble actual traffic. The dataset represents typical background traffic within a network, as well as several attack techniques such as DoS, DDoS, Brute Force, Heartbleed, Web Attack, Infiltration and Botnet Sharafaldin et al. (2018). The dataset includes typical data features seen in network data such as time of package, destination port, source port, protocol and so on. The dataset comes in two forms: the raw values and the processed version, where the non-integer values were converted to integer form. The dataset contains mostly benign attacks, and has fewer scenarios when compared to some other datasets such as the ICS cyberattack dataset Elmrabit et al. (2020).

A recent study comparing the effectiveness of different machine learning algorithms on the UNSW-NB15, CICIDS-2017 and ICS cyber-attack IDS datasets Elmrabit et al. (2020) identified the Random Forest algorithm as the most effective, reaching accuracy values between 0.97 and 1 and precision, recall and f1-score values of similar quality for multi-classification. Within the same paper, CNN still achieved comparable rates, which demonstrates its potential.

2 EXPERIMENT

Convolution neural networks are designed to process image data, typically in the form of a 2D matrix representing a monochrome image or a three-dimensional matrix representing three colour channels of images. It processes this form of data by utilising a kernel-based convolution, thus there is an intrinsic statistical element placed on the positioning of data points relative to its neighbours. This is accounted for in real world image data, where the data represented in pixels is an abstraction of real-world positioning. This statistical relationship is not typically accounted for in non-image data, and the placement of data within a matrix is arbitrary. The impact of this statistical relationship on machine learning is investigated further.

In order to increase effectiveness of machine learning methods, it is standard practice to pre-process the data. This practice typically involves such concepts as handling null values, standardisation, categorical variables, one-hot encoding, multicollinearity and normalisation among others. Beyond these procedures, in order to increase effectiveness of learning methods, statistically redundant data is also typically removed by the dimensionality reduction techniques mentioned above.

Basic preprocessing is done to expedite the learning algorithms, and takes the form of null value handling (converting null values to the average value of each class, inf to maximum and negative to minimum positive value) and normalisation (converting every value to a range between 0 and 1). To expedite processing, random sample of 1% of the dataset is used for both the control group, positioning group and the feature mapping groups.

2.1 Model

Description of PC Specs	
System	Windows
Version	10.0.19041
System Type	64-bit operating system, x64-based processor
Machine	AMD64
Processor	Intel64 Family 6 Model 158 Stepping 10, Genuine Intel
Processor Information	Intel(R) Core(TM) i7-8700K CPU @ 3.70GHz 3.70 GHz
Installed Ram	16.0 GB

The model used to test the effectiveness of the preprocessing methods is a CNN model, that follows this structure:

- 2D Convolution Layer / 120 nodes / padding = same
- 2D Convolution Layer / 60 nodes / padding = same
- 2D Convolution Layer / 30 nodes / padding = same
- Flatten and Softmax / 15 nodes

Convolution Layer. A convolution layer is the basic component of a CNN. The basic operation of a convolution layer is that it takes the input image and uses a kernel-based operation on it to extract features. Typically, this operation would reduce the dimensions of the data, as the function of the kernel is to abstract a subgroup of inputs into one output, however, with the ”same” padding setting in the convolution layers, the dimensions of the input are preserved by adding ”padding” - an outer axis of zeroes for the kernel to process.

Flatten and Softmax. The flatten and softmax layer is the classifier for the model. It takes all the nodes that are created as part of the model, and compresses them down to 15 output nodes - 15 representing the 15 classes that are within the CICIDS17 dataset. The softmax function is applied to ensure that the probabilities of each class add up to 1 - demonstrating that all the options are accounted for and there are no missing probabilities.

2.2 Control Group

A control group is used to compare the effectiveness of the feature mapping method. It uses a simple reshape, as it is efficient since it takes the least amount of time to convert data into a trainable form. It is also efficient in the amount of space it occupies - the smallest square matrix is created that could house all the features of the data.

The original data has 78 features, which need to be converted from a (78, 1) vector into a (9, 9, 1) matrix. This is performed by a simple reshaping of the vector, filling the 3 null values created as result with zeroes as padding.

Fundamentally, this is the most basic and efficient preprocessing method. This representation retains all original features, and adds minimal redundant information to enable the square matrix.

$$\begin{bmatrix} a & b & c & d & e & f & g & h & i \end{bmatrix} \longrightarrow \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$

(a) Feature vector to matrix conversion for Control Group.

$$\begin{bmatrix} \begin{bmatrix} a & b & c & d & e & f & g & h & i \end{bmatrix} \\ \begin{bmatrix} 3 & 4 & 6 & 2 & 8 & 1 & 5 & 9 & 7 \end{bmatrix} \end{bmatrix} \longrightarrow \begin{bmatrix} f & d & a \\ b & g & c \\ i & e & h \end{bmatrix}$$

(b) Feature vector to matrix conversion for Sequence Group.

Figure 1. Vector to Matrix Conversions

2.3 Feature Sequencing

Adequately representing the relationship that neighbouring pixels have within a real-world image is difficult within the framework of non-image data, since ideally each feature of the data will carry as much information as possible while having the smallest correlation with other features. The DeepInsight framework Sharma et al. (2019) provides a novel way to achieve this by using t-SNE. Rather than reducing the dimensions of each data entry, this framework converts the range of values a given feature takes into a Cartesian plane representation. This enables similar features to be grouped closer together, and dissimilar features to be further apart.

The basic data undergoes the same process as the control group initially. Once the data is standardised, t-SNE is used to determine the feature mapping positions for each feature within the dataset. Rather than reducing the dimensionality of each data entry in the dataset, t-SNE is used to reduce the dimensionality of each feature, where the dimension being reduced is the value each particular feature takes over the whole dataset. This procedure results in a two-dimensional representation of each feature, which can be used to map said feature on a Cartesian plane. This mapping process typically leaves a considerable amount of empty space around the mapping, which is removed by finding the minimum bounding rectangle for the features and transforming the points to fit that rectangle.

To investigate the effectiveness of feature mapping, the dataset will undergo an intermediary step. Instead of performing the DeepInsight process of mapping features into their positions, the features are sorted based on their mapping coordinates and reshaped into a matrix of 9 x 9 x 1 dimensions, to compare more generally between the control group and the reduced mapping group. This is referred to as sequencing as the positions of the features are not accurate, only the relative sequence in which they appear.

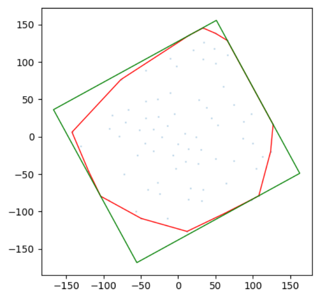
2.4 Feature Mapping

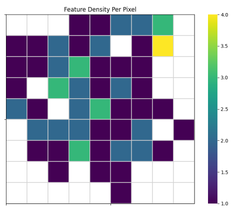
Once a feature map is created, each data entry is mapped using it, where every feature of the data entry is placed in the location that it occupies on the Cartesian plane. The coordinates are then converted into a matrix representation - if two or more features occupy the same matrix position, the average is taken. This matrix is then used as input for the model. The mapping is done in two varieties: the full and the reduced. The reduced has the dimensions of the control and sequence groups, while the full is more spaced out in order to negate the large overlap that occurs at low pixel rates. The matrix dimensions for the full mapping are set to 50 x 50 x 1 for this experiment, but can be modified.

The goal of this technique is not to reduce the amount of data that is being processed by the machine. In fact, the dimensionality typically increases as result of feature mapping. However, to account for it, a reduced Feature Mapping Matrix is also used which shares the dimensions of the control and positioning groups.

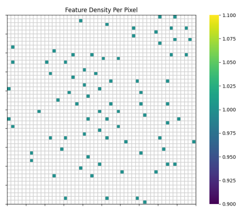
3 RESULTS

Fundamentally, since the model and dataset is the same across both trials, any discrepancy between the results is a consequence of the difference in preprocessing methods. Thus, an inspection of the results reveals that the accuracy fell when the larger, spatial data representation was used, while also severely affecting machine performance and increasing processing time. To measure the effectiveness of machine learning models, a confusion matrix methodology is used. Within confusion matrix calculations, a ”true” result refers to when


(a) The features mapped on a Cartesian plane, minimum bounding rectangle to minimise noise



(b) Reduced Feature Mapping Matrix



(c) Full Feature Mapping Matrix

Figure 2. Feature Mapping Visuals

the classifier identified the data correctly and ”false” incorrectly, while ”Positive” refers to the attribute the model is trying to classify.

$$\begin{aligned} TP &= TruePositive \\ TN &= TrueNegative \\ FP &= FalsePositive \\ FN &= FalseNegative \end{aligned} \quad (1)$$

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \quad (2)$$

Within a confusion matrix, accuracy is determined by the amount of correctly classified data entries over the whole dataset. This is most frequently used metric to determine the effectiveness of a dataset, however, it fails to account for the difference between a true positive and a true negative, so while the model might be very accurate at identifying the majority of classes, it could fail to identify a certain minority class.

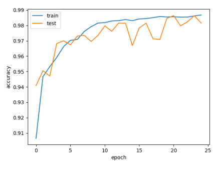
Loss within machine learning refers to a measure of an incorrect prediction of the model. A lower loss indicates the prediction is closer to the actual result. For categorisation of a multi-class dataset such as CICIDS 2017 a sparse categorical cross entropy loss function is used, as seen in (3), where y_i is the correct label, $\hat{y_i}$ is the estimated label, i is the index of the element in the dataset and N is the total number of elements.

$$SparseCategoricalCrossEntropy = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y_i}) + (1 - y_i)\log(1 - \hat{y_i})] \quad (3)$$

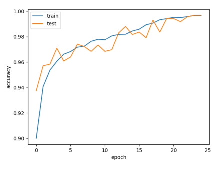
Precision is a measure of how correctly the model identifies the ”positive” data - in other words, how correct is the model when it identifies data as belonging to a class.

$$Precision = \frac{TP}{TP + FP} \quad (4) \qquad Recall = \frac{TP}{TP + FN} \quad (5) \qquad F1\text{−}score = \frac{2}{\frac{1}{Recall} + \frac{1}{Precision}} \quad (6)$$

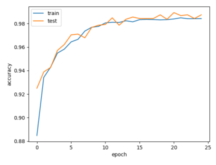
Recall refers to how well the model was able to identify all the ”positive” data - how many classes the model failed to indentify compared to how many it did. F1-score can be interpreted as the aggregate of recall and precision measurements, as it measures both the harmonic mean of the recall rate and the precision rate, and while it represents the performance of the model very accurately, it is difficult to interpret due to the combination of factors that contribute to it.



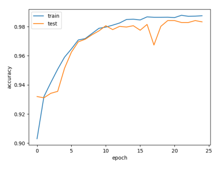
(a) Control Group



(b) Sequence Mapping

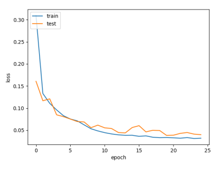


(c) Reduced Feature Mapping Matrix

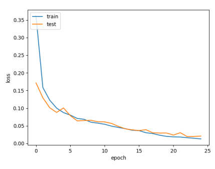


(d) Full Mapping Matrix

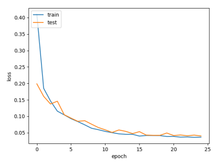
Figure 3. Accuracy Performance



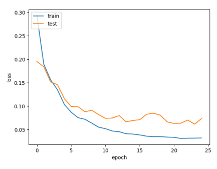
(a) Control Group



(b) Sequence Mapping



(c) Reduced Feature Mapping Matrix



(d) Full Mapping Matrix

Figure 4. Loss Performance

Model	Accuracy	Loss	Precision	Recall	f1- score
Control Group - Macro Average			0.69	0.74	0.71
- Weighted Average	0.9735	0.0879	0.98	0.97	0.97
Sequence Mapping - Macro Average			0.76	0.74	0.73
- Weighted Average	0.9573	0.1467	0.96	0.96	0.95
Reduced Mapping - Macro Average			0.78	0.75	0.76
- Weighted Average	0.9837	0.0521	0.98	0.98	0.98
Full Feature Mapping - Macro Average			0.80	0.71	0.73
- Weighted Average	0.9809	0.0589	0.98	0.98	0.98

Table 1. Experiment Confusion Matrix

The above results demonstrate an increase in performance of the trained model on most aspects of the confusion matrix. Of particular interest is the heightened accuracy rate of the Feature mapping method. This increase in the performance aligns with expectations, since while no new features are input into the data, it is processed in a way that is most suitable for the CNN model.

The difference in the accuracy values between the control group and full mapping methods indicates feature mapping assists the trained model to make accurate predictions. Lower macro average precision indicates that even though nominally the weighted performance is similar, overall the mapping assisted in the model making only accurate estimates of the data, reinforced by the lower loss. Recall and f1-score both maintain a similar macro average, with the recall macro being the only variable where the control group performed better, indicating a better ability to identify all the classes. Weighted average for both is comparable, but still higher, thus indicating a marginal improvement once the model is trained.

The feature mapping method demonstrated an improved performance across most dimensions of measurement, and it also demonstrated that sequencing of features had a negligible effect on the performance of the model. This can be attributed to only representing a portion of information gained from mapping the features to a Cartesian plane. The DeepInsight Sharma et al. (2019) methodology does not elaborate on how the different pixel dimensions affect performance, which seems to be the case, as can be seen by the increased performance in the reduced case. This indicates that contrary to expectations, while the positioning accuracy does impact performance of the models, the minimisation of null values within the data seems to impact more when data is positioned accurately - explaining how despite the minor loss of position accuracy when certain features had the same pixel value and thus were averaged, the reduced matrix performed better in all aspects except the precision. This relationship could be investigated further to determine to what extent the ratio between overlapping features and null values is the cause of the performance difference.

4 CONCLUSIONS

Fundamentally, the need for a better intrusion detection system arises from the increasing strain that contemporary attacks threaten to put on network systems. This need cannot be satisfied as long as hostile actors target networks, as they will continue to invent novel attack techniques and vectors. As result, the need to innovate the detection systems arises.

To further expand on material presented here, an interesting phenomenon of the the DeepInsight framework for data mapping was revealed - there seems to be a link between the performance of the model, the degree to which features overlap within the mapping, and the volume of null values. This relationship can be further investigated to determine the ideal pixel dimensions for optimal performance.

REFERENCES

Axelsson, S. (2000), Intrusion detection systems: A survey and taxonomy, Technical report, Technical report.

Boukhris, I., Elouedi, Z. & Ajabi, M. (2017), 'Toward intrusion detection using belief decision trees for big data', *Knowledge and Information Systems* **53**(3), 671–698.

Cannady, J. (1998), Artificial neural networks for misuse detection, *in* 'Proceedings of the 1998 National Information Systems Security Conference (NISSC'98)', pp. 443–456.

Chauhan, S., Vig, L., De Filippo De Grazia, M., Corbetta, M., Ahmad, S. & Zorzi, M. (2019), 'A comparison of shallow and deep learning methods for predicting cognitive performance of stroke patients from mri lesion images', *Frontiers in neuroinformatics* **13**, 53.

Denning, D. E. (1987), 'An intrusion-detection model', *IEEE Transactions on software engineering* (2), 222–232.

Diro, A. A. & Chilamkurti, N. (2018), 'Distributed attack detection scheme using deep learning approach for internet of things', *Future Generation Computer Systems* **82**, 761–768.

Elmrabit, N., Zhou, F., Li, F. & Zhou, H. (2020), Evaluation of machine learning algorithms for anomaly detection, *in* '2020 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)', IEEE, pp. 1–8.

Kim, D. S. & Park, J. S. (2003), Network-based intrusion detection with support vector machines, *in* 'International Conference on Information Networking', Springer, pp. 747–756.

Manikopoulos, C. & Papavassiliou, S. (2002), 'Network intrusion and fault detection: a statistical anomaly approach', *IEEE Communications Magazine* **40**(10), 76–82.

Maseer, Z. K., Yusof, R., Bahaman, N., Mostafa, S. A. & Foozy, C. F. M. (2021), 'Benchmarking of machine learning for anomaly based intrusion detection systems in the cicids2017 dataset', *IEEE Access* **9**, 22351–22370.

Schölkopf, B., Smola, A. & Müller, K.-R. (1998), 'Nonlinear component analysis as a kernel eigenvalue problem', *Neural computation* **10**(5), 1299–1319.

Sharafaldin, I., Lashkari, A. H. & Ghorbani, A. A. (2018), 'Toward generating a new intrusion detection dataset and intrusion traffic characterization.', *ICISSp* **1**, 108–116.

Sharma, A., Vans, E., Shigemizu, D., Boroevich, K. A. & Tsunoda, T. (2019), 'Deepinsight: A methodology to transform a non-image data to an image for convolution neural network architecture', *Scientific reports* **9**(1), 1–7.

Van der Maaten, L. & Hinton, G. (2008), 'Visualizing data using t-sne.', *Journal of machine learning research* **9**(11).

Zanero, S. & Savaresi, S. M. (2004), Unsupervised learning techniques for an intrusion detection system, *in* 'Proceedings of the 2004 ACM symposium on Applied computing', pp. 412–419.

Zheng, W.-F. (2020), Intrusion detection based on convolutional neural network, *in* '2020 International Conference on Computer Engineering and Application (ICCEA)', IEEE, pp. 273–277.