cryptography

Article

Automated Classical Cipher Emulation Attacks via Unified Unsupervised Generative Adversarial Networks

Seonghwan Park [1,†], Hyunil Kim [2,3,†] and Inkyu Moon [1,*]

[1] Department of Robotics and Mechatronics Engineering, Daegu Gyeongbuk Institute of Science & Technology (DGIST), Deagu 42988, Republic of Korea; tdn02007@dgist.ac.kr
[2] Department of Convergence Science, Kongju National University, Gongju 32588, Republic of Korea; hyunil89@kongju.ac.kr
[3] Basic Science Research Institution, Kongju National University, Gongju 32588, Republic of Korea
* Correspondence: inkyu.moon@dgist.ac.kr; Tel.: +82-53-785-6223
† These authors are co-first authors of this paper.

Abstract: Cryptanalysis has been studied and gradually improved with the evolution of cryptosystems over past decades. Recently, deep learning (DL) has started to be used in cryptanalysis to attack digital cryptosystems. As computing power keeps growing, deploying DL-based cryptanalysis becomes feasible in practice. However, since these studies can analyze only one cipher type for one DL model learning, it takes a lot of time to analyze multi ciphers. In this paper, we propose a unified cipher generative adversarial network (UC-GAN), which can perform ciphertext-to-plaintext translations among multiple domains (ciphers) using only a single DL model. In particular, the proposed model is based on unified unsupervised DL for the analysis of classical substitutional ciphers. Simulation results have indicated the feasibility and good performance of the proposed approach. In addition, we compared our experimental results with the findings of conditional GAN, where plaintext and ciphertext pairs in only the single domain are given as training data, and with CipherGAN, which is cipher mapping between unpaired ciphertext and plaintext in the single domain, respectively. The proposed model showed more than 97% accuracy by learning only data without prior knowledge of three substitutional ciphers. These findings could open a new possibility for simultaneously cracking various block ciphers, which has a great impact on the field of cryptography. To the best of our knowledge, this is the first study of the cryptanalysis of multiple cipher algorithms using only a single DL model

Keywords: cryptanalysis; substitution ciphers; generative adversarial networks; unsupervised deep learning

Citation: Park, S.; Kim, H.; Moon, I. Automated Classical Cipher Emulation Attacks via Unified Unsupervised Generative Adversarial Networks. *Cryptography* **2023**, *7*, 35. https://doi.org/10.3390/cryptography7030035

Academic Editor: Shay Gueron

Received: 24 May 2023
Revised: 10 July 2023
Accepted: 10 July 2023
Published: 11 July 2023

1. Introduction

Cryptanalysis refers to techniques for breaking into cryptographic security systems and gaining access to the contents of encrypted messages, even when the cryptographic key is unknown by Kirchhoff's principle, which is the paramount principle in modern cryptosystems [1,2]. It attempts to discover a meaningful pattern inside a given ciphertext to recover the corresponding plaintext or key. Such cryptanalysis techniques have been studied and gradually improved with the evolution of cryptosystems over past decades. This cryptanalysis technology can be applied to encryption for audio encryption [3] and face recognition [4] used in real life.

Recently, a new class of artificial intelligence (AI)-based cryptographic attacks on digital cryptosystems have been proposed to gain efficient, meaningful cipher cracking and classification results [5,6]. The AI algorithms based on deep neural networks have led to huge improvements in computer vision [7], medical image processing [8], machine translation [9], and the generation of virtual data [10–12] that are almost identical to the real data, and the attacks on cyber information security over the past decade [13,14]. More

recently, generative adversarial networks (GANs) [15,16] have produced great results for image generation, translation, resolution enhancement, and synthesis [12,17–20]. These advances in GAN models can even generate realistic face images from virtual people [12,21]. In addition, few studies break and emulate ciphertexts using GANs [22,23]. Reference [23] demonstrated that CipherGAN was capable of providing the underlying cipher mapping between unpaired ciphertext and plaintext for automated cryptanalysis without any prior knowledge, such as the character frequency distribution observed in natural language. This new class of AI-based cryptanalysis has been proposed to gain efficient and meaningful cipher cracking results. Especially, generative adversarial networks (GANs) have produced great results for image generation and translation. These advances in GANs models can even generate realistic data. A language model must represent both the feature distributions at sequential data point, and the possibly intricate temporal dynamics of those variables [19]. In particular, we want to properly represent the conditional distribution of temporal transitions in multivariate sequential data. The recently introduced StarGAN is an efficient method for multi-domain image-to-image translation, which takes in as input images in different domains and learns to flexibly translate the input image into the output image in the target domain, instead of learning a fixed translation, such as black-to-blond hair [24]. As a result, StarGAN can learn image-to-image translations with a single learning model that covers multiple domains.

Inspired by StarGAN, we propose new cipher emulation attacks using GAN-based unsupervised deep learning techniques for the automated analysis of classical substitutional ciphers. StarGAN seeks to discover a mapping between several domains via a single framework. Therefore, it perfectly matches our proposal of a multi-domain cipher emulation attack on classical ciphers. This method has been shown to be effective and is commonly used in computer vision applications such as creating samples of natural pictures. In this study, we present the unified GANs for ciphertext-to-plaintext translations in different domains with a single model. We demonstrate that the proposed unified cipher-generative adversarial network (UC−GAN) can understand the confusion property of classical ciphers, which is one of the principles of modern symmetric cryptography. To the best of our knowledge, this is the first study of the unified unsupervised deep learning-based cryptanalysis of multiple cipher algorithms using only a single model.

The main contributions of this study are summarized as follows. First, we propose the cipher emulation attacks based on unsupervised deep learning for the analysis of classical substitutional ciphers. Our proposed method can break classical ciphers using ciphertext-only without any prior knowledge, such as the character frequency distribution observed in natural language or the spaces between words in plaintext.

Second, we demonstrate that our proposed UC−GAN can perform ciphertext-to-plaintext translations among multiple domains (or ciphers) using only a single deep-learning model. As a result, UC−GAN can emulate both "ciphertext in different domains-to-plaintext" and "plaintext-to-ciphertext in different domains" translations.

Third, to show our model's feasibility and good performance, we compare our experimental results with the conditional GAN and CipherGAN results. These two GAN models can only perform the mapping between ciphertext and plaintext within a single domain (or cipher). Therefore, multiple models should be built separately in order to learn the mapping between ciphertext and plaintext in breaking more than two ciphers.

This paper is organized as follows. Section 2 explains and introduces the background and related works. Section 3 describes our proposed method, while Section 4 presents our experimental results. Finally, Section 5 discusses our experimental results, and Section 6 concludes this paper.

2. Background and Related Works

In this section, we first describe the basic concepts of the GAN used to design the main elements of our network model, followed by a review of classical cipher techniques.

We then introduce existing AI-based cryptanalysis for substitution ciphers. Table 1 shows some mathematical symbols used in the GAN model equations in this study.

Table 1. Mathematical symbols definition.

Symbol	Definition
$\mathbb{E}[x]$	Expectation
$\mathrm{E}(x)$	Embedding
$\mathrm{E}(x)\|\|\mathrm{c}$	Concatenated embedding and target
$\|x\|_1$	L1 norm (mean absolute error)
G	Generator
D	Discriminator
c	Target domain label
c'	Original domain label
∇	Gradient

2.1. Generative Adversarial Networks

Generative adversarial networks (GANs) have been successfully developed into various real-world applications, including image synthesis, transfer, super-resolution, and classification. The GAN algorithms simultaneously train a pair of neural networks (generator and discriminator) in competition with each other. Figure 1 illustrates the basic idea of the GAN. There are two models in GAN architecture that might be able to generate fake data that look like the real thing. When dealing with images, the generator model needs to generate images. The second model is the discriminator network, which attempts to distinguish between fake and real data. Both models compete with each other. The generator model attempts to fool the discriminator network; at that point, the discriminator model adapts to the new fake data. This information can be used to improve the generator model. Consequently, the trained generator can create fake data that are indistinguishable from real data. The discriminator and generator models are set to compete against each other as adversaries, each attempting to surpass the other, and in the process, both becoming better and better. To sum up, the two models are playing a minimax game with the following objective function [21,22]:

$$\min_G \max_D V(D,G) = \mathbb{E}[\log D(x)] + \mathbb{E}[\log(1 - D(G(z)))] \quad (1)$$

where, G is the generator network that accepts sample data, z from a noise distribution $p_z(z)$ and produces the data in a target domain, D is the discriminator network that accepts the sample x from the training dataset or the output from the generative model and predicts the probability that x or $G(z)$ came from the training dataset. When the model is trained by computing Equation (1), the generator will ultimately be able to create more sophisticated fake images. Figure 1 can be utilized as a general guideline for training GAN. Figure 1 shows the general architecture of the GAN model.

In recent years, there has been tremendous progress in the development of the GAN model to generate more exquisite samples and obtain better training stability than the original GAN model [25]. The original GAN cannot generate conditional data, meaning there is no control over the modes of generating data. The conditional GAN model was first proposed to acquire a conditional generative model by setting up the target label as an additional input for the generator and the discriminator [26]. In addition, the Pix2Pix model [27] was proposed for the first time to conduct the image-to-image translation tasks in a single direction. However, the Pix2Pix model is a supervised learning model that is characterized by training on pairs of datasets. CycleGAN [28] is the first unsupervised image-to-image translation model using unpaired datasets, which translates between two

domains in both directions [29] (see Figure 2). More recently, the unified generative model was presented for unsupervised learning to translate images between multiple domains with only one generator and discriminator, using domain classification loss with target domain labels [30,31]. In this paper, we propose a new cipher emulation attack algorithm based on the unified GANs to perform ciphertext in different domains (ciphers)-to-plaintext translations using only a single deep learning model.

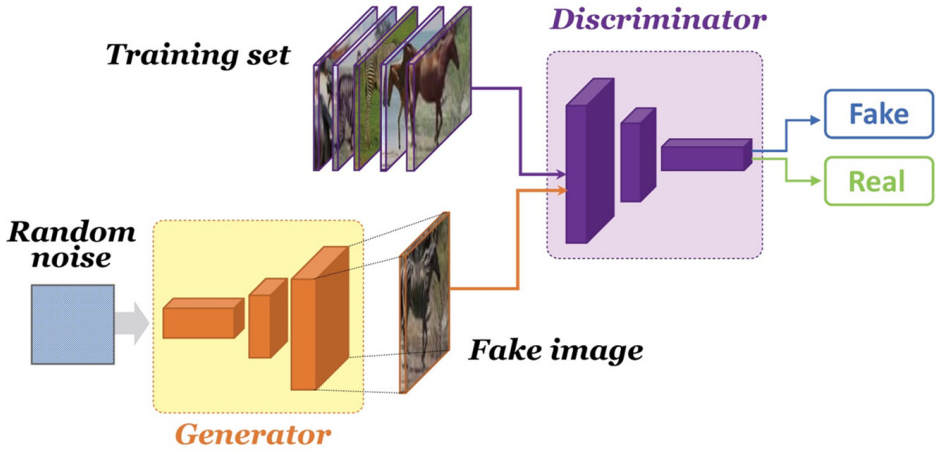

Figure 1. The general architecture of the GAN model.

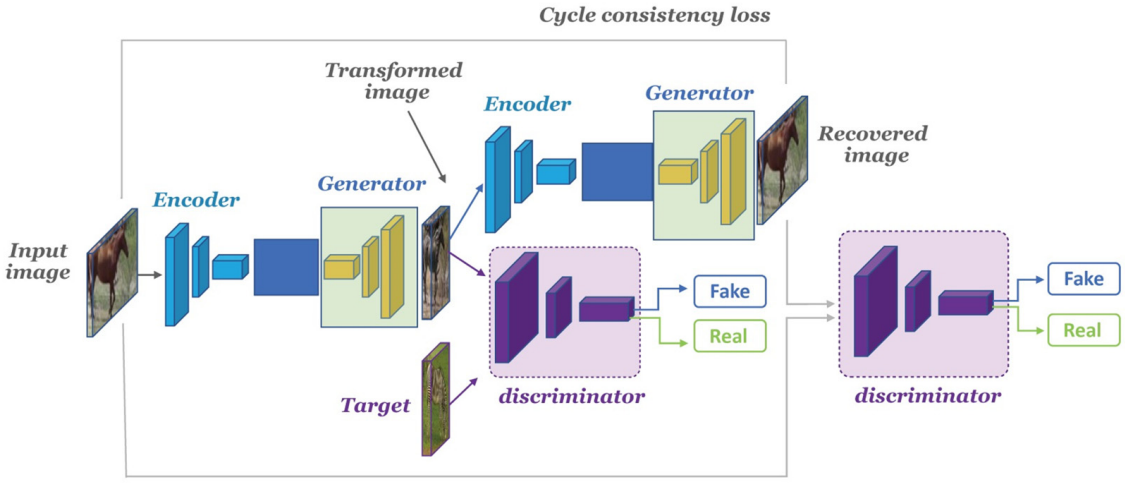

Figure 2. Schematic representation of the CylcleGAN.

2.2. Classical Substitution Ciphers

We briefly explain the classical substitution ciphers used in this study which include shift cipher, Vigenere cipher, and substitution cipher [32]. A shift cipher encryption method encrypts the original plaintext into unreadable ciphertext, where each letter in the original message is replaced with a letter corresponding to a certain number of letters up or down in the alphabet. The number of possible shifts is limited to between 0 and 25 in the English language, which is equal to the number of English letters. The receiver decrypts the ciphertext by returning each letter in the encrypted data. A Vigenere cipher is categorized as a poly-alphabetic cipher that encrypts a plaintext into a set of different letters using the key with the total number of possible keys of 26^m, i.e., all the possible sequences of letters of length m. The substitution cipher deploys any permutation of the 26 letters as a key. Therefore, the total number of possible keys are $26! \approx 2^{88.4}$ (! is factorial, $\approx$ is approximation).

2.3. AI-Based Cryptanalysis

Substitution ciphers have a confusion property that provides obscurity between the encryption key and the corresponding ciphertext. In particular, confusion is the basic building block of modern symmetric cryptography with the diffusion property. Therefore, understanding the structure and the vulnerability of substitution ciphers can be the basis of the cryptanalysis of modern cryptography. Table 2 shows the various AI-based cryptanalysis research. Reference [23] trains language data encrypted using shift and Vigenere ciphers in a stable way using the CycleGAN model. The studies of AI-based cryptanalysis for modern block ciphers were also introduced [33–35]. A recent work by Baek et al. [33] classifies AI-based attacks into three methods: key recovery attack, cipher identification, and cipher emulation attack. Regarding attacks on modern lightweight ciphers, such as Speck32/64, the authors introduced a deep learning-based ciphertext distinguisher that classifies between the ciphertexts obtained by the Speck32/64 algorithm with 11 rounds and random values [34]. In addition, using the neural distinguisher, the authors performed a partial key recovery attack. Furthermore, recent work by Baksi et al. [35] proposed an improved neural distinguisher, especially some lightweight ciphers such as Gimli, Ascon, Knot, and Chaskey. However, some approaches using neural distinguishers by references [34,35] are specific for some ciphers. In addition, there is a small improvement in accuracy results rather than the original differential distinguisher. This means that AI-based approaches will have to try various challenges against confusion and diffusion property, to break S-box using state-of-the-art deep learning methods like GANs. Such improvements will be a basic building block for AI-based cryptanalysis.

Table 2. AI-based cryptanalysis research.

Method	Objectives	Data	Basic Model
Gomez et al. [23]	Cipher cracking	Shift and Vigenere ciphers	CycleGAN
Baek et al. [33]	AI-based attacks review	Block ciphers	Dense, CNN
Gohr et al. [34]	Ciphertext distinguisher	Lightweight ciphers (Speck32/64)	Resnet
Baksi et al. [35]	Ciphertext distinguisher	Lightweight ciphers (Gimli, Ascon, Knot, and Chaskey)	MLP, CNN, LSTM
Sirichotedumrong et al. [10]	Image transformation scheme	CIFAR-10, CIFAR-100	GAN
Ding et al. [36]	Private key generation	Medical images (Stream cipher)	GAN
Panwar et al. [37]	End-to-end image encryption survey	-	GAN, Diffusion, CNN

Also, some approaches studied advanced cryptanalytic approaches in image domains. A new GAN-based image transformation technique for privacy-preserving DNN that can apply images without visual information to DNN and increase the robustness for ciphertext-only attack has been proposed [10]. A novel deep learning-based key generation network (DeepKeyGen) is proposed as a stream cipher generator for generating private keys that can be used for encryption and decryption of medical images [36]. K. Panwara et al. [37] summarized the latest trends in deep learning-based end-to-end encryption techniques. Existing deep learning-based encryption systems are classified into three categories: encryption with style transfer, style transfer with enhanced diffusion properties, and combining deep neural networks with chaotic systems, and each methodology is discussed and related conclusions are made. However, their approaches focus only on the image domain with image encryption. On the other hand, we have focused on the natural language domain with a substitutional cipher on symmetric cryptosystems.

3. Overview of the Proposed UC−GAN Cryptanalysis Model

We propose a UC−GAN network model for multi-domain cryptanalysis that is based on StarGAN using a single unified generator and discriminator model. In our model, as in CipherGAN, our objective is to train the generator on multiple cipher and plain domains without any prior knowledge. However, the two models differ in the number of generators. Figure 3 shows the process of UC-GAN and CipherGAN generating plaintext from the ciphertext. As the CipherGAN model is based on the CycleGAN model, each generator must be used for each ciphertext domain. However, the proposed UC-GAN model uses only a single unified generator for multiple ciphertext domains based on the StarGAN model.

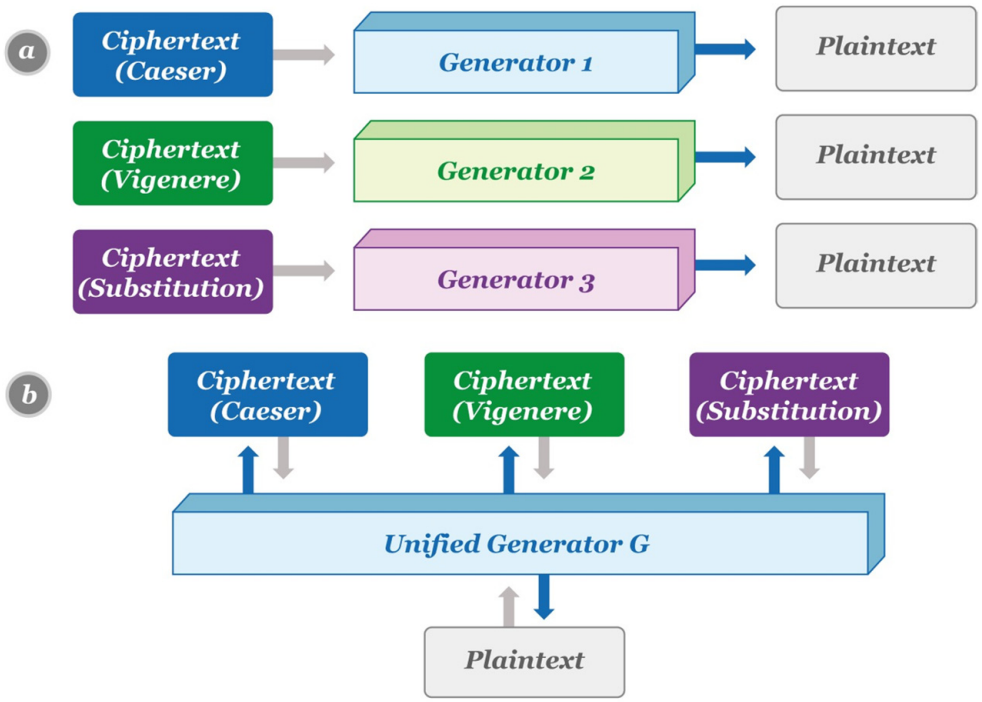

Figure 3. The process of generating the plaintext from the ciphertext with (**a**) the CipherGAN model; (**b**) the proposed UC-GAN model.

We therefore present an enhanced form of StarGAN for multi-domain cryptanalysis. We specifically consider training objectives and network designs regarding conditional techniques. In addition, we use embedding space for continuous relaxations [23,38] before main training. Furthermore, we suggest an adversarial loss that supports the conversion of all source domain data into the target single data (multi-cipher to single-plain) or (single-plain to multi-cipher) with only one training process. Similar to CipherGAN, the goal of our approach is to train the generator G on various cipher and plain domains with no previous information. The fundamental distinction is that we employ a single unified generator (G) and discriminator (D) across several cipher domains. The primary distinction between our model and CipherGAN is that the CipherGAN requires a ratio of the number of G and D to the number of ciphertext/plaintext domains. Therefore, our method is greatly improved in representing multi-domain on discrete random variables. The experiment results show that our proposed model can break and emulate multiple substitution ciphers (Caesar, Vigenere, and substitution cipher) in only one training process.

Figure 4 shows the process of creating data used for model training. To build UC−GAN, we use embedding E for continuous relaxations before the main process as embedding space W_{emb} and timing space W_{time}. The embedding E consists of a one-hot vector step and a simplex to embedding step. In the one-hot vector step, each character is represented by a one-hot vector with the length of v ($v = 26$ in our setting). To make

embedding vector $E(x)$, it computes $r = t \cdot W_{emb}$ and $E(x) = \mathrm{r}||W_{time}$, where the original one-hot vector t from the original data x, with embedding space W_{emb} and additional timing space W_{time}. Such continuous relaxation makes it possible to back-propagate when we are training neural network models with preserved information. In our setting, we assume that the original data x consists of the number of N characters. Therefore, $t \in \{0,1\}^{N \times v}$ and $W_{Emb} \in \mathbb{R}^{v \times E}$ and $W_{time} \in \mathbb{R}^{N \times T}$ are trained parameters which we have to update when the training is processed. Finally, the input data of the model are created by concatenating the embedding data and the target domain label.

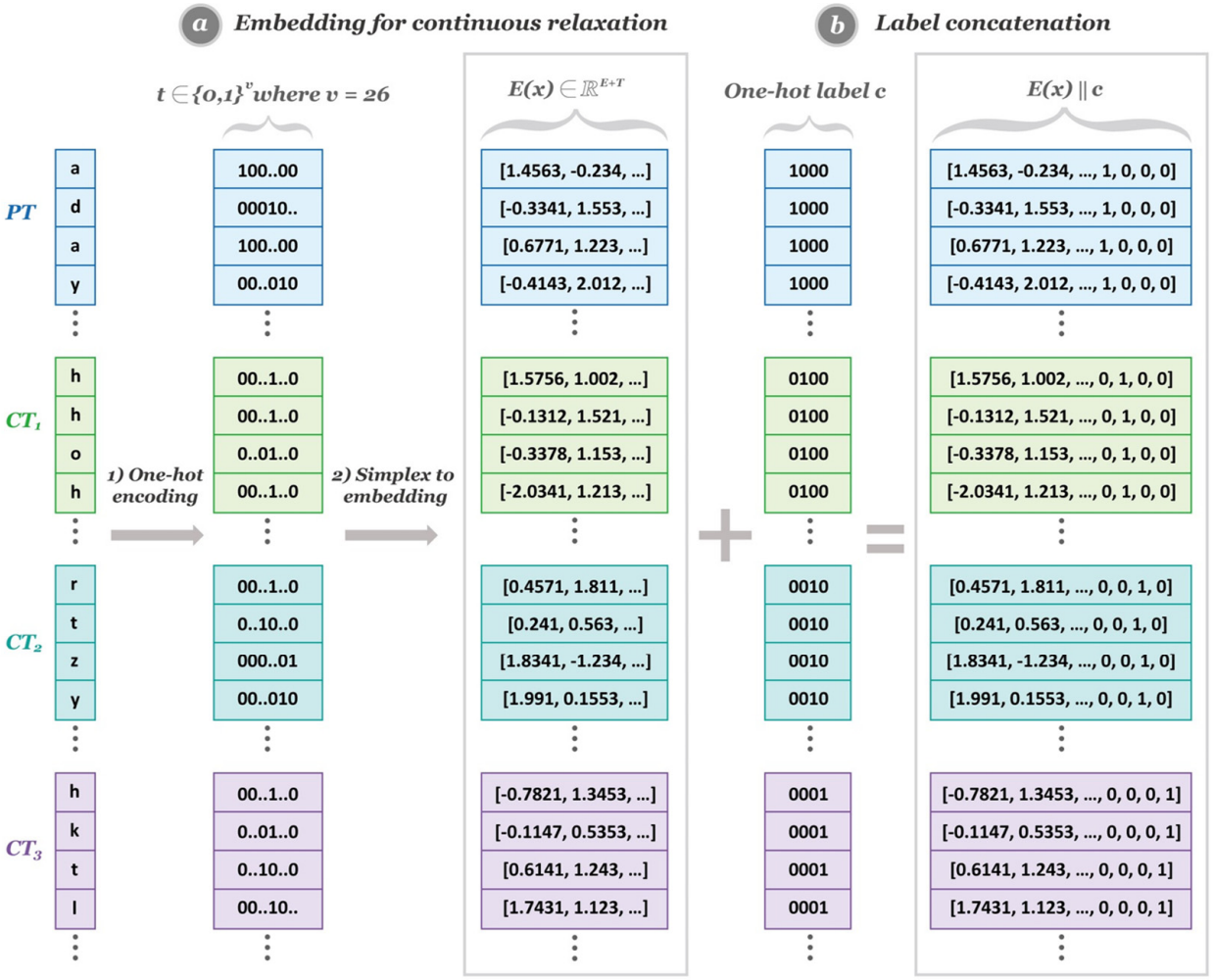

Figure 4. The process of creating data used for model training. (**a**) Embedding for continuous relaxation; (**b**) label concatenation. Finally, concatenated $E(x) \parallel c$ is an input for the unified generator G.

Figure 5 shows the training process of UC-GAN when using ciphertext as an input. The generator of UC-GAN can generate plaintext from ciphertext. In this case, inputs of generator are embedded ciphertext and target domain label indicating plaintext. In our proposed model, we have four domains PT, CT_1, CT_2, and CT_3. These domain labels are set as the plaintext domain PT to 0, Caesar ciphertext domain CT_1 to 1, Vigenere CT_2 to 2, and substitution cipher CT_3 to 3. The discriminator produces two probability distributions that distinguish whether the source used as the input of the discriminator is real or generated by the generator (D_{src}), and checks whether the domain of the source is the same as the target domain label c (D_{cls}).

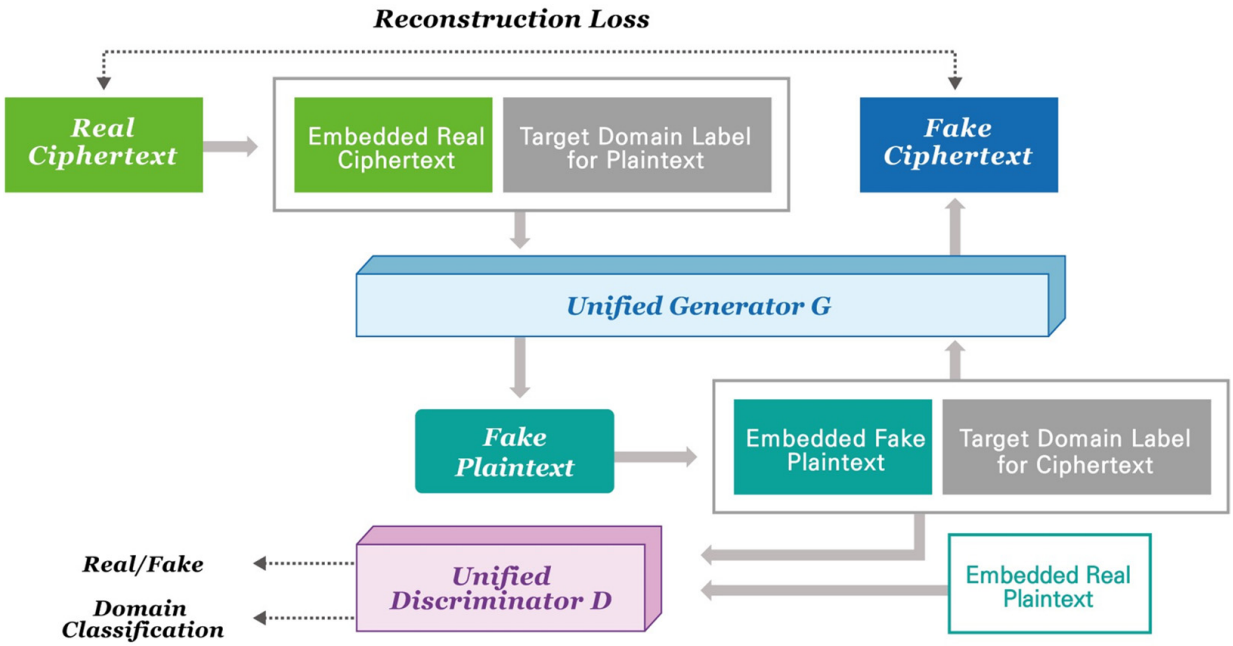

Figure 5. The process of training the UC-GAN using ciphertext as an input. The generator generates fake plaintext by embedding the real ciphertext and using it as an input to the generator along with the target domain label. After embedding the generated fake plaintext, the fake ciphertext can be generated by using it as an input to a generator, and results for domain classification and real or fake classification can be obtained by using it as an input to a discriminator. It is trained the same way when using plaintext as an input.

The loss for generator G consists of three losses: adversarial loss, domain classification loss, and reconstruction loss. Adversarial loss for the generator is a value for learning the generator using the results of the discriminator. Afterward, the model learns in the direction in which the loss is reduced so that the output generated from the generator is classified as real by the discriminator. We use adversarial loss with the least squared loss [39]. The adversarial loss for generator is described as follows:

$$L_{adv\_g} = \mathbb{E}_{x,c}[(D_{src}(E(G(E(x), c))) - 1)^2] \tag{2}$$

where, x is the input of the generator, which can be either plaintext or ciphertext. c is the target domain label, and the generator G transforms x into the domain corresponding to c. E is an embedding function and D is a discriminator.

The domain classification loss for generator is a value that the discriminator determines whether the output generated by the generator is converted to the desired target domain label. The domain classification loss for generator is described as follows:

$$L_{cls}^{f} = \mathbb{E}_{x,c}[-\log D_{cls}(c|E(G(E(x), c)))] \tag{3}$$

Reconstruction loss is a cycle-consistency loss used in cycleGAN, which is a value calculated by calculating the difference with the input when returning to the domain of the original input after passing through the generator twice. First, the original image is transformed into an image of the target domain, and then the original image is reconstructed from the transformed image. We used the L1 norm for reconstruction loss. The domain classification loss for generator is described as follows:

$$L_{rec} = \mathbb{E}_{x,c,c'}[||x - G(E(G(E(x), c)), c')||_1] \tag{4}$$

where, G takes in the translated image $G(E(x), c)$ and the original domain label c' as the input and tries to reconstruct the original image x. Therefore, the loss for the generator is described as the summation of Equations (2)–(4):

$$L_g = L_{adv} + \lambda_{cls} L_{cls}^f + \lambda_{rec} L_{rec} \tag{5}$$

where, λ_{cls} and λ_{rec} are hyper-parameters for L_{cls}^f and L_{rec}. The generator is trained to minimize Equation (5).

The loss for discriminator D consists of three things. These are adversarial loss, domain classification loss, and Wasserstein gradient penalty loss. The adversarial loss for the discriminator is a value that the discriminator learns to determine the actual ciphertext or the actual plaintext for real and to determine the generated sentence for fake. The adversarial loss for discriminator is described as follows:

$$L_{adv} = \mathbb{E}_x[(D_{src}(E(x)) - 1)^2] + \mathbb{E}_{x,c}[(D_{src}(G(E(x), c)))^2] \tag{6}$$

The domain classification loss for the discriminator is the value by which the discriminator determines whether the source used as the input to the discriminator matches the target domain label. The domain classification loss for discriminator is described as follows:

$$L_{cls}^r = \mathbb{E}_{x,c'}[-\log D_{cls}(c'|E(x))] \tag{7}$$

where, D_{cls} takes in the real data $E(x)$ and the original domain label c' as the input.

Actually, the loss architecture of vanilla GAN [40] is based on Kullback–Leibler divergence (KL divergence), which is a type of statistical distance. However, gradient vanishing problem occurs sometimes, therefore it makes worse results. To prevent such issues, the Wasserstein gradient penalty loss [41] has been proposed. They used 1-Wasserstein distance (Earth Movers's distance) with constrained 1-Lipschitz function instead of KL divergence. Additionally, improved versions have been proposed for enforcing the Lipschitz constraint, by directly constraining the gradient norm of the discriminator's output with respect to its input [42]. We used this gradient penalty for a stable training process on continuous relaxed random variables which is described as follow:

$$L_{gp} = \mathbb{E}_{\hat{x}}[(||\nabla_{\hat{x}} D_{src}(E(\hat{x}))||_2 - 1)^2] \tag{8}$$

where, $\hat{x}$ is a random samples $\hat{x} \sim \mathbb{P}_{\hat{x}}$, where $\mathbb{P}_{\hat{x}}$ is a sampling uniformly along straight lines between pairs of points sampled from the data distribution and the generator distribution. Therefore, the loss for the discriminator is described as the summation of Equations (6)–(8):

$$L_D = L_{adv} + \lambda_{cls} L_{cls}^r + \lambda_{gp} L_{gp} \tag{9}$$

where, λ_{cls} and λ_{gp} is hyper-parameters for L_{cls}^r and L_{gp}. The discriminator is trained to minimize Equation (9).

Network Architecture

Figure 6 shows the overall structure of our network components. The network comprises 4 convolution blocks (1D Convolution-Layer Norm-ReLU) for channel expansion, a bottleneck layer consisting of 5 residual layers, and 2 convolution blocks (1D Convolution-Layer Norm-ReLU) to make the channel to 26. The last layer of the generator outputs the form of a vector with 26 components, and only one component is highlighted by Softmax, and the corresponding position means the position of the alphabet. Three convolution layers make up the discriminator network, which extracts the characteristics of the recovered plaintexts. However, there are significant distinctions between our network and StarGAN. First, we train language domains using a 1-dimensional (1D) convolution layer. Second, there are no deconvolutional operations in our generator network's upsampling layers. Third, we decreased the depth of the generator and the discriminator to obtain

a more reliable training outcome. We employ layer normalization for the generator and discriminator, respectively, and rectified linear units (ReLU) and leaky rectified linear units (LeakyReLU) as activation functions. Additionally, the hyperparameters are adjusted for steady training.

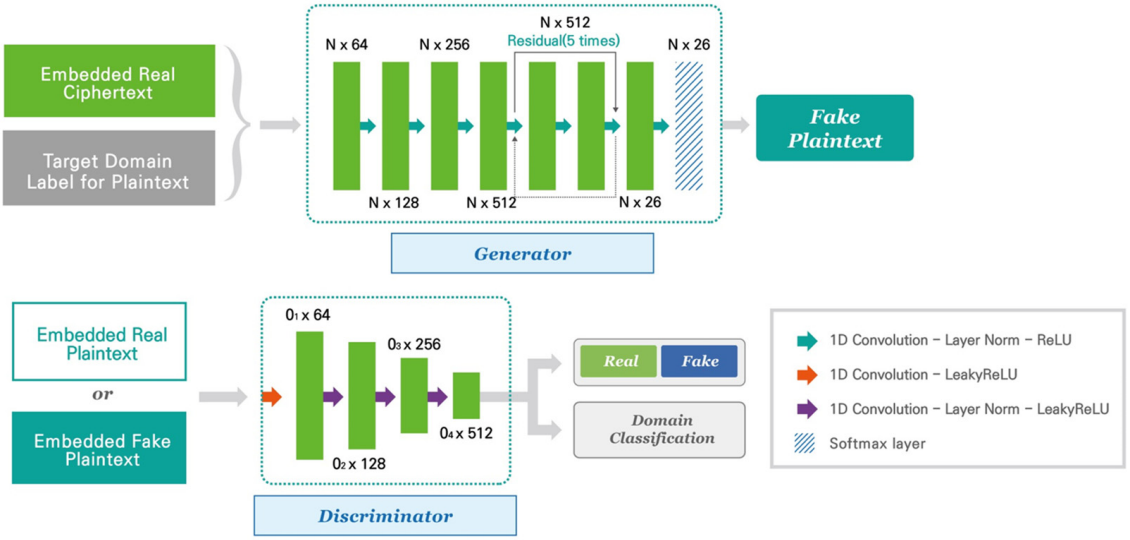

Figure 6. Schematic of the overall network components of the proposed UC−GAN. All layers consist of 1-dimensional (1D) convolution to train language domains. In addition, N and O_i are the size of output texts from the previous layer. The figure above shows the process in which ciphertext is used as an input, the generator generates plaintext, and the discriminator evaluates the generated plaintext. It is trained the same way when using plaintext as an input.

4. Experimental Results

We now assess how effectively our GAN-based method recovers substitution cipher. However, our adversarial approach may theoretically be used for many kinds of data. We also offer experimental support for this assertion in the first experiment. In the second set of experiments, we compare the Pix2Pix network model against the CipherGAN model for plain-to-cipher and cipher-to-plain recovery. In the third set of experiments, we compare the results of our UC−GAN model with the CycleGAN model.

4.1. Datasets

We employ the Brown corpus, a collection of digitized text samples in American English, for our experiments. There are four domains in the training dataset and the test dataset plaintext (PT), Caesar cipher (CT_1), Vigenere cipher (CT_2), and substitution cipher (CT_3). The dataset is encrypted with Caesar, Vigenere, and Substitution ciphers. Table 3 briefly explains how we may obtain these ciphertexts. We delete all special characters and spacings. Furthermore, each data row in our dataset consists of $N = 100$ characters. As a result, we can extract 4,537,600 characters. For the training dataset, each domain has a number of 9600 data rows to consider both the known-plaintext attack (KPA) scenario and the ciphertext-only attack (COA) scenario. In KPA settings, the attacker can access the encryption method, which means the attacker has the number of n pairs. Otherwise, in COA settings, the attacker has only ciphertexts, which means they have the number of n ciphertexts. We extract 9600 data rows for each 4 domains to show accurate unsupervised learning results. Therefore, the training dataset has $9600 \times 4 \times \mathrm{N}(100) = 3{,}840{,}000$ characters. In addition, we set the data row to 3200 (320,000 characters) in the test dataset. To show the exact effectiveness of our model, the remaining 377,600 characters are abandoned. Table 3 shows examples of the plaintext and corresponding ciphertext encrypted with different substitution ciphers:

Table 3. Examples of plaintext and ciphertext encrypted with different classical cipher methods.

Plaintext	Encrypted Sentence	Encryption Method/Key
eelementaryschoolodequindrewhich hasbeenattendedthisyearbyfourofth ekowalskichildreninclоudingchristine	*hhohphqwdubvfkrrorghtxlqguhzklfkkdvehhqd wwhqghgwklvbhduebirxuriwk hnrzdovnlfkloguhqlqforxglqjfkulvwlqh*	Caesar/3 shift to the right
	hiqkpiszdvdyfltuosiktyntgvjckmh nkexhhisgwxjtgiizkmxehewhbjtauskzkipu zeqynmhnlpixhrntfptagmsmflwoovxnth	Vigenere/defg
	ttstdtfzqknleiggsgrtjxofrktvioeiiqlwttfqzztfr trziolntqkwnygxkgyzitagv qslaoeiosrktfofesgxrofueikolzoft	Substitution/ qwertyuiopasdfghjkzxcvbnm

4.2. System Equipment

Table 4 lists the system configuration for training the proposed network models.

Table 4. System configuration for training.

Component	Description
CPU	Intel Cori7-7700
GPU	GTX 1080Ti
Language	Python
Memory	16 GB
System type	64-bit operating system
OS type	Window 10/64

4.3. Default Hyperparameter

Using the Adam optimizer, our model was trained with a batch size of 32, and a learning rate of 1.8×10^{-4}. The learning rate is exponentially warmed up over 5000 steps and then remains constant. Each epoch equals 1200 rounds. To obtain more precise results for text recovery, we update the generator once after the discriminator twice. The domain classification loss λ_{cls}, reconstruction loss λ_{rec} and gradient penalty λ_{gp} hyperparameters are set to 1, 10, and 10, respectively. For each of W_{emb} and W_{time}, we set the embedding space E and the concatenation space T to 256. The proposed method was implemented using Python for encryption and Pytorch for our experiment UC-GAN, and we train on a single NVIDIA Geforce GTX 1080Ti GPU. The GPU in general provides speedups that are at least 5 to 10 times greater than the Central Processing Unit (CPU).

4.4. Cipher Emulation Results

As described above, we demonstrate that our UC-GAN can successfully break the Caesar, Vigenère, and Substitution ciphers using only one unified generator. In addition, our unified method can learn these discrete distributions for all multi-ciphers-to-plain domains (See Figure 7a). Our encryption emulation can reconstruct all types of ciphertexts from a single plaintext, such as $PT \rightarrow CT_1$, $PT \rightarrow CT_2$, and $PT \rightarrow CT_3$ for the plain-to-multi-ciphers domain (See Figure 7b). In addition, the proposed model can emulate all types of ciphertext to a single plaintext for multi-cipher-to-plain. We measured the accuracy of the model using test data per each epoch. To test the model, the test data was used as the input of the model, and one generator generates three ciphertexts according to the labeled target. The model accuracy was calculated by comparing the generated ciphertext with the target ciphertext. Figure 7c,d show the accuracy of the test data per epoch and the highest accuracy in each cipher. As shown in Figure 7, the model reached nearly 100% accuracy for all types of ciphers in 115 training epochs. However, the convergence speed was different for different ciphers and Caesar

and Substitution were broken faster than Vigenère. Table 5 shows the results of the ciphertext generated by the model from the plaintext and the target ciphertext created by the three cipher methods. Texts that do not match between the target ciphertext and the generated ciphertext are marked in red, showing that most of the texts match. Table 6 demonstrates the actual network result for the cipher-to-plain recovery experiment.

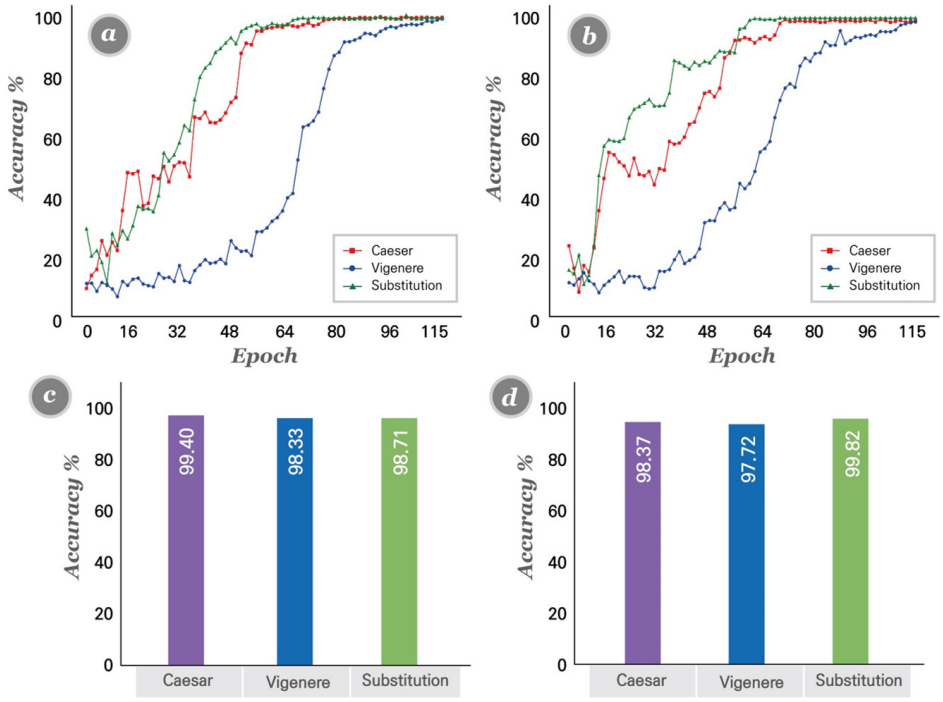

Figure 7. The proposed method tests the process and accuracy results. (**a**) proposed method test curves for multi-cipher-to-single plain recovery; (**b**) proposed method test curves for single-plain-to-multi-cipher recovery; (**c**) multi-cipher-to-single-plain recovery test results; (**d**) single-plain-to-multi-cipher recovery test results.

Table 5. Generated ciphertext and target ciphertext for plain-to-cipher emulation experiment.

Encryption Method	Original Plaintext	Target Ciphertext	Generated Ciphertext
Caeser	*medicalpiratesannuallyyouwillcomeu pwithafrighteningtotalthatswhythef datheamericanmedicalassociationa*	*phglfdosludwhvdqqxdoobbrxzloofrphx szlwkdiuljkwhqlqjwrwdowkdwvzkbw khigdwkhdphulfdqphglfdodvvrfldwlrqd*	*phglfdosludwhvdqqxdooxxrxzloofrphxs zlwkdiuljkwhqlqjwrwdowkdwvzkxwkh igdwkhdphulfdqphglfdodvvrfldwlrqd*
Vigenere	*medicalpiratesannuallyyouwillco meupwithafrighteningtotalthatswhyt hefdatheamericanmedicalassociationa*	*piiofeqvlvfzhwftqyfrocduxanrogtshyu clxmgivnmkxjtlrlzrxfrwlfzvamewljl geynherkumhgqqjjlgfrdwxufmfzlssg*	*piiofeqvlvfzhwftqyfrosduxanrogtshyucl xmgivnmkxjtlrlzrxfrwlfzvamew ljlgeynherkumhgqqjjlgfrdwxufmfzlssg*
Substitution	*medicalpiratesannuallyyouwillcom eupwithafrighteningtotalthatswhyt hefdatheamericanmedicalassociationa*	*dtroeqshokqztlqffxqssnngxvosse gdtxhvoziqykouiztfofuzgzqsziqzlvinz ityrqzitqdtkoeqfdtroeqsqllgeoqzogfq*	*dtroeqshokqztlqffxqssnngxvosseg dtxhvoziqykouiztfofuzgzqsziqzlvinz ityrqzitqdtkoeqfdtroeqsqllgeoqzogfq*

To ensure the accuracy of our proposed model, we evaluated the accuracy for each character and presented a confusion matrix (See Figure 8). Considering that the character frequencies in the data are imbalanced, we constructed a confusion matrix with the frequency ratio of each character. We observed that high-frequency characters performed well while low-frequency characters showed relatively low performance. The characters b, j, k, q, x, and z exhibit comparatively poor performance, and their infrequent usage can be verified by analyzing the character frequency. We calculated a weighted F1 score that considers the imbalanced character frequencies. We multiplied the F1 score calculated for each character by its frequency ratio and then calculated the total calculated values.

Looking at the calculated F1 score, with a value very close to 1, the proposed model shows that it is possible to break three ciphers simultaneously. To exactly validate our generated ciphertext results, we have made target ciphertext by CrypTool [43], which is officially released for non-commercial use and provides different types of cryptographic algorithms including classical cryptosystems and block ciphers.

Table 6. Generated plaintext and target plaintext for cipher-to-plain emulation experiment.

Encryption Method	Original Ciphertext	Target Plaintext	Generated Plaintext
Caeser	*gdxjkwhuplvvvxvdqdqqylhwkwrpufrq udgzdoovrqrigufrqudgzdoodqgpuvqhoo nhqqhgbzdoowkhpduuldjhzlooehtxllhwo*	*daughtermisssusanannviethtomrco nradwallsonofdrconradwallandmrsn ellkennedywallthemarriagewillbequietl*	*daughtermisssusanannviethtomrcon radwallsonofdrconradwallandmrsn ellkennedywallthemarriagewillyequietl*
Vigenere	*rtzykesjsvtjkmroqxtzkiukujjiwmttwljbh xjxdrrgqelkuwfcdwfzkvnromsms sxylfnrlxdzkitrgqftzexgoqtywxtussxy*	*opushandprodhimintotheperfectionth eveteranmanagersawasathrillingposs ibilitytheoldmanwasalmosttooposs*	*opushandprodhimintotheperfectio nthepeteranmanagersawasathrillingpo ssiyilitytheoldmanwasalmosttooposs*
Substitution	*hktltfztrzgzitzgvfeqxfeossqlzfou izqlviqzolightrvosswtzityoklzlzthofg wzqofofuqigdtkxsteiqkztkygkzitzg*	*presentedtothetowncauncillast nightaswhatishopedwillbethefirstste pinobtainingahomerulecharterfortheto*	*Presentedtothetowncouncillastnightaswh atishopedwillpethefirststepinoptaininga homerulecharterfortheto*

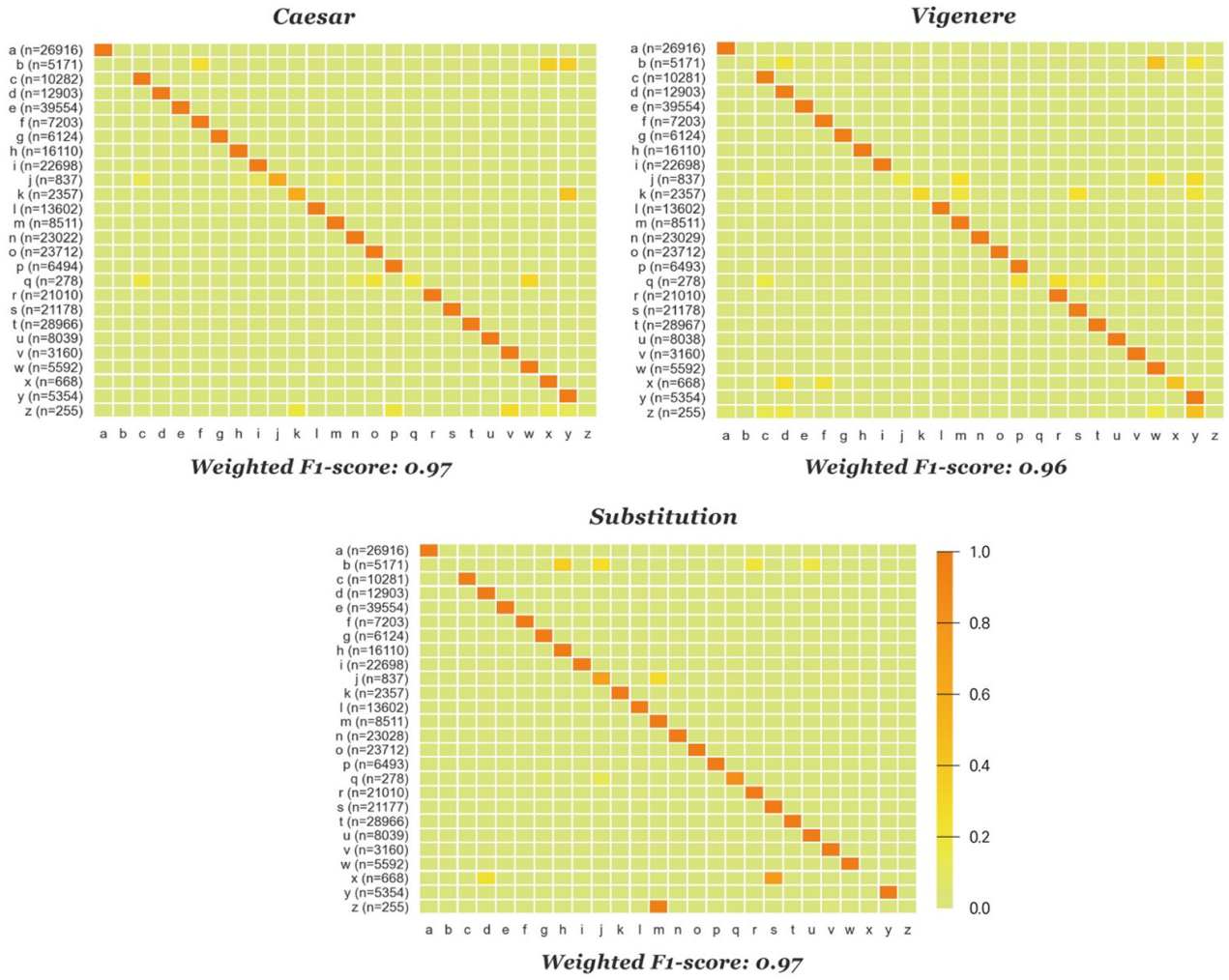

Figure 8. The confusion matrix results with the frequency ratio of each character for multi-cipher-to-single plain recovery. The *x*-axis is predicted characters and *y*-axis is true characters with the frequency of characters. Weighted F1-scores are displayed under the confusion matrix.

4.5. Computational Complexity and Memory Usage

Deep learning models require a significant amount of computation and memory to train and evaluate. Two important metrics used to quantify the computational complexity and memory usage of deep learning models are multiply-and-accumulate operations (MACs) and memory usage. MACs are a measure of the number of arithmetic operations required to perform forward and backward passes through a neural network model. Therefore, the higher the amount of computation, the higher the index of MAC. Memory usage refers to the amount of RAM or GPU memory occupied by the model and its associated data structures during training or inference. Memory usage varies depending on the size of the data being used. We calculated the MACs and memory usage of the generator's process that converts from ciphertext to plaintext when the batch size was 1. A batch size of 1 means that we used 1 row consisting of $N = 100$ characters as the input. The result is MACs of 0.28 B (billion) and memory usage of 21.54 MB. For Discriminator, MACs of 0.04 B and memory usage of 13.92 MB were shown. Since the proposed model has a small network structure based on 1D convolution, it shows lower complexity than Resnet50 (4.13 B and 206.57 MB), which is mainly used for image-based research. As it has low complexity, it shows that it is possible to train quickly even in a relatively low-cost computation environment.

4.6. Model Result with Various Hyperparameters

The task of selecting a collection of ideal hyperparameters for a learning algorithm is known as hyperparameter optimization or tuning in machine learning. To generalize diverse data patterns, the same machine learning model may require different constraints, weights, or learning rates that highly impacts the training processes. The effect of tweaking hyperparameters on network training was examined utilizing several different experiments. The highly important parameters including batch size and embedding space W_{emb} and W_{time} were investigated. The batch size refers to the quantity of data utilized to train the network in each epoch. A big batch size hinders network convergence, whereas a small batch size causes the network to fluctuate without attaining adequate performance. To obtain the optimized parameters we conducted several experiments. Table 7 demonstrates different hyperparameters settings used to run different experiments. Especially, we empirically found that the embedding space W_{emb}, W_{time} and batchsize bs are crucial factors in achieving stable training results in our scenario. However, the values for lambda in the classification loss λ_{cls} and the reconstruction loss λ_{rec} are highly sensitive to change making it challenging to train. Therefore, following StarGAN [24], we fixed λ_{cls} to 1 and λ_{rec} to 10 during our experiments. Similarly, changing the learning rate lr can result in unstable training times. After many times experimenting, we found that 18×10^{-4} is the best learning rate for our model. We then tested various values for bs, W_{emb}, and W_{time} with a fixed value for λ_{cls}, λ_{rec}, and lr. We changed the batch size by a factor of 4 and the embedding space by a factor of 2, to ensure that these hyperparameters significantly impacted the model.

In the first round of the experiment, we investigated the impact of batch size on the different cipher attacks. First, we set the batch size to 8 and ran the training process for both ciphers to plain and plain to cipher attacks. Figure 9a and b demonstrate the network training process on a cipher to plain and plain to cipher, respectively. As can be seen from Figure 9, the network could not converge for all types of ciphers in 120 epochs. The converge speed is lower than the batch size of 32 meaning that the lower batch size needs more training epochs to converge all ciphers. Afterwards, we set the batch size to 128 and ran the training process. Figure 9c and d demonstrate the network training process on a cipher to plain and plain to cipher, respectively, for a batch size of 128, respectively. Similarly, the network could not converge for all types of ciphers. Figure 9 demonstrates the test accuracy result comparison for batch size 8 (Figure 9e) and batch size 128 (Figure 9f).

Table 7. Hyperparmeter tuning experiments.

Parameter	Default Setting	Experiment 1	Experiment 2	Experiment 3	Experiment 4
Learning rate (*lr*)	1.8×10^{-4}	1.8×10^{-4}	1.8×10^{-4}	1.8×10^{-4}	1.8×10^{-4}
Batch size (*bs*)	32	8	128	32	32
Embedding space W_{emb} and W_{time}	256	256	256	128	512
Lambda for classification loss function (λ_{cls})	1	1	1	1	1
Lambda for reconstruction function (λ_{rec})	10	10	10	10	10

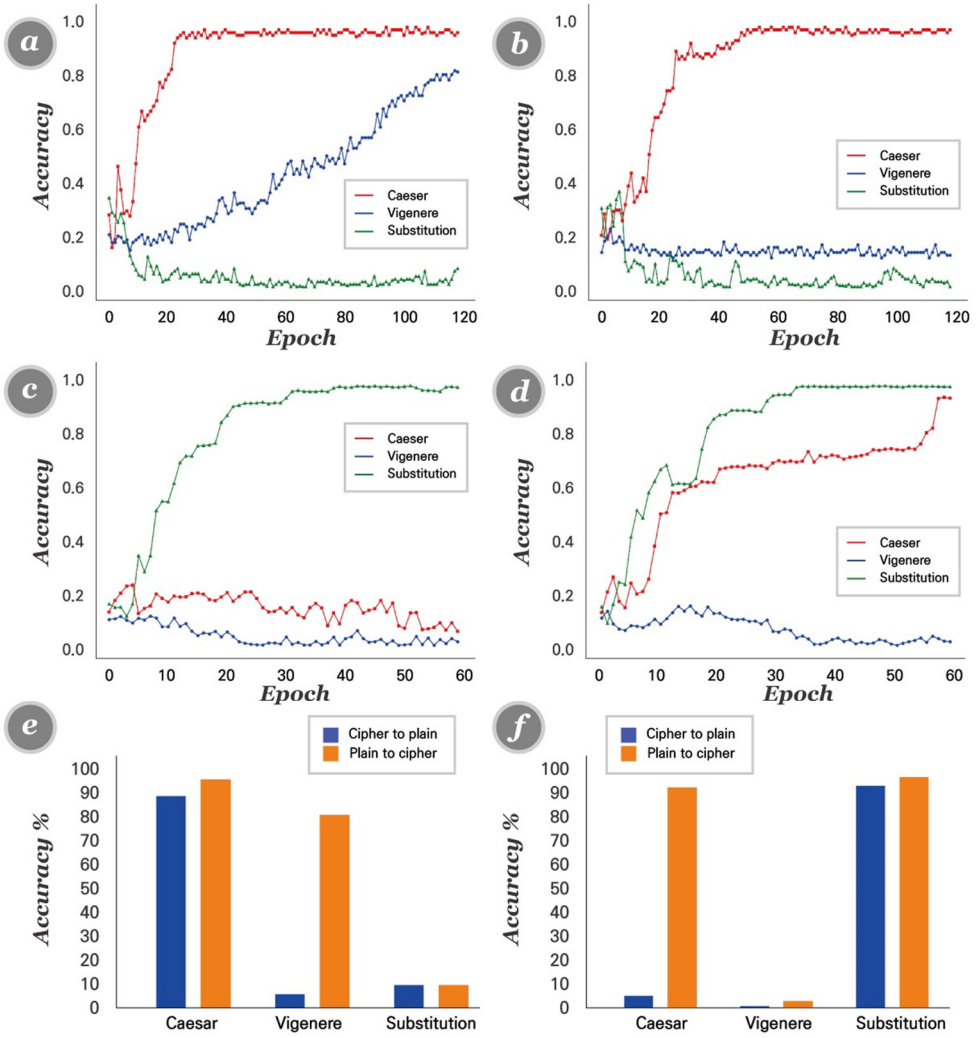

Figure 9. The impact of batch size on the training process. (**a**,**b**) network training process for a batch size of 8 for a cipher to plain and plain to cipher attacks, respectively; (**c**,**d**) network training process for a batch size of 128 for a cipher to plain and plain to cipher attacks, respectively; (**e**) proposed method test results for batch size 8; (**f**) proposed method test results for batch size 128.

Afterwards, we fixed other parameters and investigated the impact of embedding space W_{emb} and W_{time} on network training. In the first round of the experiment, we set the embedding spaces W_{emb} and W_{time} from 256 to 128 and run the network. Figure 10

demonstrates the network training process for different embedding spaces W_{emb} and W_{time}. Figure 10a and b show the network could converge for all types of ciphers but with a lower speed compared to the default setting for a cipher to plain and plain to cipher attacks, respectively. Afterwards, we set the embedding spaces W_{emb} and W_{time} from 256 to 512 and run the network, respectively. Figure 10c and d demonstrate the network training process for the embedding space W_{emb} and W_{time} of 512 for a cipher to plain and plain to cipher attacks, respectively. It has a large size compared to the default setting which demonstrates to be inefficient and did not converge for all ciphers. Figure 10e and f demonstrate the proposed method tests accuracy results for embedding spaces W_{emb} and W_{time} 128, and embedding spaces W_{emb} and W_{time} 512, respectively. Looking at the results in Figure 10, it shows that the results of learning can vary depending on the embedding spaces, and in particular, it can be seen that convergence is slow or does not converge at all, resulting in ciphers with very low accuracy.

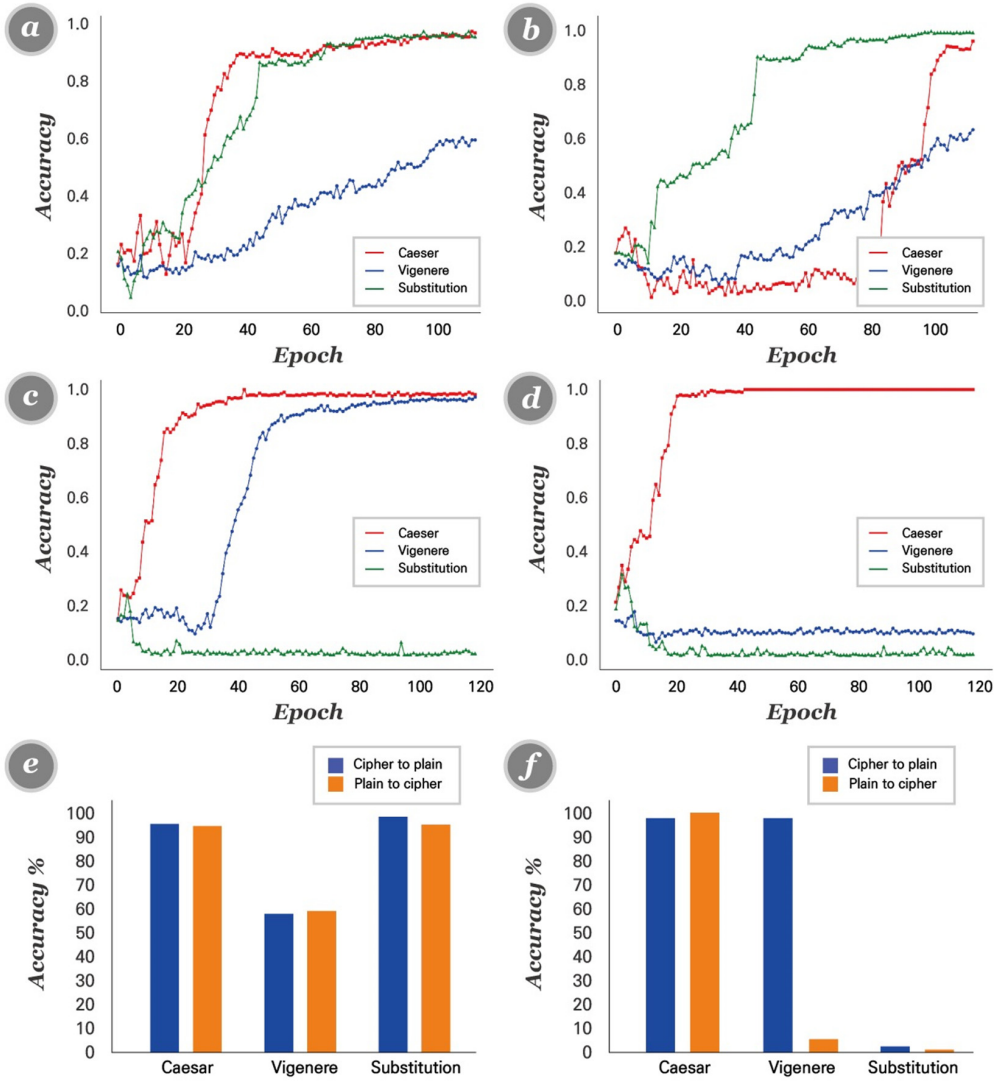

Figure 10. The proposed method training process of different sizes of embedding spaces W_{emb} and W_{time}. (**a**,**b**) proposed method training process on embedding spaces W_{emb} and W_{time} of 128, for a cipher to plain and plain to cipher attacks, respectively; (**c**,**d**) proposed method training process on embedding spaces W_{emb} and W_{time} of 512 for a cipher to plain and plain to cipher attacks, respectively; (**e**) tests accuracy results for embedding spaces W_{emb} and W_{time} 128; (**f**) embedding spaces W_{emb} and W_{time} 512.

4.7. Model Comparison

We conducted a comparison of the proposed model GAN-based state-of-the-art and the existing models. The Pix2Pix model and the CipherGAN model are the popular GAN-based cryptanalysis models we used for model comparison. In effect, the original Pix2Pix model is based on supervised learning which is trained in one direction using a paired dataset. To compare the model between supervised and unsupervised settings, we directly implement an improved Pix2Pix model with our embedding technique for cryptanalysis. Therefore, the Pix2Pix model can only learn one cipher at one training, in one direction (plain to cipher or cipher to plain). CipherGAN is based on unsupervised learning using the cycleGAN structure. Unlike supervised learning, unsupervised learning is trained in both directions (plain to cipher and cipher to plain) using an unpaired dataset. However, CipherGAN can also be trained for only one cipher in one training. Pix2Pix is a KPA scenario because it requires a paired dataset, and CipherGAN is a COA scenario because it uses an unpaired dataset.

Figure 11 shows the test results trained with the Pix2Pix model and the CipherGAN model. Since the Pix2Pix model learns only one cipher in one training and is unidirectional, the results of each training are combined and displayed. The results of cipher to plain and plain to cipher are very similar (See Figure 11a,b). In both processes, Caeser and Substitution approached 100% when epochs exceeded 30, and Vigenère approached 100% at 70 epochs. Therefore, a total of six training sessions are required to attack three ciphers using the Pix2Pix model. CipherGAN also shows the combined results of each training because only one cipher is learned in one training (See Figure 11c,d). CipherGAN, like Pix2Pix, has similar results of cipher to plain and plain to cipher. However, the test results of Substitution are different. Pix2Pix was trained stably and showed a graph with gradually increasing accuracy, while CipherGAN dropped once after 40 epochs and then rapidly rose and changed to a stable shape. In the case of Vigenère, training was completed last, the same as Pix2Pix. The bar graph in Figure 11e and f shows the highest test accuracy of each cipher for each model. All results show that over 99% of the model can be trained to break each cipher. This shows that once the model is trained, it can eventually be decrypted. Although there is a difference in the size of the epoch to be trained, it can be seen that the proposed model reaches high accuracy much faster when viewed as a training process for all ciphers. Table 8 shows the test results of our proposed model and the test results of Pix2Pix and CipherGAN, which are comparative models. However, UC-GAN can be considered more efficient because it simultaneously broke three ciphers in one training.

Table 8. Experimental results of the Pix2Pix, CipherGAN, and UC-GAN.

Emulation Method	Target	Network Model Accuracy (%)		
		Pix2Pix	**CipherGAN**	**UC−GAN**
Single cipher To Plain	Caesar to Plain	99.96	99.53	99.40
	Vigenere to Plain	99.84	99.79	98.33
	Substitution to Plain	99.84	99.45	98.71
Plain to Single cipher	Plain to Caesar	99.95	99.44	98.37
	Plain to Vigenere	99.84	99.79	97.72
	Plain to Substitution	99.84	99.45	99.82

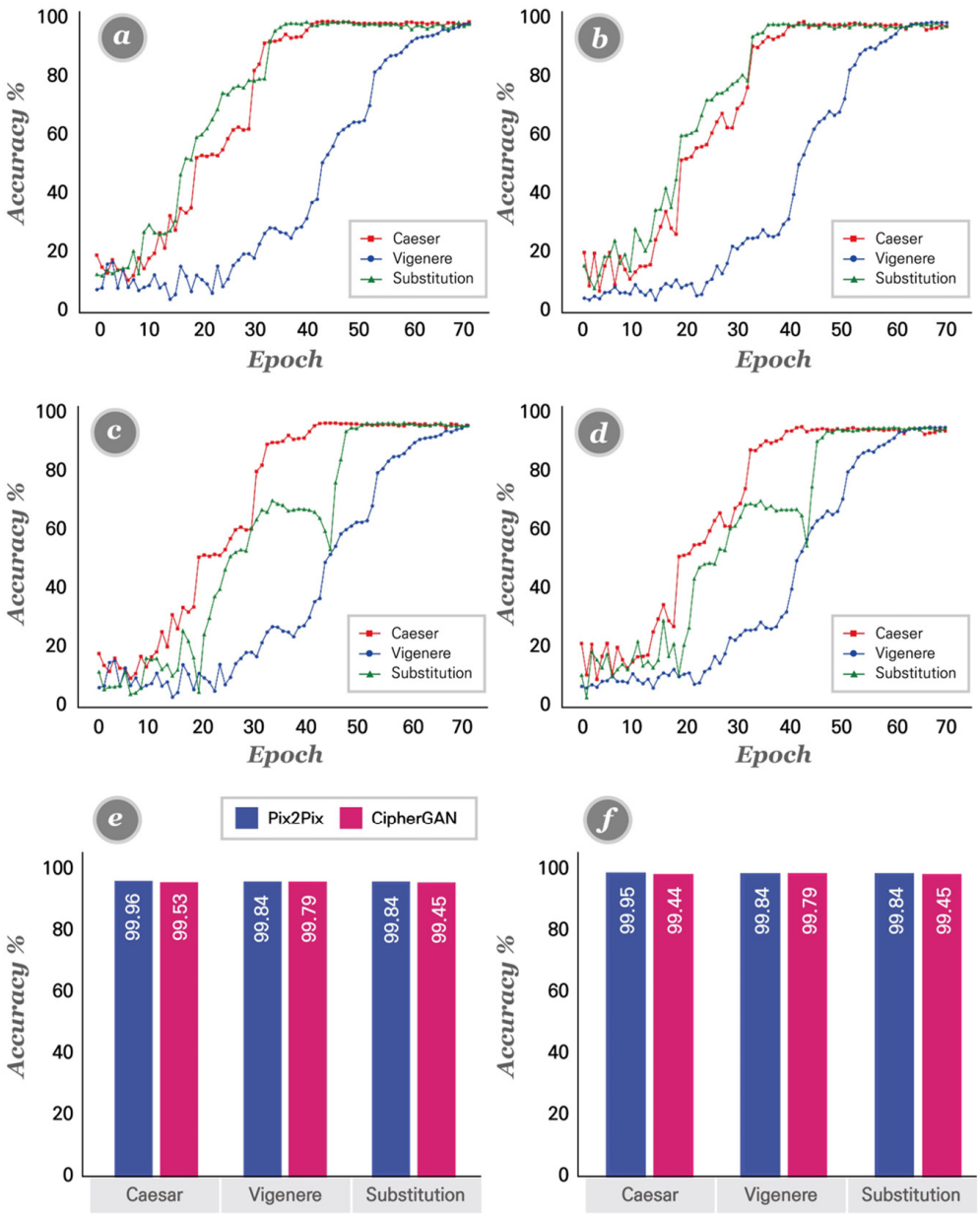

Figure 11. Test accuracy comparison of the Pix2Pix network (KPA) model with the CipherGAN (COA) network model. (**a**) Ciphertext-to-plaintext result with Pix2Pix network model; (**b**) plaintext-to-ciphertext result using Pix2Pix network mode; (**c**) plaintext-to-ciphertext result using CipherGAN; (**d**) plaintext-to-ciphertext using CipherGAN; (**e**) Test accuracy result of the Pix2Pix network model in plaintext-to-ciphertext recovery against the CipherGAN model; (**f**) test accuracy result of the Pix2Pix network model in ciphertext-to-plaintext recovery against the CipherGAN model.

5. Discussion

We suggested a novel unsupervised deep-learning model that is more flexible and efficient than existing information extraction techniques for translating ciphertext-to-plaintext across various cipher domains. The proposed model is based on generative adversarial networks (GANs). As described above, by competing for a generative deep neural network against a discriminative deep neural network, the GAN model creates samples that seem to come from the training set. Especially, CipherGAN is a GAN-based model that is an unsupervised cryptanalytic method to break substitution ciphers without any prior knowledge. However, CipherGAN requires the number of

generators and discriminators to be in proportion to the number of ciphertext/plaintext domains. Unlike CipherGAN, we proposed UC−GAN, which consists of a unified generator and a discriminator using only a single deep neural network for multiple domains. The proposed UC−GAN model was able to perform extremely well on multiple substitution ciphers, achieving near-flawless accuracy. The experimental results were carried out using three types of classical ciphers Caesar, Vigenere, and Substitution ciphers. In addition, we compared the model performance of our model with that of the CycleGAN model. The experimental results demonstrate that the proposed UC−GAN model can perform similar test accuracy results to the CipherGAN model, but unlike CipherGAN and the proposed UC-GAN uses only a single domain. This means that the efficiency of emulation in multiple domains is 3 times faster than CipherGAN. Second, UC−GAN shows that discrete GANs can be evolved much more like the original GANs. In addition, as shown in Figure 6, the UC−GAN model is notably insensitive to the random initialization of weights and demonstrates constant trends in testing. Furthermore, we experimentally compared the performance of the Pix2Pix network model with that of the CipherGAN network model for cracking classical substitution ciphers. The experiment was conducted under two possible circumstances; first, cipher-to-plain, and second, plain-to-cipher. In both cases, the Pix2Pix network model performed relatively better than the CipherGAN model. Our proposed model usage is not limited only to cipher cracking, but can also be utilized in other multi-domain networks, such as image generation, voice convertor, or automated speech translation for multiple audiences.

6. Conclusions

In this paper, we proposed a fully automated multi-domain cipher cracking system for substitution ciphers without any prior knowledge, such as language frequencies. Our approach is based on generative adversarial networks (GANs). We use GANs in a new way by utilizing a unified generator and discriminator of the proposed UC−GAN in just one deep neural network for multi-domain cryptanalysis. In this paper, we showed that the proposed model has great breaking emulation results for multi-domain cryptanalysis. We evaluated the proposed method using three types of classical ciphers: the Caesar, Vigenere, and Substitution ciphers. The experimental results demonstrated that the proposed model was able to crack multiple ciphers in a single training process. The proposed model showed more than 97% in overall accuracy and more than 0.96 in weighted F1-score considering unbalanced character frequencies. Furthermore, we showed that the proposed model has very stable training results with hyperparameter tuning in several experiments.

Block ciphers are one of the important encryption systems in modern cryptography for data confidentiality. The block ciphers are applied to a wide variety of practical applications that are highly secure with complicated structures. Therefore, in the future, we plan to utilize the current method to crack block ciphers or a component of block ciphers such as S-boxes, which is the main component of block ciphers.

Author Contributions: Conceptualization, H.K. and I.M.; methodology, S.P. and H.K.; software, S.P. and H.K.; validation, H.K. and S.P.; formal analysis, I.M.; writing–original draft preparation, H.K. and S.P.; writing–review and editing, I.M.; visualization, S.P. and H.K.; supervision, I.M.; funding acquisition, I.M. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by an Institute of Information & Communications Technology Planning & Evaluation (IITP) grant, funded by the Korea Government (MSIT) (No.2020-0-00126, Research on AI-based cryptanalysis and security evaluation).

Data Availability Statement: The source codes and datasets can be found at the github address (https://github.com/tdn02007/UC-GAN-Unified-cipher-generative-adversarial-network).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Courtois, N.T.; Oprisanu, M.-B.; Schmeh, K. Linear cryptanalysis and block cipher design in East Germany in the 1970s. *Cryptologia* **2019**, *43*, 2–22. [CrossRef]
2. Al-Shabi, M. A survey on symmetric and asymmetric cryptography algorithms in information security. *Int. J. Sci. Res. Publ. (IJSRP)* **2019**, *9*, 576–589. [CrossRef]
3. Wu, R.; Gao, S.; Wang, X.; Liu, S.; Li, Q.; Erkan, U.; Tang, X. Aea-NCS: An audio encryption algorithm based on a nested chaotic system. *Chaos Solitons Fractals* **2022**, *165*, 112770. [CrossRef]
4. Gao, S.; Wu, R.; Wang, X.; Liu, J.; Li, Q.; Tang, X. EFR-CSTP: Encryption for face recognition based on the chaos and semi-tensor product theory. *Inf. Sci.* **2023**, *621*, 766–781. [CrossRef]
5. Ahmadzadeh, E.; Kim, H.; Jeong, O.; Moon, I. A novel dynamic attack on classical ciphers using an attention-based LSTM encoder-decoder model. *IEEE Access* **2021**, *9*, 60960–60970. [CrossRef]
6. Ahmadzadeh, E.; Kim, H.; Jeong, O.; Kim, N.; Moon, I. A deep bidirectional LSTM-GRU network model for automated ciphertext classification. *IEEE Access* **2022**, *10*, 3228–3237. [CrossRef]
7. Chan, T.-H.; Jia, K.; Gao, S.; Lu, J.; Zeng, Z.; Ma, Y. PCANet: A simple deep learning baseline for image classification? *IEEE Trans. Image Process.* **2015**, *24*, 5017–5032. [CrossRef]
8. Ahmadzadeh, E.; Jaferzadeh, K.; Shin, S.; Moon, I. Automated single cardiomyocyte characterization by nucleus extraction from dynamic holographic images using a fully convolutional neural network. *Biomed. Opt. Express* **2020**, *11*, 1501–1516. [CrossRef]
9. Fomicheva, M.; Sun, S.; Yankovskaya, L.; Blain, F.; Guzmán, F.; Fishel, M.; Aletras, N.; Chaudhary, V.; Specia, L. Unsupervised quality estimation for neural machine translation. *Trans. Assoc. Comput. Linguist.* **2020**, *8*, 539–555. [CrossRef]
10. Sirichotedumrong, W.; Kiya, H. A gan-based image transformation scheme for privacy-preserving deep neural networks. In Proceedings of the 2020 28th European Signal Processing Conference (EUSIPCO), Virtual, 18–22 January 2021; pp. 745–749.
11. Lu, J.; Li, N.; Zhang, S.; Yu, Z.; Zheng, H.; Zheng, B. Multi-scale adversarial network for underwater image restoration. *Opt. Laser Technol.* **2019**, *110*, 105–113. [CrossRef]
12. Zhu, J.-Y.; Park, T.; Isola, P.; Efros, A.A. Unpaired image-to-image translation using cycle-consistent adversarial networks. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2223–2232.
13. Falco, G.; Viswanathan, A.; Caldera, C.; Shrobe, H. A master attack methodology for an AI-based automated attack planner for smart cities. *IEEE Access* **2018**, *6*, 48360–48373. [CrossRef]
14. Rao, B.S.; Premchand, P. A Review on Combined Attacks on Security Systems. *Int. J. Appl. Eng. Res.* **2018**, *4562*, 16252–16278.
15. Wang, K.; Gou, C.; Duan, Y.; Lin, Y.; Zheng, X.; Wang, F.-Y. Generative adversarial networks: Introduction and outlook. *IEEE/CAA J. Autom. Sin.* **2017**, *4*, 588–598. [CrossRef]
16. Hong, Y.; Hwang, U.; Yoo, J.; Yoon, S. How generative adversarial networks and their variants work: An overview. *ACM Comput. Surv. (CSUR)* **2019**, *52*, 1–43. [CrossRef]
17. Mahdizadehaghdam, S.; Panahi, A.; Krim, H. Sparse generative adversarial network. In Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops, Seoul, Korea, 27–28 October 2019.
18. Li, W.; Ding, W.; Sadasivam, R.; Cui, X.; Chen, P. His-GAN: A histogram-based GAN model to improve data generation quality. *Neural Netw.* **2019**, *119*, 31–45. [CrossRef]
19. Yu, L.; Zhang, W.; Wang, J.; Yu, Y. Seqgan: Sequence generative adversarial nets with policy gradient. In Proceedings of the AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 4–9 February 2017.
20. Tang, H.; Xu, D.; Liu, G.; Wang, W.; Sebe, N.; Yan, Y. Cycle in cycle generative adversarial networks for keypoint-guided image generation. In Proceedings of the 27th ACM International Conference on Multimedia, Nice, France, 21–25 October 2019; pp. 2052–2060.
21. Creswell, A.; White, T.; Dumoulin, V.; Arulkumaran, K.; Sengupta, B.; Bharath, A.A. Generative adversarial networks: An overview. *IEEE Signal Process. Mag.* **2018**, *35*, 53–65. [CrossRef]
22. Cai, Z.; Xiong, Z.; Xu, H.; Wang, P.; Li, W.; Pan, Y. Generative adversarial networks: A survey toward private and secure applications. *ACM Comput. Surv. (CSUR)* **2021**, *54*, 1–38. [CrossRef]
23. Gomez, A.N.; Huang, S.; Zhang, I.; Li, B.M.; Osama, M.; Kaiser, L. Unsupervised cipher cracking using discrete gans. *arXiv* **2018**, arXiv:1801.04883.
24. Choi, Y.; Choi, M.; Kim, M.; Ha, J.-W.; Kim, S.; Choo, J. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 8789–8797.
25. Wang, H.; Wang, J.; Wang, J.; Zhao, M.; Zhang, W.; Zhang, F.; Li, W.; Xie, X.; Guo, M. Learning graph representation with generative adversarial nets. *IEEE Trans. Knowl. Data Eng.* **2019**, *33*, 3090–3103. [CrossRef]
26. Mirza, M.; Osindero, S. Conditional generative adversarial nets. *arXiv* **2014**, arXiv:1411.1784.
27. Abdelmotaal, H.; Abdou, A.A.; Omar, A.F.; El-Sebaity, D.M.; Abdelazeem, K. Pix2pix conditional generative adversarial networks for scheimpflug camera color-coded corneal tomography image generation. *Transl. Vis. Sci. Technol.* **2021**, *10*, 21. [CrossRef] [PubMed]

28. Welander, P.; Karlsson, S.; Eklund, A. Generative adversarial networks for image-to-image translation on multi-contrast mr images-a comparison of cyclegan and unit. *arXiv* **2018**, arXiv:1806.07777.
29. Zhu, M.; Gong, S.; Qian, Z.; Zhang, L. A brief review on cycle generative adversarial networks. In Proceedings of the 7th IIAE International Conference on Intelligent Systems and Image Processing (ICISIP), Taiwan, 5–9 September 2019; pp. 235–242.
30. Kaneko, T.; Kameoka, H.; Tanaka, K.; Hojo, N. Stargan-vc2: Rethinking conditional methods for stargan-based voice conversion. *arXiv* **2019**, arXiv:1907.12279.
31. Choi, Y.; Uh, Y.; Yoo, J.; Ha, J.-W. Stargan v2: Diverse image synthesis for multiple domains. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 8188–8197.
32. Abd, A.J.; Al-Janabi, S. Classification and identification of classical cipher type using artificial neural networks. *J. Eng. Appl. Sci.* **2019**, *14*, 3549–3556. [CrossRef]
33. Baek, S.; Kim, K. Recent advances of neural attacks against block ciphers. In Proceedings of the 2020 Symposium on Cryptography and Information Security (SCIS 2020), Kochi, Japan, 28–31 January 2020.
34. Gohr, A. Improving attacks on round-reduced speck32/64 using deep learning. In Proceedings of the Advances in Cryptology–CRYPTO 2019: 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, 18–22 August 2019; pp. 150–179.
35. Baksi, A.; Baksi, A. Machine learning-assisted differential distinguishers for lightweight ciphers. In *Classical and Physical Security of Symmetric Key Cryptographic Algorithms*; Springer: Singapore, 2022; pp. 141–162.
36. Ding, Y.; Tan, F.; Qin, Z.; Cao, M.; Choo, K.-K.R.; Qin, Z. DeepKeyGen: A deep learning-based stream cipher generator for medical image encryption and decryption. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, *33*, 4915–4929. [CrossRef]
37. Panwar, K.; Kukreja, S.; Singh, A.; Singh, K.K. Towards deep learning for efficient image encryption. *Procedia Comput. Sci.* **2023**, *218*, 644–650. [CrossRef]
38. Isola, P.; Zhu, J.-Y.; Zhou, T.; Efros, A.A. Image-to-image translation with conditional adversarial networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 1125–1134.
39. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. In Proceedings of theAdvances in Neural Information Processing Systems 30 (NIPS 2017), Long Beach, CA, USA, 4–9 December 2017; Volume 30.
40. Mao, X.; Li, Q.; Xie, H.; Lau, R.Y.; Wang, Z.; Paul Smolley, S. Least squares generative adversarial networks. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2794–2802.
41. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative adversarial networks. *Commun. ACM* **2020**, *63*, 139–144. [CrossRef]
42. Arjovsky, M.; Chintala, S.; Bottou, L. Wasserstein generative adversarial networks. In Proceedings of the International Conference on Machine Learning, Sydney, Australia, 6–11 August 2017; pp. 214–223.
43. CrypTool Portal:Cryptography for everybody. Available online: https://www.cryptool.org/en/ (accessed on 2 September 2022).