Utilizing InfoGAN and PE Header Features for Synthetic Ransomware Image Generation: An Experimental Study

Viet Trung Le[1], Phuc Hao Do[2], Sylvestre Uwizeyemungu[3], Thang Le-Dinh[3], and Tran Duc Le[1,3](✉)

[1] University of Science and Technology–The University of Danang, Danang, Vietnam
letranduc@dut.udn.vn
[2] Da Nang Architecture University, Danang, Vietnam
[3] Université du Québec à Trois-Rivières, Trois-Rivières, Québec, Canada

Abstract. Ransomware is a growing threat in the digital world, posing significant challenges to malware detection systems due to its rapidly evolving nature. Addressing this issue requires innovative approaches and robust datasets for training advanced machine learning models. This paper presents a method for generating synthetic ransomware image samples using the InfoGAN model in conjunction with Portable Executable (PE) Header features. The generated samples mimic real ransomware's structural characteristics, enhancing their realism and utility for model training. A detailed implementation of the Information Maximizing Generative Adversarial Network (InfoGAN) model and an evaluation of its performance in generating high-quality ransomware images are provided. The utility of the generated samples is further validated through classification experiments using a Convolutional Neural Network (CNN) model. The results demonstrate the promise of the proposed method in enhancing malware detection capabilities, particularly in the context of ransomware.

Keywords: GAN · InfoGAN · PE Header · CNN

1 Introduction

The integration of Artificial Intelligence (AI) is increasingly becoming a crucial determinant in the growth of any nation's digital economy, a subject of discussion in academia for many years [1]. In cybersecurity, recent years have seen concerted efforts toward developing AI-centric solutions [2]. One specific area in the cybersecurity field, malware analysis, stands to gain significantly from this computational assistance [3].

Malware analysis involves studying malware's behavioral patterns to detect and neutralize it. However, malware analysis faces challenges, such as the need for more automation and integrated tools, which makes tracking malware patterns over time and identifying connections and similarities among different malware families in large datasets difficult [4]. AI can deal with this difficulty.

T. K. Dang et al. (Eds.): FDSE 2023, CCIS 1925, pp. 226–239, 2023.
https://doi.org/10.1007/978-981-99-8296-7_16

However, real-world datasets are typically chaotic, disorganized, and unstructured, posing challenges to AI performance [5]. The success of an AI model heavily relies on the quantity, quality, and relevance of the dataset, but achieving the right balance is a challenging task. For specific problem statements, assembling a domain-specific dataset, cleaning, visualizing, and comprehending its relevance become essential to obtain desired outcomes [6].

One contemporary approach to address this issue involves employing AI models and algorithms to emulate real-world datasets. Such datasets are referred to as synthetic datasets. Generative Adversarial Networks (GANs) [7] are among the premier methods for creating these synthetic datasets. Based on neural network models, these architectures generate datasets that closely resemble real-world data. This characteristic renders GANs especially suitable for creating malware samples to serve AI or machine learning models in malware analysis and cybersecurity [8].

In this study, we use image processing capabilities to classify malware, emphasizing ransomware due to its increasingly prevalent and destructive nature [9]. It entails the conversion of malware samples from binary into image form, followed by applying machine learning models and AI methodologies to these converted images, as opposed to the direct application to the actual malware samples. The rationale behind this methodology of image-based analysis arises from several factors. Image representation of binary malware samples facilitates pattern recognition that may remain undetected in raw binary form. The features derived in this manner can be processed more efficiently by image-centric models. Furthermore, this approach offers an added layer of security as it mitigates the risk associated with the direct execution of malware. The method also accommodates applying transfer learning techniques using pre-existing, pre-trained AI models. Image data also offer opportunities for augmentation, thereby enhancing the robustness of the model. Consequently, the model's ability to classify various forms of malware is amplified.

Portable Executable (PE) Header features[1] are pivotal in this research. They represent the structural information of the executable files, providing valuable insights into the underlying behavior of the malware. PE Header features such as the image size, the number of sections, and the characteristics of these sections can serve as significant indicators of malware presence. In our study, these features are utilized as part of the input to the Information Maximizing Generative Adversarial Network (InfoGAN) model, guiding the generation process to produce synthetic ransomware samples that mimic the structural characteristics of the original ransomware. By doing so, we aim to enhance the realism and quality of the generated samples, thereby improving their utility for model training and testing in ransomware detection tasks.

Our main contributions are as follows:

- Utilizing the InfoGAN model to generate synthetic ransomware images from a specific ransomware family;
- Using Portable Executable features as input for generating the ransomware images with custom features;
- Validating the synthetic ransomware samples' utility through classification experiments with a CNN model.

[1] https://learn.microsoft.com/en-us/windows/win32/debug/pe-format.

The remaining part of this paper encompasses the presentation of several relevant studies in Sect. 2, followed by an overview of the GAN and InfoGAN models in Sect. 3. Section 4 will introduce the methodology, while the evaluation and results will be elucidated in Sect. 5. The final section will provide the conclusions.

2 Related Works

This section presents an extensive review of existing studies that align closely with the subject matter of this paper. It aims to explore and discuss key research efforts, methodologies, and outcomes in synthetic dataset generation using AI, with particular emphasis on applying GAN.

Moti et al. [10] developed a deep generative adversarial network to create signatures for potential future malware, enhancing classifier training. The method uses executable file headers and a neural network for feature extraction, improving classification accuracy by at least 1%. However, its effectiveness may be limited due to the diverse range of malware and insufficient header information. Classification is done using random forest, SVMs, and logistic regression. In the study [11], Ding et al. introduced a method for creating adversarial malware using feature byte sequences. The method outperforms random and gradient-based techniques but is effective only for CNN-based detectors and needs prior algorithmic knowledge.

In the study [12], Lu and Li used the Deep Convolutional Generative Adversarial Network (DCGAN) to create synthetic malware samples, boosting ResNet-18's accuracy by 6%. However, the method requires large datasets and lacks comparison with other classifiers.

In the study [13], Singh et al. developed a GAN-based model for creating labeled malware image datasets to improve classifier training. The method benefits from incorporating domain knowledge but requires such knowledge, which may not always be accessible. The study is limited to a single dataset, not covering all malware types.

In the study [14], Gao et al. presented the MaliCage framework for accurate malware classification. It has three main components: a packer detector, a deep neural network-based malware classifier, and a packer generative adversarial network. The framework identifies and classifies packed and unpacked malware using synthetic samples to improve training and accuracy. The evaluation shows that it effectively mitigates the impact of packed malware on machine learning models.

The use of Information Maximizing GAN (InfoGAN) [15] in creating malware images is an emerging field that improves malware detection and classification. InfoGANs generate images with specific features, allowing for exploring and analyzing different malware variants. This approach excels at identifying unique malware characteristics that traditional methods may miss, such as specific encryption or packing techniques.

3 Overview of GAN Models

Generative Adversarial Networks (GANs) consist of a generator network (G) and a discriminator network (D).

The generator network inputs a random noise vector and generates synthetic data (like images) from this noise. The generator aims to produce data indistinguishable from the real-world data it tries to mimic. On the other hand, the discriminator network takes both real-world data and the synthetic data produced by the generator as input. Its task is to distinguish between real and synthetic data. In other words, it tries to classify whether each input data is real or fake.

The Generator tries to fool the Discriminator by generating increasingly realistic data. In contrast, the Discriminator tries to better distinguish real data from the fake data produced by the Generator. This competition improves both networks, leading to the Generator producing highly realistic data. The mathematical formulation of the GAN model [15] can be written as:

$$\underset{G}{min}\underset{D}{max}V(D, G) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))] \quad (1)$$

Here, $V(D, G)$ is the objective function of the GAN. The first term, $\mathbb{E}_{x \sim p_{\text{data}}(x)}[\log D(x)]$, is the expected log probability that the Discriminator correctly classifies real data (drawn from the true data distribution $p_{\text{data}}(x)$). The second term, $\mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]$, is the expected log probability that the Discriminator correctly classifies synthetic data (generated by passing noise z drawn from the noise distribution $p_z(z)$ through the Generator).

The loss function for the Discriminator can be derived from the objective function and is given by:

$$\mathcal{L}_D = -\mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{z} \sim p_z(z)}[\log(1 - D(G(\boldsymbol{z})))] \quad (2)$$

The loss function for the Generator is:

$$\mathcal{L}_G = -\mathbb{E}_{\boldsymbol{z} \sim p_z(z)}[\log D(G(\boldsymbol{z}))] \quad (3)$$

Figure 1 shows the design of the GAN architecture.

Nevertheless, a conventional GAN is not intended for multiclass data and additional information. Therefore, we aim to employ InfoGAN, an improved version of GAN that incorporates a class label and extra information into the generative model.

InfoGAN aims to make the generated data more interpretable and meaningful. It maximizes the mutual information between a fixed small subset of the GAN's noise variables and the observations. These variables could represent specific, meaningful characteristics of ransomware that we want to vary in a controlled way, such as the type of encryption used, the type of files targeted, or the message displayed.

For the case of generating ransomware samples, the original GAN loss function is modified in InfoGAN to include an additional term that represents the mutual information between the generated ransomware samples and a subset of the input noise variables. It encourages the model to use these variables meaningfully, leading to more interpretably generated ransomware samples.

The objective function of InfoGAN [15] can be written as: $\underset{G}{min}\underset{D}{max}V(D, G) - \lambda I(c; G(z, c))$. Here, $V(D, G)$ is the original GAN objective function. The term $-\lambda I(c; G(z, c))$ encourages the Generator to use the variable c in meaningfully, leading to ransomware samples that vary in a controlled and interpretable way based on c. c is also called the interpretable latent code, learned by InfoGAN. It represents different aspects of the data that the GAN is trying to generate.

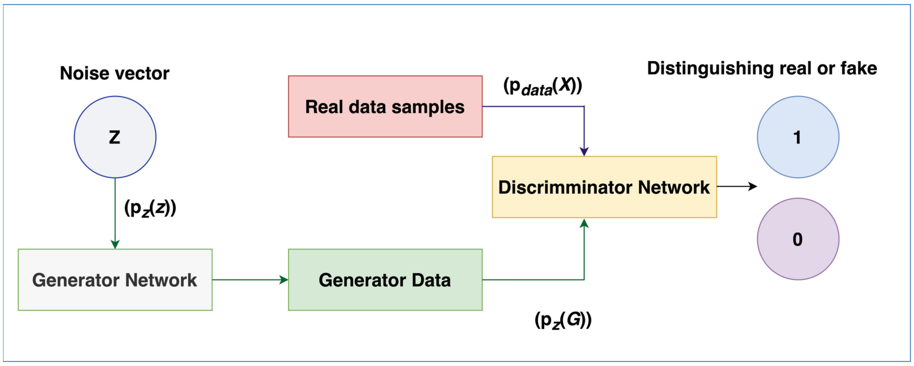

Fig. 1. Design of the GAN architecture.

The parameter λ is a hyperparameter that controls the trade-off between the original GAN objective and the mutual information term. A higher λ places more emphasis on maximizing mutual information.

The mutual information $I(c; G(z, c))$ can be difficult to compute directly, so in practice, an auxiliary distribution $Q(c \mid G(z, c))$ is introduced, and the mutual information is maximized by maximizing a lower bound:

$$I(c; G(z, c)) \geq \mathbb{E}_{c,z}[\log Q(c \mid G(z, c))] + H(c) \tag{4}$$

Here, $H(c)$ is the entropy of c, which is a constant with respect to Q so maximizing this lower bound is equivalent to maximizing the expectation $\mathbb{E}_{c,z}[\log Q(c \mid G(z, c))]$. It can be done using standard backpropagation and gradient ascent, like the rest of the GAN training process.

4 Methodology

The methodology employed in our research is a multi-step process revolving around using the InfoGAN model and PE features. Our approach encapsulates a comprehensive strategy from ransomware collection to testing synthesized images using a CNN classifier. The step-by-step procedure of the methodology is visually depicted in Fig. 2, providing a clear and concise overview of our research design.

In the initial phase, we collected ransomware samples as the primary data for our study. It served as the basis for our analysis and the foundation of our image dataset. Following the data collection, we performed two crucial steps. Firstly, the binary data of the ransomware were converted into images, and secondly, we extracted and selected the Portable Executable (PE) header features.

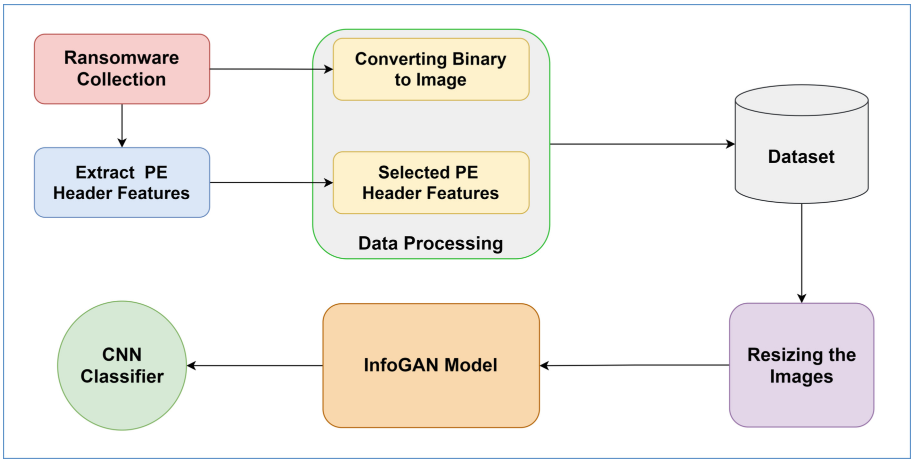

Fig. 2. Research flow.

After the conversion and selection, a dataset of these images for further processing was created. We then resized the images to ensure uniformity and consistency in the data, which is an essential pre-processing step in image analysis and processing tasks.

Having prepared our image dataset, we applied the InfoGAN model with multiple input values. The synthesized images generated by the InfoGAN model were then subjected to a CNN classifier for testing.

The details of each step, including the nuances of the InfoGAN model application and the CNN classifier testing, will be discussed in the subsequent sections.

4.1 Converting Binary to Image

Obtaining a comprehensive dataset of ransomware proved challenging, prompting the decision to create our own dataset, referred to as the "*ransomware dataset*". We manually searched, downloaded, and classified the ransomware samples (only Windows executable binary files) from *Virusbay, Hybrid-analysis, Bazaar, Virusshare, GitHub, and Virustotal*.

Leveraging the *Dataloader* modules provided by *PyTorch*, we extracted images and labels from the ransomware executable files. Furthermore, the transform functions were employed to address our pre-processing requirements effectively.

At this phase, since the ransomware files are executable binaries, they can be converted into images using a similar approach as used in [16]. According to it, the binaries were converted into pixels, as described in Fig. 3. The samples belonging to the same variant will have similar pixel distribution. It should be noted that the pixel distribution may no longer accurately represent the variants of the samples due to the obfuscations introduced. Consequently, the reliability of the results for classifying malware variants may be compromised. To minimize this impact, we tested and removed obfuscated ransomware samples using *PEiD* and *Exeinfo PE* tools.

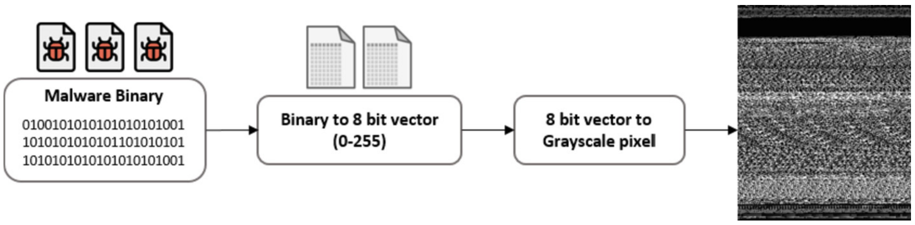

Fig. 3. Process of converting the binary file to image.

Figure 4 shows the distribution of the ransomware dataset in each ransomware family.

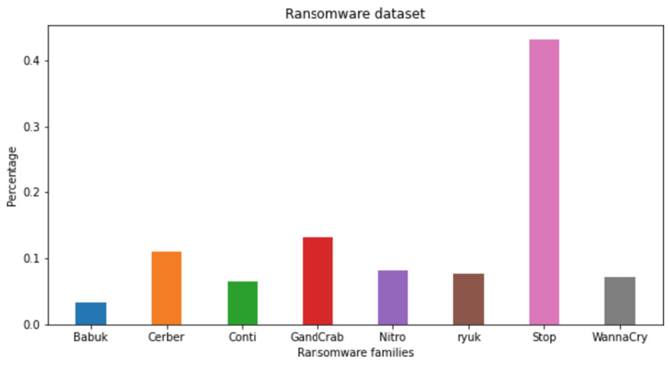

Fig. 4. Distribution of ransomware families in the dataset.

Table 1 presents the number of samples for each family. It is observed that the sample distribution within each family exhibits a relatively balanced distribution. The *Stop* family stands out with a notably higher sample count than the other families. In order to address the inherent data imbalance, we implemented shuffling of the data during the training phase and employed balanced accuracy as the evaluation metric during testing. This approach ensures that each class is equally represented during training, mitigating the potential bias introduced by the imbalanced data distribution. By utilizing balanced accuracy, we account for the disproportionate class sizes and comprehensively assess the model's performance across all classes.

When converting binary executable files into images, the selection of the image width is based on the corresponding file size range (Table 2). This choice aims to represent the original file effectively while balancing visual details and computational efficiency. By associating each file size range with a specific image width, we ensure that the resulting images accurately capture the essence of the files. Narrower images with lower pixel counts are assigned to smaller file sizes, while wider images with higher pixel counts represent larger file sizes. This approach successfully tra a diverse range of executable files into image representations, optimizing memory usage and computational resources.

Upon completing the transformation process, the images were resized to predetermined dimensions to create an input dataset tailored for training the model. This resizing

Table 1. Number of samples in each ransomware family.

Ransomware Families	Number of samples	Ransomware Families	Number of samples
Babuk	43	Nitro	104
Cerber	141	Ryuk	99
Conti	84	Stop	556
GandCrab	170	WannaCry	92

Table 2. Width (in pixel) of image based on the size of the binary.

File Size Range	Image Width	File Size Range	Image Width
<10 kB	32	100 kB–200 kB	384
10 kB–30 kB	64	200 kB–500 kB	512
30 kB–60 kB	128	500 kB–1000 kB	768
60 kB–100 kB	256	>1000 kB	1028

operation enabled us to obtain images that offer various scale views of the underlying data. Figure 5 exemplifies the diverse scale views captured by the images depicting a *Nitro* family.

4.2 Extract PE Header Features and Apply Them to the InfoGAN Model

Portable Executable Header features are a critical aspect of our methodology, serving as a key input for the InfoGAN model in generating synthetic ransomware samples. The PE Header, which forms the structural metadata of the executable files, provides crucial insights into the behavior of the malware. The use of PE Header features in this manner serves a dual purpose. Firstly, it enhances the quality of the generated samples, making them more representative of real ransomware. Secondly, it allows for the exploration of specific features and their variations in the generated samples, aiding in understanding their influence on ransomware detection.

In the first step, we extract the PE Header features from a dataset of known ransomware samples. This process is facilitated using a Python library named *pefile*, which parses the PE Header information from the binary executable files. Table 3 presents some important features of the PE Header derived from an analysis of a *WannaCry* ransomware sample.

Given the extensive number of PE feature values, an evaluation was conducted to assess the influence of these features through various models, thereby determining which values to use for training. Machine learning models, specifically *XGBoost* and *CatBoost*, were employed due to their superior performance in machine learning competitions hosted on Kaggle[2]. These models were used to assess the impact of features on the

[2] https://www.kaggle.com/code/nholloway/catboost-V-xgboost-V-lightgbm.

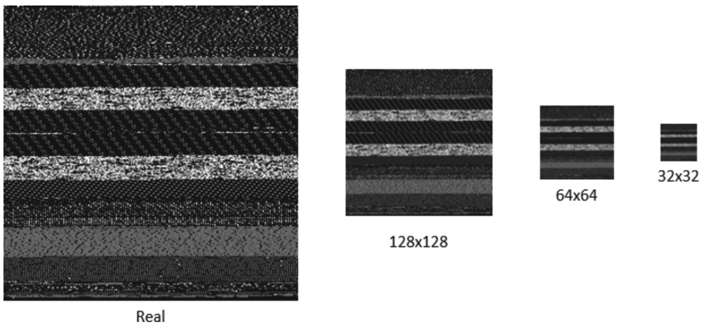

Fig. 5. Images of a *Nitro* family from other scale views.

Table 3. A short list of *Wannacry's* PE features.

PE Features	Size (Bytes)	PE Features	Size (Bytes)
Machine	2	BaseOfCode	4
SizeOfOptionalHeader	2	BaseOfData	4
Characteristics	2	ImageBase	8
MajorLinkerVersion	1	SectionAlignment	4
MinorLinkerVersion	1	DllCharacteristics	2
SizeOfCode	4	CheckSum	4
SizeOfInitializedData	4	FileAlignment	4
SizeOfUninitializedData	4	MajorOperatingSystemVersion	2
AddressOfEntryPoint	4	SizeOfStackCommit	8

ability to classify ransomware families. While the quantity of these features was not extensive, their impact on classification results and model accuracy was significantly superior to other features within the PE header. The average influence of the features was then calculated as presented in Table 4, incorporating the top five features that exert the greatest influence on the model. Nevertheless, in practice, the number of features may vary depending on the computational power of the hardware utilized and desired training time during the model training. The optimal determination of the number of features falls beyond the scope of this study and could be a prospective avenue for future research.

These selected features are then fed into the InfoGAN model as part of the input data, which also includes *Noise code*, *Ransomware Label*. In particular, the selected features are used as conditioning variables in the InfoGAN, influencing the generation process of the synthetic samples. By integrating the PE Header features, the InfoGAN model is guided to generate ransomware samples that appear realistic and mimic the original ransomware's structural characteristics. The model's output will be a ransomware variant learned by the model. The Implementation section will provide a more detailed presentation of the Generator and the overall InfoGAN model. This approach aligns with the primary objective of this study, which is to utilize image-processing capabilities for the classification of ransomware.

Table 4. The average influence of the five greatest features.

Features	Impact level (%)
CheckSum	12.51
SizeOfUninitializedData	11.23
SizeOfStackCommit	8.54
DllCharacteristics	7.35
MinorLinkerVersion	7.21

5 Evaluation and Results

In order to evaluate the image generation capabilities of the model, two groups of experiments are conducted:

- InfoGAN experiments: The application of the InfoGAN model was systematically examined through three distinct experiments, encompassing images of varying sizes: 32 × 32, 64 × 64, and 128 × 128 pixels.
- CNN experiments: The employment of the CNN model to differentiate the real and the synthesized ransomware samples generated by the InfoGAN model and PE header features.

5.1 InfoGAN Experiments

The application of the InfoGAN model was systematically examined through three distinct experiments, encompassing images of varying sizes: 32 × 32, 64 × 64, and 128 × 128 pixels. The dataset images were resized to the respective dimensions in each case to ensure uniformity. The InfoGAN model was trained over 1000 epochs with a batch size of 32, with each training process conducted on *Google Colab Pro* lasting approximately 10 h for the 32 × 32 and 64 × 64 image sets and about 12 h for the 128 × 128 image set.

Table 5 presents the mean loss values for each experiment conducted using InfoGAN. The results demonstrate a notable consistency, independent of the image dimensions.

As the image size increases, the average discriminator loss decreases while the average generator loss rises. This trend suggests that the InfoGAN model was able to generate more refined images as the size of the images increased. The discriminator model, tasked with distinguishing real images from synthesized ones, improved with the increase in image size. Conversely, the generator model, responsible for creating

Table 5. Average loss values of InfoGAN models.

Image Size	Average Discriminator Loss	Average Generator Loss	Average Information Loss
32 × 32	0.1899	0.5865	1.274
64 × 64	0.1212	0.7597	1.275
128 × 128	0.0525	0.9626	1.274

synthetic images, found the task progressively more challenging with the larger image size, as seen by the increase in loss.

Interestingly, the average information loss remained relatively constant across all image sizes. It indicates that the InfoGAN model preserved consistent information across all experiments, regardless of the image size.

These results collectively suggest a trade-off between the generator and discriminator performance as the image size increases. The increasing challenge for the generator, juxtaposed with the improved performance of the Discriminator, underscores the intricate dynamics at play within the GAN model. Despite this, the constancy in the information loss shows the model's stability across different image resolutions, reinforcing the versatility of the InfoGAN model for generating high-quality ransomware images.

Figure 6 demonstrates the visual comparisons of the original and synthesized images that share identical characteristics for the ransomware family, *GandCrab*. In the 32 × 32 pixel scenario context, the resolution appears to be suboptimal across all instances. However, in the cases involving 64 × 64 and 128 × 128 pixels, it becomes discernible that the synthesized samples, as generated by the InfoGAN, manifest a substantial degree of visual similarity to the original images. It indicates that the model can more accurately replicate the original image structure as the resolution increases, thus suggesting its potential efficacy in synthesizing high-resolution images for further studies.

5.2 CNN Experiments

In this section, we try to differentiate between real ransomware images and those synthesized by the InfoGAN model. The CNN model was employed, treating the original and synthesized images as separate classes within multiclass experiments. There are eight classes derived from the dataset. Therefore, this procedure results in 16 classes, each encompassing the eight original families and their corresponding synthesized classes. Subsequent sections will individually explore experiments for image sizes of 32 × 32, 64 × 64, and 128 × 128 pixels. Approximately 100 samples were generated per subclass for each case, resulting in a novel ransomware dataset of 800 synthetic images in conjunction with 1100 original images. Table 6 below delineates the accuracy scores attained on these test sets.

The accuracy is particularly high for ransomware families such as *Babuk*, *Conti*, *GandCrab*, *Nitro*, and *Stop*. These families exhibited accuracy values close to or at 100% in most test scenarios, showcasing the efficacy of both InfoGAN's ability to generate

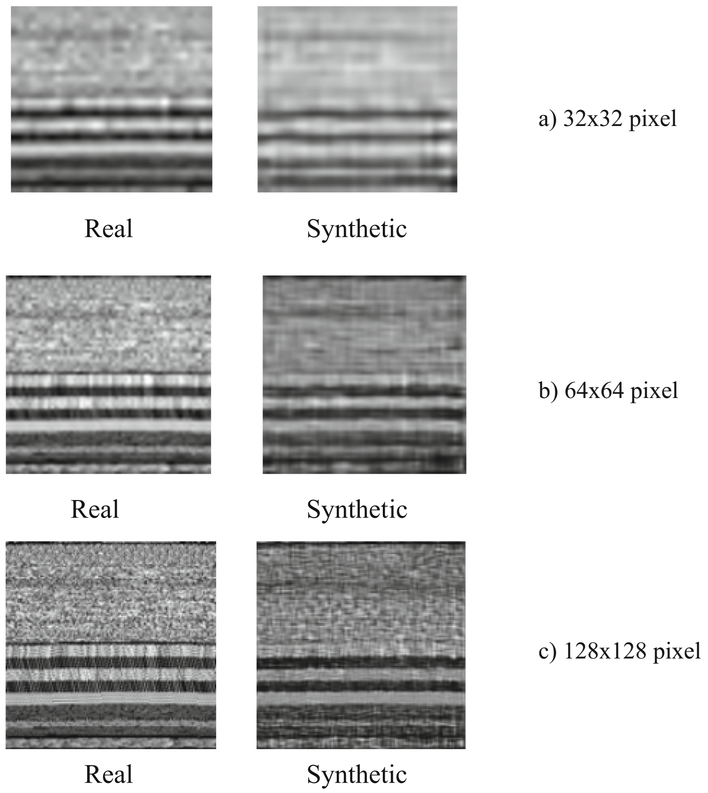

Fig. 6. Real and synthetic samples of the *GandCrap* ransomware family.

Table 6. Accuracy of test sets from prediction of CNNs model.

Class (Label)	Accuracy values of CNN model (%)					
	32 × 32 pixel images		64 × 64 pixel images		128 × 128 pixel images	
	Real	Synthesized	Real	Synthesized	Real	Synthesized
Babuk	100	93.8	100	93.8	100	98.8
Cerber	28.6	96	50	100	66.7	100
Conti	94.4	100	59.1	100	86.4	100
GandCrab	96.8	100	96.8	100	96.6	100
Nitro	100	90	94.7	100	95.2	98.7
Ryuk	33.3	96.4	100	96.4	33.3	96.7
Stop	100	100	95.8	100	97.9	100
WannaCry	91.7	85.7	57.1	100	54.5	100

high-quality synthetic ransomware images and CNN's ability to classify these images accurately.

On the other hand, the ransomware families *Cerber*, *Ryuk*, and *WannaCry* showed relatively lower accuracy rates, especially with real images at smaller resolutions (32×32 and 64×64 pixels). These discrepancies could be due to the inherent complexity of these particular ransomware families or limitations within the training data that may have resulted in less optimal feature learning.

The fluctuation in accuracy across different resolutions indicates the importance of image size in the training of the models. Larger image sizes might contain more detailed information that contributes to better feature learning and, thus, more accurate classification.

The results underscore the successful application of GANs, specifically the InfoGAN model, in generating synthetic ransomware images. The synthesized images were seemingly close enough to real images to be effectively utilized for model training and testing. It not only broadens the possibilities for data augmentation but also provides a safer and more efficient method for model training, as it mitigates the risks associated with the direct execution of malware.

6 Conclusions

In conclusion, this study has successfully demonstrated the potential of using the InfoGAN model and PE Header features for generating synthetic ransomware samples.

By leveraging PE Header features, the InfoGAN model could generate realistic samples that mimic the structural characteristics of real malware, thereby enhancing the quality and utility of the generated data.

Validation of the synthetic samples using a CNN model further underscored the realism and quality of the generated images. The high classification accuracy achieved by the CNN model on the synthetic samples attests to their potential as a valuable resource for training and testing malware detection models.

While the results of this study are promising, future work should focus on expanding the methodology to other types of malware and improving the generation process to produce more diverse samples. It would help further to enhance the generalizability and robustness of malware detection models.

References

1. Makridis, C.A., Mishra, S.: Artificial intelligence as a service, economic growth, and well-being. J. Serv. Res. **25**, 505–520 (2022)
2. Ansari, M.F., Dash, B., Sharma, P., Yathiraju, N.: The impact and limitations of artificial intelligence in cybersecurity: a literature review. Int. J. Adv. Res. Comput. Commun. Eng. (2022)
3. Majid, A.-A.M., Alshaibi, A.J., Kostyuchenko, E., Shelupanov, A.: A review of artificial intelligence based malware detection using deep learning. Mater. Today: Proc. **80**, 2678–2683 (2023)
4. Akhtar, Z.: Malware detection and analysis: challenges and research opportunities. arXiv preprint arXiv:2101.08429 (2021)

5. Halevy, A., Norvig, P., Pereira, F.: The Unreasonable effectiveness of data. IEEE Intell. Syst. **24**, 8–12 (2009)
6. Zhang, C., Bengio, S., Hardt, M., Recht, B., Vinyals, O.: Understanding deep learning (still) requires rethinking generalization. Commun. ACM **64**, 107–115 (2021)
7. Goodfellow, I., et al.: Generative adversarial nets. In: Advances in Neural Information Processing Systems, vol. 27 (2014)
8. Dutta, I.K., Ghosh, B., Carlson, A., Totaro, M., Bayoumi, M.: Generative adversarial networks in security: a survey. In: 2020 11th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), pp. 0399–0405. IEEE (2020)
9. Beaman, C., Barkworth, A., Akande, T.D., Hakak, S., Khan, M.K.: Ransomware: recent advances, analysis, challenges and future research directions. Comput. Secur. **111**, 102490 (2021)
10. Moti, Z., Hashemi, S., Namavar, A.: Discovering future malware variants by generating new malware samples using generative adversarial network. In: 2019 9th International Conference on Computer and Knowledge Engineering (ICCKE), pp. 319–324. IEEE (2019)
11. Ding, Y., Shao, M., Nie, C., Fu, K.: An efficient method for generating adversarial malware samples. Electronics **11**, 154 (2022)
12. Lu, Y., Li, J.: Generative adversarial network for improving deep learning based malware classification. In: 2019 Winter Simulation Conference (WSC), pp. 584–593. IEEE (2019)
13. Singh, A., Dutta, D., Saha, A.: MIGAN: malware image synthesis using GANs. In: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 10033–10034 (2019)
14. Gao, X., Hu, C., Shan, C., Han, W.: MaliCage: a packed malware family classification framework based on DNN and GAN. J. Inf. Secur. Appl. **68**, 103267 (2022)
15. Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., Abbeel, P.: InfoGAN: interpretable representation learning by information maximizing generative adversarial nets. In: Advances in Neural Information Processing Systems, vol. 29 (2016)
16. Nataraj, L., Karthikeyan, S., Jacob, G., Manjunath, B.S.: Malware images: visualization and automatic classification. In: Proceedings of the 8th International Symposium on Visualization for Cyber Security, pp. 1–7 (2011)