ACTA INFOLOGICA 2023;7(1):165-172
dergipark.org.tr/acin

DOI: 10.26650/acin.1282824

RESEARCH ARTICLE

Malware Detection in Forensic Memory Dumps: The Use of Deep Meta-Learning Models

Adli Bellek Dökümlerinde Kötü Amaçlı Yazılım Tespiti: Derin Meta Öğrenme Modellerinin Kullanılması

Yalçın Özkan[1]

[1](Assist. Prof. Dr.) Istinye University, Faculty of Economics, Administrative and Social Sciences, Department of Management Information Systems Istanbul, Turkiye

ORCID: Y.Ö. 0000-0002-3551-7021

Corresponding author:
Yalçın ÖZKAN
Istinye University, Faculty of Economics, Administrative and Social Sciences, Department of Management Information Systems Istanbul, Turkiye
E-mail address: yalcin.ozkan@istinye.edu.tr

Submitted: 13.04.2023
Revision Requested: 28.04.2023
Last Revision Received: 29.04.2023
Accepted: 05.05.2023
Published Online: 20.06.2023

Citation: Ozkan, Y. (2023). Malware detection in forensic memory dumps: The use of deep meta-learning models. *Acta Infologica, 7*(1), 165-172. https://doi.org/10.26650/acin.1282824

ABSTRACT

The present study aimed to design a high-performance deep meta-learning model that could be utilized in classification predictions using forensic memory datasets and propose a framework that would ensure the generalization and consistency of the predictions with the help of this model. To achieve this aim, a dataset containing malware and obtained from forensic memory dumps was addressed. First, it was subjected to the classification process with a deep learning algorithm, and a predictive model was acquired. The predictive model was found to have an accuracy metric of 98.25%. In addition to this finding, a meta-learning model consisting of five different models with the same hyperparameters was created. The accuracy of the obtained meta-model was computed as 97.69%. With the thought that this model would reduce the prediction variance and thus the predictive model could be generalized, it was ensured to be run 5 times in a row. As a result of this process, the prediction variance, indicating a very small change, was calculated as 0.000012. Accordingly, considering the acquired performance value, it can be determined that high performance is achieved in malware detection, and thus what hyperparameters ensure success can be revealed. If deep learning methods are used as a single model, the problem is that the variance between the predictions is large due to its stochastic structure. To avoid such drawbacks, a deep meta-learning model using the same parameters was designed instead of a deep learning model comprising a single model, and considerably smaller variance values were achieved, thus providing generalized and consistent predictions.
Keywords: Forensic memory, cyber security, deep learning, meta-learning

ÖZ

Bu çalışmada adli bellek veri kümelerinden yararlanılarak, sınıflandırma öngörülerinde kullanılabilecek yüksek performanslı bir derin meta öğrenme modelinin tasarlanması ve bu model yardımıyla öngörülerin genelleştirme ve tutarlılığını sağlayacak bir çerçevenin önerilmesi amaçlanmaktadır. Bu amaca ulaşabilmek için, kötü amaçlı yazılımları içeren ve adli bellek dökümlerinden elde edilen bir veri kümesi ele alınarak önce derin öğrenme algoritması ile sınıflandırma sürecine tabi tutuldu ve bir öngörü modeli elde edildi. Öngörü modelinin %98,25lik bir doğruluk metriğine sahip olduğu görülmüştür. Bu bulgunun yanı sıra, aynı hiper parametrelere sahip 5 ayrı modelden oluşan bir meta öğrenme modeli oluşturulmuştur. Elde edilen meta modelin doğruluğu %97,69 olarak hesaplandı. Bu modelin öngörü varyansını azaltacağı ve böylece öngörü modelini genelleştirilebileceği düşüncesiyle ardı ardına 5 kez çalıştırılması sağlandı. Bu işlem sonucunda çok küçük bir değişime işaret eden öngörü varyansı 0,000012 olarak hesaplandı. Sonuç olarak, elde edilen performans değeri göz önüne alındığında, kötü amaçlı yazılım tespitinde yüksek bir performansın elde edildiği ve böylece başarıyı sağlayan hiper parametrelerin neler olduğu belirlenebilmektedir. Derin öğrenme yöntemlerinin tek model olarak kullanılması durumunda, stokastik bir yapıya sahip olması nedeniyle öngörüler arasındaki varyansın büyük olması sorunuyla karşılaşılmaktadır. Bu tür sakıncaları önlemek üzere, tek modelden oluşan derin öğrenme modeli yerine, aynı parametreleri kullanan bir derin meta öğrenme modeli tasarlanarak çok daha küçük varyans değerlerine ulaşılmış, böylece genelleştirilmiş ve tutarlı öngörüler üretilmesi sağlanmıştır.
Anahtar Kelimeler: Adli bellek, siber güvenlik, derin öğrenme, meta öğrenme

1. INTRODUCTION

Installing malware in a computer's memory significantly damages computer systems and information security. Hence, it is understood that detecting malware and eliminating threats are important to prevent the hacking of computer user data, credentials, and other important information. To this end, memory analysis can enable the analysis of volatile data in a computer's memory. As with hard drive data, these data can be accessed by forensic memory analysis to investigate and identify attacks or malicious behaviors that do not leave easily detectable traces (Sihwail & Omar & Zainol, 2021). Viruses can be listed among examples of malware. Furthermore, trojans, worms, and spyware can be added to this category. Among them, computer viruses can cause significant destruction by destroying files on infected computers. If the attacker's goal is to collect data from computers, he may aim to reach financial information, such as bank and credit cards, by infecting the target computers with malware (Christensson, 2006).

Studies that address threats and propose solutions in this regard demonstrate that machine learning techniques can be used to detect malware. Various studies stress that machine learning techniques provide numerous advantages and these techniques can produce faster, more accurate, and more effective results than conventional attack prevention methods. However, it can be indicated that these techniques pose some other difficulties. For example, it has been stated that difficulties such as data quality, data size, data integrity, and other ethical and legal issues can hinder the effective use of machine learning techniques (Qadir & Noor).

Sihwail et al. (2019) used datasets acquired on the basis of memory analyses to detect malware, and classification was made through machine learning. It was found that a performance level of 98.5% was achieved by applying the support vector machine algorithm, considered among the conventional classification methods. Additionally, a false positive rate, revealing false alarms, was obtained as 1.7%. However, due to the limitations of conventional classification methods, the tendency to turn to the advantages of deep learning algorithms has begun. Deep learning algorithms in particular, such as convolutional neural networks (CNN) and long short-term memory (LSTM), can be preferred to classify memory dumps and extract important information (Yang et al., 2021; Karamitsoz et al., 2020).

Both traditional and deep learning models can be preferred in the analysis of memory dumps. In a study conducted in 2022, Dener et al. (2022) made a classification using the Random Forest, Decision Tree, Gradient Boosted Tree, Logistic Regression, Naive Bayes, Linear Vector Support Machine, Multilayer Perceptron, Deep Feed Forward Neural Network, and Long Short-Term Memory (LSTM) algorithms and compared their performances. As a result of this classification study, it was seen that the performance of machine learning algorithms approached 99.97%.

In machine learning, meta-learning refers to learning algorithms that learn from other learning algorithms. Meta-learning means the use of machine learning algorithms that learn how best to combine predictions from other machine learning algorithms in the ensemble learning field (Brownlee, 2021). Another purpose of meta-learning is to train a model on various learning tasks so that it can solve new learning tasks using a small number of training examples (Finn et al., 2017). In the present study, a framework on how to acquire both high-performance and generalized and consistent predictions with deep learning and deep meta-learning methods is proposed by considering memory dump data.

2. MATERIALS AND METHODS

2.1. Deep learning networks

Artificial neural networks are among the machine learning methods inspired by biological neural networks and used in solving numerous problems. Artificial neural networks are trained on datasets and perform operations such as classification and prediction development by identifying patterns among data. Artificial neural networks are preferred in machine learning, particularly to obtain high-performance predictions. The artificial neural network model consists of three interconnected layers, called input, hidden, and output. Such networks contain interconnected neural chains forming the neural architecture. If the number of hidden layers exceeds one, the artificial neural network is called a "deep learning network." As seen in a very small size example in Figure 1, a deep neural network can consist of an input, two outputs, and three hidden layers.

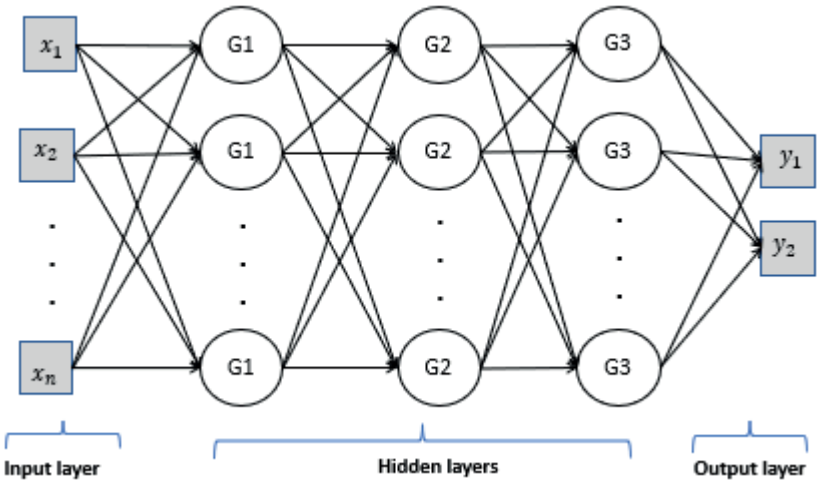


Figure 1. Deep learning classification network

The neurons in deep learning networks are formed as a result of the interconnection of the nerves indicated in Figure 2. A nerve is in contact with all neurons in the next layer. Within the framework of this connection, there is data transfer between the nerves. Every relationship between neurons is provided by numerical values called "weights." Each connection has a numerical weight. Weights are parameters that the neurons in neural networks use when performing a process. The weights of a neuron multiplied by the input signal are an important factor determining the neuron's output signal. During the training of the network, it is essential to set the weights correctly in order to improve the network's performance. To generalize, the input value of a nerve is obtained with the help of equation (1). While calculating the inputs, the used $w_1, w_2, \dots, w_n$ values denote the weights, and $x_1, x_2, \dots, x_n$ values represent the input data. The expression b_1 in the equation is considered the "threshold value."

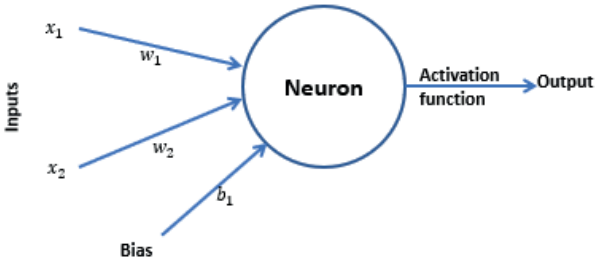


Figure 2. The behavior of a neuron in forward propagation

$$Input = w_1x_1 + w_2x_2 + \cdots + w_nx_n + b_1 \quad (1)$$

The output value is calculated after the total input values for a neuron are computed. The outputs can be expressed as a sigmoid activation function using the inputs, as specified in equation (2), and, thus, a conversion operation is carried out.

$$Output = \frac{1}{1+e^{-Input}} \quad (2)$$

The specified operations are calculated one by one by following the connections for each neuron. Thus, "forward propagation" on the network is performed. Forward propagation is the calculation of outputs by processing the input data of the network over the connections between the neurons in the layers. "Backpropagation" ensures that the weights are updated to minimize the error function of the network.

Backpropagation is performed based on an error function. The error function, also known as the loss function, is expressed as L_{total} and measures the difference between the outputs produced by the network and the actual outputs. The backpropagation method is employed to minimize this error function by changing the weights and threshold values. The error function

measures how close the network is to its intended output. The said function is among the most important factors that determine the targeted performance in network training. The selection of the error function significantly impacts the success of the network in the training process. Mean squared error, cross-entropy, and log loss are among the most common loss functions. After the error function value is computed, the total loss amount is distributed to all weights in the network. To this end, in order to determine the effect of the change to be made in each weight w_i on the total error, its derivatives are calculated according to the mentioned weight and multiplied by η, the learning rate, and thus the amount of change is calculated. This amount is updated by subtracting it from the previous weight, as seen in formula (3).

$$w_{\mathrm{i}}^{new} \longleftarrow w_i - \eta \frac{\partial L_{total}}{\partial w_{\mathrm{i}}} \qquad (\mathrm{i} = 1,2,\ldots,\mathrm{n}) \tag{3}$$

2.2. Deep meta-learning

Deep learning neural networks have a nonlinear structure. They offer more flexibility and can be scaled proportionally to the amount of training data available. However, a disadvantage of this flexibility is that they learn through a stochastic training algorithm. The use of stochastic models such as deep learning means that they are sensitive to the characteristics of the training data and can find a different weight set each time they are trained, and accordingly, they can produce different predictions. Since the prediction variance acquired in deep learning models is high, this variance can be reduced using "meta-learning" methods (Brownlee, J. 2021).

A deep learning-based meta-learning model can be created, as shown in Figure 3, for classification purposes. To this end, the raw dataset is first passed through the preprocessing stage. Afterwards, the test and training datasets are randomly generated. With the training data, a model is obtained in accordance with the deep learning algorithm, and the accuracy value is computed using this model together with the test data. Following the preprocessing stage, a meta-learning model comprising n models is obtained. To find the performance of meta-learning with n models obtained, datasets that consist of the predictive values of each model constituting it are used. These data are combined, and the final performance is acquired using another machine learning model.

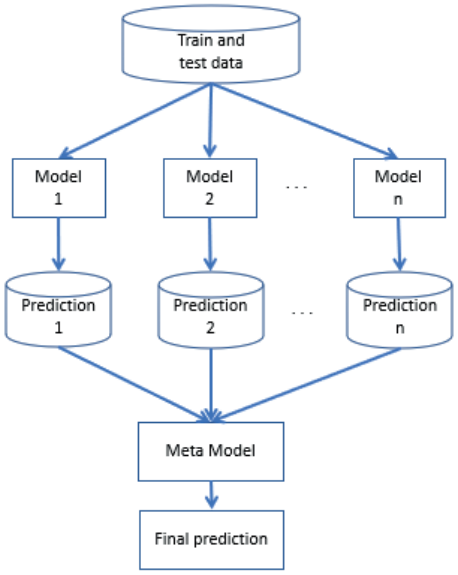


Figure 3. Meta-learning model

2.3. Memory analysis

Memory analysis is generally based on the principle of obtaining the current state of system memory as a snapshot file, also known as a "memory dump." The mentioned capture process is performed by running special software. This file obtained can be later moved out of the system, and analyses can be conducted on it. Support is received from analyzers to perform such operations. The said software can convert data to a CSV file. In this way, it becomes possible to apply machine learning

algorithms on the obtained files. Analyzers not only capture malware footprints but also have additional features that can be used to extract the hidden original code (Lashkari et al., 2020).

3. RESULTS

3.1. Dataset

A dataset must first be provided, and a model must be trained with this dataset to create predictive models in machine learning applications. To achieve this, a dataset CIC-MalMem-2022 was used in the current study (UNB, 2023). The dataset was balanced to comprise 50% malicious memory dumps and 50% benign memory dumps. The table below contains a breakdown of the malware families included in the dataset. The dataset contains a total of 58,596 records, of which 29,298 are normal and 29,298 are malicious (Carrier et al., 2022). Table 1 lists the malware included in the dataset.

Table 1
Distribution of malware in the dataset

Software category	Software families	Number of observations
Trojan	Zeus	1950
	Emotet	1967
	Refroso	2000
	Scar	2000
	Reconyc	1570
Spyware	180Solutions	2000
	CoolWebSearch (CWS)	2000
	Gator	2200
	Transponder	2410
	TIBS	1410
Ransomware	Conti	1988
	MAZE	1958
	Pysa	1717
	Ako	2000
	Shade	2128
Total		29298

3.2. Preprocessing

Preprocessing activities were carried out before proceeding to creating the deep learning model. In this regard, one attribute that was thought to be not useful in model creation was extracted from the dataset, and spaces in the variable names were eliminated. To apply the classification model on the data set, 70% of the total data was separated as training data, and the remaining 30% was separated as test data using the random sampling method. Thus, 41017 observations were allocated for the training of the deep learning model, and 17579 observations were allocated for testing. In this study, data balancing processes were not applied since an equality was provided in terms of the distribution of classes. Of the observations reserved for training, 20516 consist of normal control observations, whereas the remaining 20501 consist of malware observations. Normal observations are labeled “Benign” and malware observations are labeled “Malware”.

3.3. Deep learning model architecture

The stage of designing the deep learning model is proceeded to after the preprocessing steps are completed. At the said stage, the layers that would form the model were prepared in the Python environment in line with the principles determined by the Keras library (Chollet et al., 2015) located on the TensorFlow website (Abadi et al., 2014). In this model, two hidden layers were determined apart from the input and output layers. As seen in Figure 4, the nerve numbers and activation functions of each layer were identified. The “softmax” activation function was selected in the output layer, and the “relu” activation function was selected in all other layers.

The deep learning architecture reveals the structure where the deep learning algorithm will be applied. The training model was identified in accordance with this structure. In this application, the “Keras” library was used for model training. Eighty epochs were applied in the development of the model. Of the training data, 20% was reserved for the validation process to

be used during model creation. While designing the model, “categorical_crossentropy” was selected as the loss function, “Adam” was selected as the optimizer, and the “accuracy” parameter was selected as a metric.

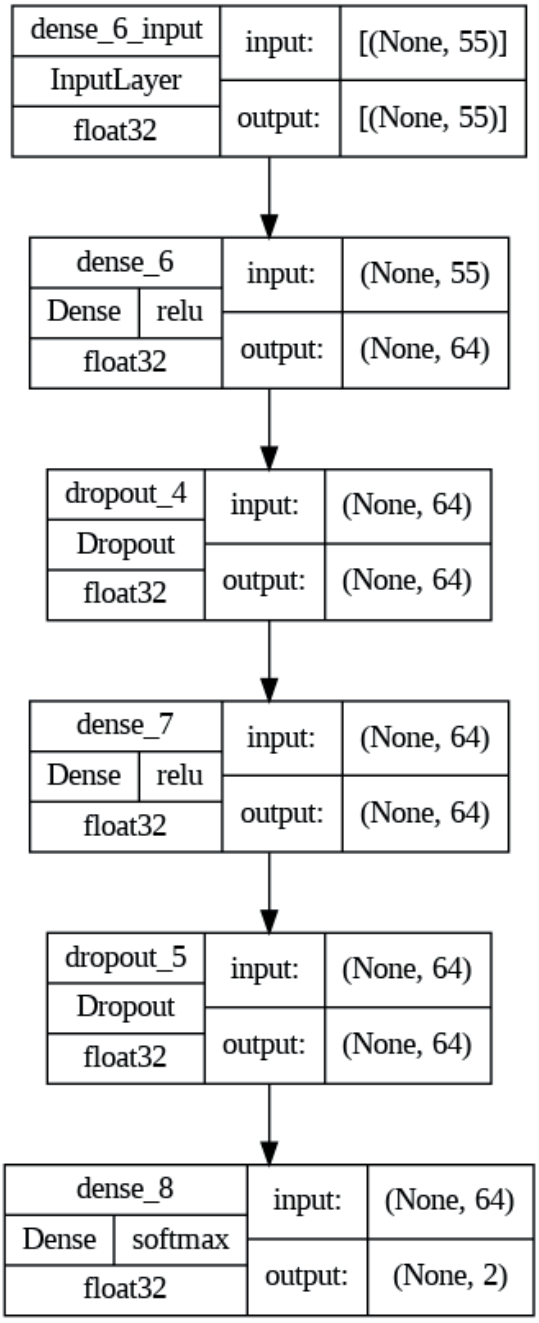
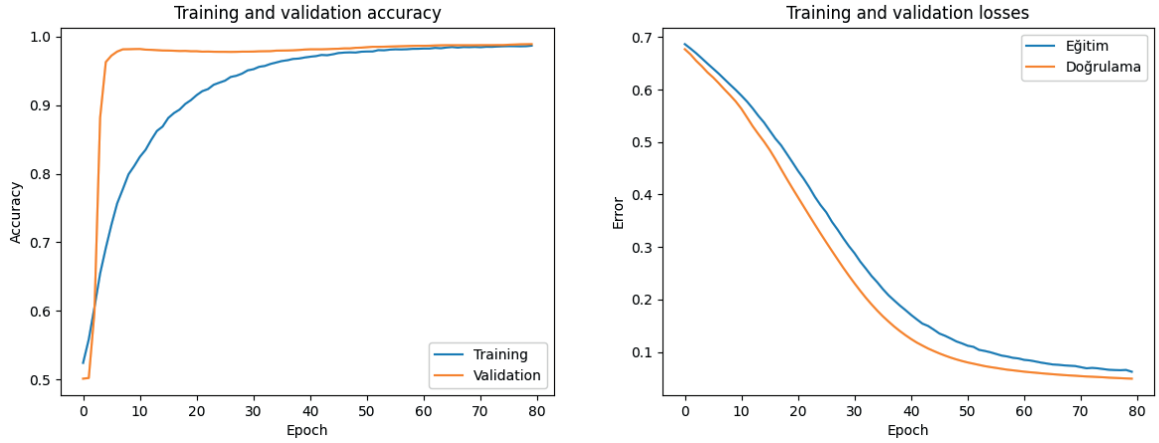


Figure 4. Deep learning architecture

3.4. Model performance

The model was started to be trained after the deep learning architecture in Figure 4 was established and its parameters were identified. At this stage, it was determined that the model initially encountered an “overfitting” problem after a few iterations (epochs). The learning rate of the model was reduced to 0.000001 to solve this problem. During the model’s training, the performance values and error amounts acquired in each iteration were computed, and the graphs displayed in Figure 5 were drawn. Changes in the model’s performance can be observed in the first graph (a), while the change in losses can be observed in graph (b). The classification performance was calculated as 98.25% using the test data of the trained model.



(a) (b)

Figure 5. Performance of the training model and error distribution.

3.5. Meta-learning model

The meta-model addressed in this study was prepared based on the components in Figure 3. To acquire the meta-model, a deep learning model was first prepared. Afterwards, an ensemble of 5 deep learning models with the same hyperparameters was prepared. Predictions were obtained using the test data for each ensemble member. These predictions were combined and used as input for a regression model. In conclusion, the performance of the regression model was accepted as the performance of the deep meta-model. The performance of the first meta-model was calculated as 0.9769 in terms of accuracy. Furthermore, the variance between these predictive performance values was found to be 0.0008. The "Model 1" column in Table 2 explains these operations.

By repeating the same operations, the meta-learning model was run 4 more times, and similar operations were specified in separate columns for model 2, model 3, model 4, and model 5. The purpose of these model iterations is to compare the prediction variances of the final model obtained at different times. The meta-model performance was computed as 0.9764 for model 2, 0.9771 for model 3, 0.9696 for model 4, and 0.9780 for model 5. The variance between the performances acquired as a result of running the deep meta-models 5 times was 0.000012.

Table 2
Deep meta-learning models and performance values for each element

Model elements	**Meta 1**	**Meta 2**	**Meta 3**	**Meta 4**	**Meta 5**
1	0.9925	0.9864	0.9445	0.9842	0.9634
2	0.9209	0.9858	0.9143	0.9280	0.9916
3	0.9683	0.9841	0.9662	0.9917	0.9899
4	0.9859	0.9299	0.9685	0.9020	0.9670
5	0.9890	0.9745	0.9885	0.9852	0.9896
Inter-model variance	0.0008	0.0006	0.0008	0.0016	0.0002
Meta-model performance	0.9769	0.9764	0.9771	0.9696	0.9780

4. CONCLUSION

This study aimed to develop a high-performance and generalized classification framework by considering a dataset that included normal and malware observations. Initially, a deep learning model was developed based on the forensic memory dataset acquired from memory dumps, and a 98.25% accuracy value was obtained as predictive performance. Although this is a high-performance value, lower or higher performance values can be obtained in the next run. Because of the stochastic nature of deep learning models, different results may be encountered at each run of the developed model.

A meta-learning model was developed in the present study in order to have low variance between prediction performances, in other words, to obtain generalized predictions. Instead of calculating the model performance based on a single model, a common performance acquired from an ensemble of five models was used, and a meta-learning architecture developed to this end was applied. The prediction variances of the obtained meta-model were computed as 0.000012. This result is considerably smaller than the variance of each of model 1, model 2, model 3, model 4, and model 5 meta-models, showing that the developed deep meta-model framework can be used to obtain high-performance and low-variance generalized predictive models.

Peer-review: Externally peer-reviewed.
Conflict of Interest: The author has no conflict of interest to declare.
Grant Support: The author declared that this study has received no financial support.

REFERENCES

1. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., ... & Zheng, X. (2016). TensorFlow: Large-scale machine learning on heterogeneous distributed systems. In 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI) (pp. 265-283).
2. Brownlee, J. (2021, April 26). *What is meta-learning in machine learning?* MachineLearningMastery.com. Retrieved from https://machinelearningmastery.com/meta-learning-in-machine-learning/.
3. Carrier, T., Victor, P., Tekeoglu, A., & Lashkari, A. (2022). *Detecting obfuscated malware using memory feature engineering.* Proceedings of the 8th International Conference on Information Systems Security and Privacy. https://doi.org/10.5220/0010908200003120.
4. Chollet, F., & others. (2015). *Keras*. GitHub. Retrieved from https://github.com/fchollet/keras
5. Christensson, P. (2022, Nov 19). *Malware Definition*. Retrieved from https://techterms.com
6. Dener, M., Ok, G., & Orman, A. (2022). *Malware detection using memory analysis data in Big Data Environment*. Applied Sciences, 12(17), 8604, https://doi.org/10.3390/app12178604.
7. Finn C., Abbeel P., & Levine S. (2017). *Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks.* https://arxiv.org/abs/1703.03400.
8. Karamitsos, I., Afzulpurkar, A. & Trafalis, T. (2020) *Malware Detection for Forensic Memory Using Deep Recurrent Neural Networks*. Journal of Information Security, 11, 103-120. doi: 10.4236/jis.2020.112007.
9. Lashkari, A. H., Li, B., Carrier, T. L., & Kaur, G. (2021). *VolMemLyzer: Volatile memory analyzer for malware classification using feature engineering.* 2021 Reconciling Data Analytics, Automation, Privacy, and Security: A Big Data Challenge (RDAAPS). https://doi.org/10.1109/rdaaps48126.2021.9452028.
10. Qadir, S., & Noor, B. (2021). *Applications of machine learning in Digital Forensics.* 2021 International Conference on Digital Futures and Transformative Technologies (ICoDT2). https://doi.org/10.1109/icodt252288.2021.9441543.
11. Sihwail, R., Omar, K., & Ariffin, K. (2021). *An effective memory analysis for malware detection and classification*. Computers, Materials & Continua, 67(2), 2301–2320. https://doi.org/10.32604/cmc.2021.014510.
12. Sihwail, R., Omar, K., & Ariffin, K., Al-Afghani, S. (2019). *Malware detection approach based on artifacts in memory image and dynamic analysis*. Applied Sciences, 9(18), 3680. https://doi.org/10.3390/app9183680.
13. UNB datasets. (2023, March 31). *Malware Memory Analysis: CIC-MalMem-2022* Retrieved from https://www.unb.ca/cic/datasets/malmem-2022.html.
14. Wikimedia Foundation. (2022, April 20). *Memory forensics*. Wikipedia. Retrieved from https://en.wikipedia.org/wiki/Memory_forensics.
15. Yang, L., Chen, Y., & Chen, X. (2021). *Forensic Memory Analysis Using Deep Learning Techniques*. IEEE Access, 9, 137108-137117. doi: 10.1109/ACCESS.2021.3115735.