applied sciences

MDPI

Article

Optimizing GANs for Cryptography: The Role and Impact of Activation Functions in Neural Layers Assessing the Cryptographic Strength

Purushottam Singh, Sandip Dutta and Prashant Pranav *

Department of Computer Science and Engineering, Birla Institute of Technology, Mesra, Ranchi 835215, India; purushottamsingh@outlook.com (P.S.); sandipdutta@bitmesra.ac.in (S.D.)
* Correspondence: prashantpranav19@gmail.com

Abstract: Generative Adversarial Networks (GANs) have surfaced as a transformative approach in the domain of cryptography, introducing a novel paradigm where two neural networks, the generator (akin to Alice) and the discriminator (akin to Bob), are pitted against each other in a cryptographic setting. A third network, representing Eve, attempts to decipher the encrypted information. The efficacy of this encryption–decryption process is deeply intertwined with the choice of activation functions employed within these networks. This study conducted a comparative analysis of four widely used activation functions within a standardized GAN framework. Our recent explorations underscore the superior performance achieved when utilizing the Rectified Linear Unit (ReLU) in the hidden layers combined with the Sigmoid activation function in the output layer. The non-linear nature introduced by the ReLU provides a sophisticated encryption pattern, rendering the deciphering process for Eve intricate. Simultaneously, the Sigmoid function in the output layer guarantees that the encrypted and decrypted messages are confined within a consistent range, facilitating a straightforward comparison with original messages. The amalgamation of these activation functions not only bolsters the encryption strength but also ensures the fidelity of the decrypted messages. These findings not only shed light on the optimal design considerations for GAN-based cryptographic systems but also underscore the potential of investigating hybrid activation functions for enhanced system optimization. In our exploration of cryptographic strength and training efficiency using various activation functions, we discovered that the "ReLU and Sigmoid" combination significantly outperforms the others, demonstrating superior security and a markedly efficient mean training time of 16.51 s per 2000 steps. This highlights the enduring effectiveness of established methodologies in cryptographic applications. This paper elucidates the implications of these choices, advocating for their adoption in GAN-based cryptographic models, given the superior results they yield in ensuring security and accuracy.

Keywords: neural network; activation function; security; generative adversarial network; cryptography

Citation: Singh, P.; Dutta, S.; Pranav, P. Optimizing GANs for Cryptography: The Role and Impact of Activation Functions in Neural Layers Assessing the Cryptographic Strength. *Appl. Sci.* **2024**, *14*, 2379. https://doi.org/10.3390/app14062379

Academic Editor: Nuno Silva

Received: 11 October 2023
Revised: 18 November 2023
Accepted: 29 November 2023
Published: 12 March 2024

1. Introduction

Generative Adversarial Networks (GANs) have become the avant-garde of deep learning, pushing the frontiers of what machines can imagine. At its core, a GAN is a contest of wits between two neural networks—the generator, which strives to produce realistic data, and the discriminator, which endeavors to distinguish between genuine data and the fabrications of the generator. This tug-of-war leads to a symbiotic evolution, with the generator crafting ever-more convincing outputs, and the discriminator honing its discerning capabilities. The magic of GANs is in this adversarial process, where competition breeds sophistication, resulting in the creation of art, realistic images, and even music that is virtually indistinguishable from those produced by humans. Now, consider translating this adversarial magic into the domain of cryptography, giving birth to "GAN Cryptography".

Here, the art of encryption is treated not just as a mathematical exercise, but as an evolving, dynamic challenge. Alice, acting as our data "generator", crafts encrypted messages, while Eve, the "discriminator", works tirelessly to decrypt them without the key. But there is a twist. In the mix is Bob, who, armed with the correct decryption key, ensures that the communication remains intelligible to the intended recipient. The objective is clear: evolve encryption methodologies that are transparent to Bob yet inscrutable to Eve.

GANs introduced by Goodfellow et al. in 2014 [1] have emerged as a groundbreaking advancement in the field of deep learning. By creating a two-player minimax game involving a generator and a discriminator, GANs have enabled the generation of highly realistic data, unlocking new potentials in various applications such as image synthesis, data augmentation, artistic creations, and most recently in cryptography.

The essence of GAN Cryptography is in harnessing the adversarial power of GANs to revolutionize secure communication. Just as GANs have advanced the realms of image generation and style transfer, GAN Cryptography promises a dynamic and ever-adapting approach to security. Traditional cryptographic methods rely on fixed algorithms and the belief in the computational difficulty of certain tasks. In contrast, GAN Cryptography introduces a fluidity where cryptographic methods adapt and evolve based on potential threats, ensuring a level of security that is proactive rather than reactive.

In the vast landscape of artificial intelligence, GANs stand out as a beacon of innovation, blending competition and cooperation to achieve excellence. By extending this paradigm to cryptography, we are not only reimagining how we view secure communication but also championing a future where security is dynamic, robust, and ever-evolving.

The integration of GANs with cryptography is an innovative intersection that has drawn considerable attention in the research community. The core idea of adversarial training resonates with cryptographic principles, where competing entities strive to challenge and surpass each other, mirroring the cryptographic struggle between encoding and decoding, or hiding and revealing information.

Abadi and Andersen [2] explored the concept of adversarial neural cryptography, where neural networks are trained to communicate securely amidst the presence of an adversary. Their work laid the foundation for a series of subsequent investigations into the cryptographic properties of neural networks and GANs.

In the realm of data security, the coupling of GANs with cryptographic techniques has led to new methodologies for secure data transmission [3] and privacy-preserving data synthesis [4]. Furthermore, GANs have been employed in cryptographic function approximation [5], demonstrating their utility in approximating complex cryptographic primitives.

The development of cryptographic GANs has also spawned new avenues for research in digital watermarking [6] and steganography [7], strengthening the ties between deep learning and cryptographic techniques.

From a theoretical perspective, the interplay between GANs and cryptography prompts exciting questions concerning the security guarantees and potential vulnerabilities of neural networks. Several researchers have delved into the understanding of adversarial attacks and defenses within GANs, illuminating new facets of neural security [8,9].

The combination of Generative Adversarial Networks with cryptography represents an interdisciplinary nexus at the cutting edge of artificial intelligence and information security. It is a burgeoning field replete with challenges and opportunities, promising to redefine traditional paradigms and forge novel pathways for technological advancement.

In the burgeoning field of neural-network-based cryptography, particularly within the realm of Generative Adversarial Networks (GANs), a critical research gap persists in understanding the impact of diverse activation function configurations on cryptographic strength and efficiency. This study embarks on an explorative journey to fill this void, meticulously analyzing the effects of various activation function pairs such as "LeakyReLU and tanh", "Mish and Sigmoid", and "ReLU and Sigmoid" on both hidden and output layers. Our research challenges entrenched cryptographic norms and illuminates the intricate relationship between neural architecture and encryption robustness. By systematically test-

ing these diverse combinations, our work not only contributes to the theoretical landscape of GAN-based cryptography but also sets new benchmarks for secure communication, marking a pivotal step in the evolution of impregnable encryption methods. Hence, this investigation stands as a critical endeavor in unearthing the potential of neural cryptography, potentially revolutionizing data security in our increasingly digital world.

2. Related Works

As we know, Cryptography, derived from the Greek words for "secret" and "writing", encompasses a broad array of techniques designed to ensure the privacy, authenticity, integrity, and non-repudiation of information. The rise of digital networks has facilitated unprecedented levels of interconnectedness and data exchange; however, it has also led to a corresponding surge in the demand for secure communications. Cryptography provides a framework to meet this need, but its understanding, application, and evolution remain subjects of intense scrutiny and research.

GANs and cryptography may initially appear as disparate fields, but there is a growing intersection between them that is forging new directions in secure data processing and protection. GANs, a class of machine learning models, are designed to mimic and generate data distributions that are similar to real data. When applied to cryptography, GANs can introduce robust methods for enhancing cryptographic algorithms, creating new ways to secure data, and even providing innovative approaches to breaking cryptographic systems.

2.1. Cryptographic Mechanisms

At the core of cryptography lies the ability to transform information into an unreadable format using algorithms and mathematical functions, rendering it inaccessible to unauthorized parties. There are two primary forms of cryptography; these forms are shown in Figure 1:

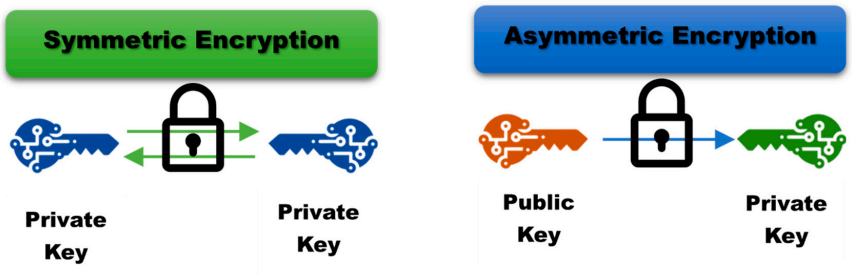

Figure 1. Symmetric and asymmetric encryption.

2.2. Symmetric Cryptography

It is a foundational pillar of modern cryptographic systems, employs a singular, shared, secret key for both the encryption and decryption processes. This mechanism ensures streamlined, efficient, and robust protection of data during transmission and storage. However, its strength is equally its vulnerability; the security of the entire system rests on the confidentiality and secure management of this single key. Consequently, while symmetric cryptography offers high speed and computational efficiency, its centralized key management demands rigorous security measures to guard against potential breaches and unauthorized access.

2.3. Asymmetric Cryptography

It is a cornerstone of contemporary digital security, operating on a dual-key mechanism: a public key for encryption and a distinct private key for decryption. This groundbreaking structure ensures that even if a malicious actor intercepts the encrypted data, they cannot decipher it without the closely guarded private key. Notably, this approach mitigates key distribution challenges inherent in symmetric systems, allowing for secure communication between parties who have never met. However, its computational intensity can be a

trade-off in terms of speed. While asymmetric cryptography fundamentally revolutionized digital trust by enabling secure public key exchanges and digital signatures, it necessitates rigorous private key protection to maintain its security fortress against potential intrusions.

2.4. Need for Cryptography

The digital era has brought about significant opportunities as well as challenges. From protecting personal information and financial transactions to ensuring the confidentiality of sensitive government and corporate data, cryptography plays an essential role in preserving the integrity of digital interactions. The increasing sophistication of cyber-attacks, coupled with the growth of IoT devices and quantum computing, necessitates continuous research and innovation in cryptography.

2.5. Major Contribution

The field of cryptography is marked by rapid advancements and deep intellectual challenges. Prominent research contributions include the following:

Shor's Algorithm: this quantum algorithm, capable of breaking widely used cryptographic protocols, has urged the development of post-quantum cryptography [10].

Fully Homomorphic Encryption: enabling computations on encrypted data without decryption, this breakthrough has extensive applications in secure data processing [11].

Blockchain and Cryptocurrencies: The cryptographic principles behind digital currencies like Bitcoin have ignited research in decentralized security mechanisms [12].

The paper based on Generative Deep Neural Networks supporting clinical data sharing [13] outlines a method for privacy-preserving data sharing in medical research using GANs, exemplifying their application in ensuring data confidentiality.

The paper Adversarial Machine Learning for Cybersecurity and Computer Vision [14] presents insights into adversarial machine learning using GANs within cryptographic contexts, allowing for the development of advanced defense mechanisms.

The expanding array of cryptographic techniques, along with the increasing complexity of the digital landscape, underscores the criticality of ongoing research in this fascinating and ever-evolving field.

2.6. Neural Networks and GANs

Neural networks are like brains for computers, helping them learn from data. One exciting type of neural network is called a Generative Adversarial Network (GAN), where two networks learn by competing against each other. Now, imagine taking this competition into the world of cryptography, which is all about securing communication.

Figure 2 shows the working structure of the neural network. In this concept called GAN Cryptography, neural network principles are merged with security: one network generates secret keys while another evaluates them, enhancing data security through competitive learning and adaptation. It is like blending the magic of creating secret codes with a friendly competition to make the codes as strong as possible.

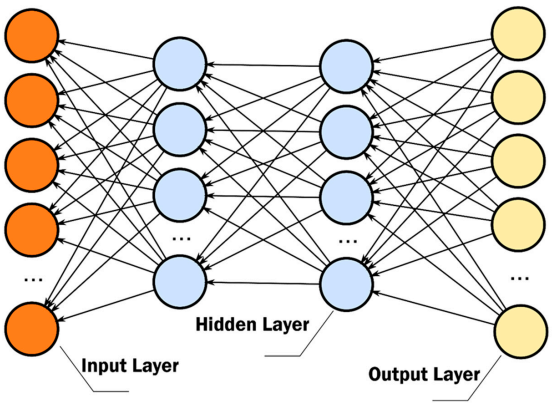

Figure 2. Structure of neural network.

3. Why GAN Cryptography?

The integration of GANs within the cryptographic paradigm heralds a transformative shift in the way we approach data security and privacy. With GANs' ability to generate, mimic, and adapt, the cryptographic landscape is set to experience a renaissance, unlocking novel methods of ensuring data integrity while simultaneously expanding the horizons of potential vulnerabilities. This convergence amplifies the urgency of comprehensive research and development in GAN Cryptography. As we stand on the precipice of a new era of interconnectedness, the need for GAN-enhanced cryptographic solutions is not merely an academic curiosity but a vital imperative for the digital age, securing our information ecosystem against evolving threats while fostering innovation, trust, and privacy in our increasingly connected world.

3.1. GAN Cryptography

Comprising two distinct neural network components, the generator and the discriminator, GANs have demonstrated unprecedented abilities in generating data that are almost indistinguishable from real data.

The generator is responsible for creating synthetic data, learning to mimic the true data distribution. It often includes multiple layers of neural networks, utilizing activation functions, batch normalization, and other advanced techniques to generate intricate data patterns [15].

The discriminator, on the other hand, functions as a critic, discerning between real and generated data. It is often structured as a deep neural network that incrementally learns the features and characteristics that define authentic data [16].

The synergy between these two neural network components leads to a continuous arms race where both entities evolve, giving rise to complex models that can recreate reality with astounding fidelity.

In the realm of cryptography, GANs have found an innovative and transformative application. Abadi and Andersen in 2016 were among the first to explore neural cryptography, using adversarial training to allow neural networks to devise encryption protocols. This pioneering work opened the door to the amalgamation of deep learning with traditional cryptographic objectives.

3.2. GAN Cryptography Working Mechanism

Generative Adversarial Network (GAN) cryptography is a relatively new and evolving field, representing a fusion of machine learning and cryptographic principles. Here is an overview of how GAN Cryptography works, broken down into several key components:

3.3. Structure of GANs

A GAN consists of two neural networks: a generator (G) and a discriminator (D), which are trained simultaneously through a competitive process:

In Figure 3, we can see the working structure of the Generative Adversarial Network.

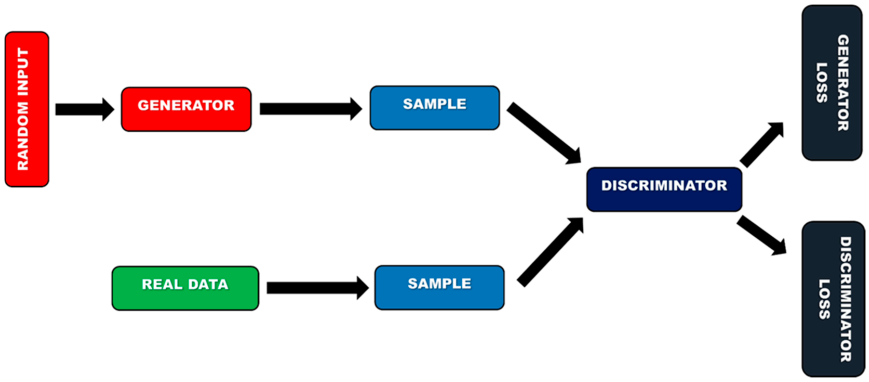

Figure 3. Structure of Generative Adversarial Network.

Generator: It tries to create data that are similar to some genuine data.

Discriminator: It tries to distinguish between genuine data and fake data created by the generator.

4. Organizational Experimental Setup

In the realm of security, a standard situation includes three participants: Alice, Bob, and Eve, depicted in Figure 1. Alice wants to send Bob a single confidential message, referred to as plaintext (P), while Eve tries to intercept it. Alice processes this plaintext to produce ciphertext (C), with both Bob and Eve receiving C and attempting to recover P. Their respective computations are represented by P_{Bob} and P_{Eve}. Unlike Eve, Alice and Bob share a secret key (K), treated as an additional input for them. They may use one unique key per plaintext, but the lengths of K and P are not necessarily the same. Interestingly, all three parties—Alice, Bob, and Eve—are modelled as neural networks with different parameters $(\theta_A, \theta_B, \theta_C)$, which means that encryption and decryption may not be identical functions even if Alice and Bob have the same structure. They operate over floating-point numbers instead of sequences of bits, leading to the possibility that C, P_{Bob}, and P_{Eve} might be arbitrary floating-point numbers, even though P and K may consist of 0s and 1s. Although constraints, in practice, limit these values to a specific range (e.g., -1 to 1), intermediate values are permitted.

In Figure 4, we can see the flowchart of the experimental setup. Despite exploring alternative methods, the essential focus is not on those, but on a rudimentary setup that enables the possibility of Alice and Bob using K as a one-time pad, encrypting and decrypting simply by XORing the key K with the plaintext P and the ciphertext C, respectively. However, Alice and Bob are not mandated to function this way, and indeed, further experiments might reveal other schemes. This configuration, while basic, serves foundational designs, ignoring aspects like key generation from a seed or the use of randomness for probabilistic encryption. Such enhancements could be explored in future work, broadening the scope and complexity of this standard security situation.

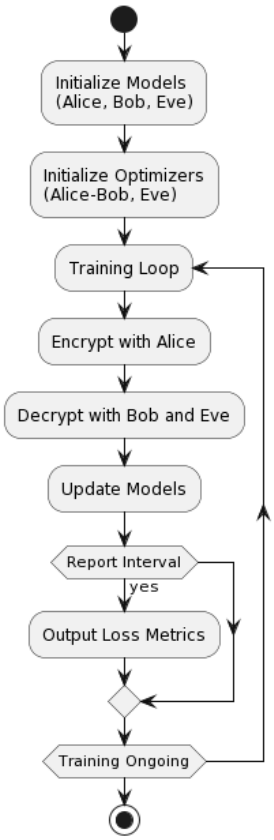

Figure 4. Flowchart of experimental setup.

4.1. Loss Function

In the rapidly evolving landscape of cryptography, the integration of adversarial methodologies, inspired by Generative Adversarial Networks (GANs), introduces a paradigm shift in ensuring secure communication. At the heart of this transformation lies the loss function, an essential tool to gauge and guide the performance of cryptographic models. Within the GAN framework, the loss function serves as a battlefield where two entities, the generator and the discriminator, contest their strengths and weaknesses. Translating this to the cryptographic domain, Alice's encryption and Eve's decryption processes are the adversarial counterparts, with Bob acting as a neutral evaluator of Alice's success. The loss function's role in GAN Cryptography transcends traditional metrics. It is the pivot around which the security dynamics revolve, guiding the cryptographic models to their optimal performance and ensuring robust, secure communication in an adversarial setting

We denote $A(m,\ k)$ as Alice's encryption function on input message m and key k, and $B(c,\ k)$ as Bob's decryption function on input ciphertext c and key k. Similarly, we write $E(c)$ for Eve's decryption function on input c.

We use a loss function L on decrypted messages [17]. We also use the mean absolute error (L1 distance) for this function defined in Equation (1):

$$L(m, m') = \frac{1}{N}\sum_{i=1}^{N}|m_i - m_i'| \tag{1}$$

where N is the length of the message.

Loss functions for Alice and Bob and for Eve are then defined as in Equation (2):

- Alice and Bob's loss:

$$L_{AB}(m,k) = L(m, B(A(m,k),k)) \tag{2}$$

Intuitively, this represents the error in Bob's decryption of the message encrypted by Alice. The goal for Alice and Bob is to minimize this loss.

- Eve's loss:

$$L_E(m,k) = L(m, E(A(m,k))) \tag{3}$$

This represents the error in Eve's decryption of the message encrypted by Alice. The goal for Eve is also to minimize this loss, unlike in some cryptographic settings where Alice and Bob would try to maximize Eve's error as defined in Equation (3).

The "optimal" Alice and Bob are obtained by minimizing the loss L_{AB} with respect to the parameters of Alice's and Bob's models as defined in Equation (4):

$$(OA, OB) = \arg min_{m,k}(L_{AB}(m,k)) \tag{4}$$

Similarly, the "optimal" Eve is obtained by minimizing the loss L_E with respect to the parameters of Eve's model as defined in Equation (5):

$$OE = \arg min_{m,k}(L_E(m,k)) \tag{5}$$

The training process consists of alternating updates to Alice and Bob's parameters to minimize L_{AB} and to Eve's parameters to minimize L_E. This process starts with random initializations for Alice, Bob, and Eve, and iteratively moves towards the optimal solutions (or close to them).

While the detailed theoretical framework might involve additional complexities, such as adversarial relationships between the parties, this code implements a more straightforward optimization of the two separate loss functions. Both Alice and Bob, and Eve, are independently trying to minimize their respective errors.

4.2. Training

For training the model, we used a system configuration with the following parameters:

System Specification:
Processor: AMD Ryzen 7 5800X (AMD, Santa Clara, CA, USA).
GPU: Nvidia RTX 3070 (NVIDIA, Sunnyvale, CA, USA).
RAM: 16.00 GB.
Operating System: 64-bit Windows 11.

All the implementations in this work were performed in Python 3 and MATLAB R2023a for the analysis of the results.

Training is conducted using a "minibatch" size of 100 entries. TensorFlow's Adam optimizer is employed, utilizing its default learning rate, typically 0.001. We do not set a specific learning rate or control its reduction over time, allowing Alice, Bob, and Eve to adapt to changes in each other's parameters and hopefully achieve a robust solution.

Adam, which stands for "Adaptive Moment Estimation", combines the ideas of two other optimization algorithms: Momentum and RMSprop [18]. It maintains a moving average of both the gradients (to get the momentum-like effect) and the squared gradients (to get the RMSprop-like effect) [19] as shown in Equation (6).

Given

$$\begin{gathered} g_t = gradient\ at\ timestep \\ m^t = moving\ average\ of\ the\ gradient \\ v_t = moving\ average\ of\ the\ squared\ gradient \\ \beta_1,\ \beta_2 = decay\ rates\ (typically\ close\ to\ 1) \\ \alpha = learning\ rate \\ \in = small\ constant\ to\ avoid\ division\ by\ zero \end{gathered} \tag{6}$$

The updates are computed as

1. $m_t = \beta_1 \cdot m_{t-1} + (1-\beta_1) \cdot g_t \rightarrow$ *Moving average of the gradient*.
2. $v_t = \beta_2 \cdot v_{t-1} + (1-\beta_2) \cdot g_t^2 \rightarrow$ *Moving average of the squared gradient*.
3. $\hat{m}_t = \frac{m_t}{1-\beta_1^t} \rightarrow$ Bias-corrected *moving average of the gradient*.
4. $\hat{v}_t = \frac{v_t}{1-\beta_2^t} \rightarrow$ *corrected moving average of the squared gradient*.
5. Weight update: $\theta_{t+1} = \theta_t - \alpha \cdot \frac{\hat{m}_t}{\sqrt{v_t}+\in}$.

The training process updates both Alice and Bob's models and Eve's model once per training step. Unlike other approaches, this lack of a specified ratio between the training of Alice–Bob and Eve does not afford any computational advantage to Eve.

The different experimental framework is an exploration into the promising realm of adversarial neural cryptography. Unlike traditional cryptographic algorithms that rely on mathematical theorems and principles [20], this setup explores the ability of neural networks to evolve and form secure communication channels on their own.

4.3. Experimental Setup 1

The first experimental setup utilizes "LeakyReLU" activation for the hidden layer and "tanh" activation for the output layer.

The architecture of Alice, Bob, and Eve is designed to investigate the potential of neural networks to develop a secure communication scheme, without rigidly adhering to any specific cryptographic algorithm. The model chosen for Alice and Bob features a "concatenate and transform" design, commencing with a concatenation of the plaintext and key bits, followed by a fully connected (FC) layer with 64 units, where each output can be a linear blend of the input bits, thus encouraging—but not imposing—interplay between the key and plaintext bits subsequent to the FC layer [21].

The Leaky Rectified Linear Unit (LeakyReLU) is a type of activation function that is a variant of the Rectified Linear Unit (ReLU). The formula is shown in Equation (7):

$$f(x) = \begin{cases} x,\ if\ x > 0 \\ \alpha x,\ x \leq 0 \end{cases} \tag{7}$$

where

x is the input to the function.

α is a small positive constant (typically very small, around 0.01) that determines the slope of the function for negative values of x. It is called "leaky" because it allows a small gradient when the unit is not active (i.e., when $x \leq 0$), which can help mitigate the "dying ReLU" problem, where neurons can sometimes get stuck during training and stop updating their weights altogether [22].

In the used method, the following applies:

1. Relu: this is the standard ReLU function defined as $f(x) = \max(0, x)$.

2. Sigmoid: this function outputs values between 0 and 1 and is defined as s in Equation (8).

$$f(x) = \frac{1}{1 + e^{-x}} \tag{8}$$

LeakyReLU activation permits a nuanced activation function [23], succeeded by another FC layer, resulting in an output size appropriate for either encrypted or decrypted plaintext, without predefining the transformation function. The architecture for Eve parallels that of Alice and Bob, differing only in the absence of key input. Unlike conventional methods, the design encourages the learning of bit combinations rather than prescribing them [24]. The decision to use LeakyReLU and tanh activation functions was deliberate to offer a more expressive representation. The models refrain from further restrictions, such as tying parameters, to foster an environment where the network learns encryption, decryption, and eavesdropping naturally [25].

The use of LeakyReLU instead of ReLU and the tanh activation function reflects the model architecture. The models are trained over 500,000 steps, with progress being printed every 2000 steps. The training loop generates random messages and keys, encrypts the message, and then decrypts it using both Bob and Eve, computing the losses and updating the models accordingly.

Figure 5 shows that starting with an aberrant low at 3.16, the training predominantly showcases execution times clustering around the 18 to 20 s bracket. Although there are sporadic surges reaching up to 22.46 s, the system does not show a clear ascending or descending trajectory. The occasional fluctuations in performance hint at varying computational challenges or dynamic system responses. By the end of the training, a familiar pattern emerges, as performance often falls back to the 18 to 19 s range, suggesting a possible equilibrium or consistent operational capability at this level.

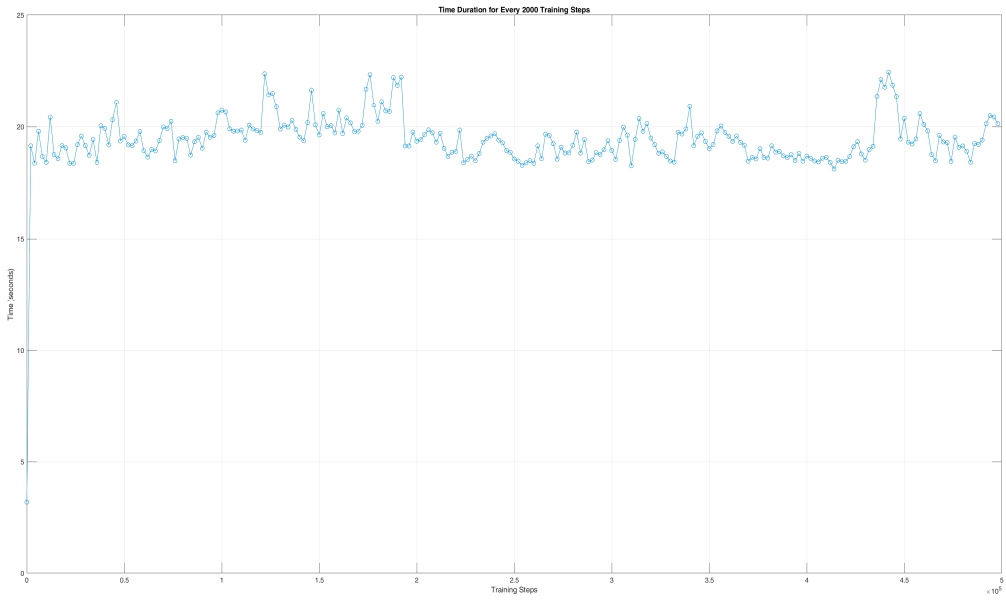

Figure 5. Time series analysis for training during experimental setup 1.

Figure 6 demonstrates the outcomes of a particularly compelling run, charting the progression of Bob's reconstruction error and Eve's reconstruction error against the number of training steps spanning from step 0 to step 500,000 for N = 16-bit plaintext and key pairs, utilizing a minibatch size of 100. Each data point on the curve corresponds to the average error amassed from 100 test instances. In a perfect scenario, we anticipate Bob's reconstruction error to descend to nil, while Eve's reconstruction error should ideally ascend to 8, translating to an error rate of 50% [26]. At the outset, as delineated by the graph, both reconstruction errors are considerably elevated. As training progresses, an intriguing dynamic unfolds: Alice and Bob begin to synchronize their communication efficiently, albeit this inadvertently provides Eve with an initial advantage, facilitating its deciphering prowess. A notable inflection point is observed close to step 150,000, marking a strategic adjustment by Alice and Bob to impede Eve's growing proficiency. Initiating at a loss of roughly 0.56, Alice–Bob's metrics gradually reduce to an approximate 0.015 by the 126,000th step. Similarly, Eve embarks from a close 0.014. By the juncture of step 200,000, the primary objectives of the training seem largely accomplished. The fluctuations in Alice–Bob's loss hover between 0.015 and 0.014, presenting no drastic shifts and hinting at a model that is finding its stability, albeit without converging sharply. Eve's trajectory also stabilizes, settling between 0.014 and 0.013. From a nebulous start, by the half-millionth step, Eve's eavesdropping prowess crystalizes, registering a definitive loss of 0.012. This is not a good sign for the security of the communication, but like the Alice–Bob loss, it is also not showing signs of drastically increasing. The subsequent steps, up to 500,000, highlight an indiscernible upsurge in Eve's error rates, which indicates that Eve can decipher Alice and Bob's communication in this experimental setup.

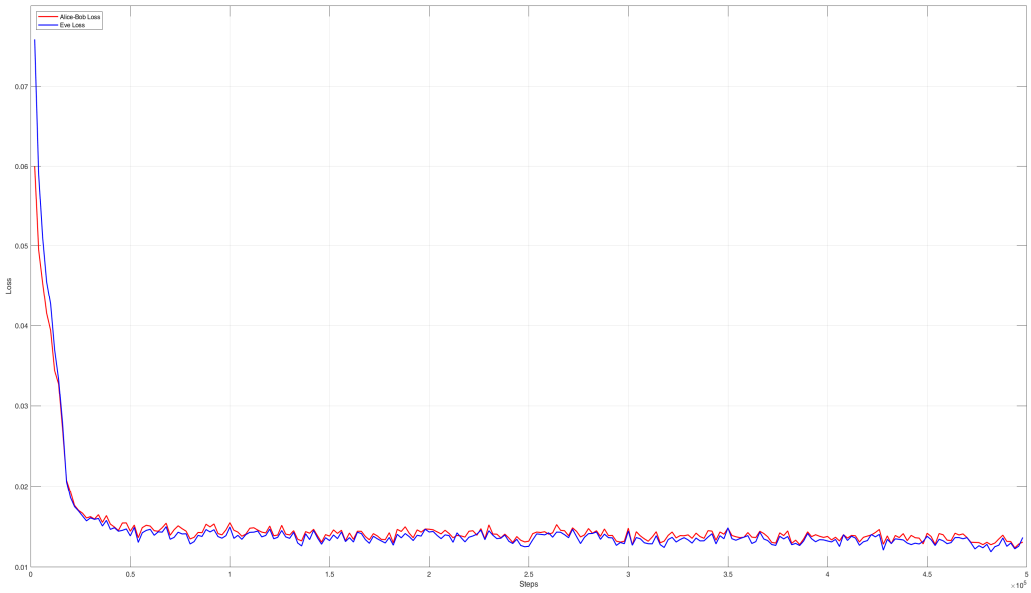

Figure 6. Reconstruction error of Bob and Eve during training during experimental setup 1.

Figure 7 shows that after examining the time metrics over the span of the initial 500,000 training steps, it is clear that the processing duration demonstrates fairly consistent behavior, hovering around a mean duration of approximately 19.42 s for each 2000-step increment.

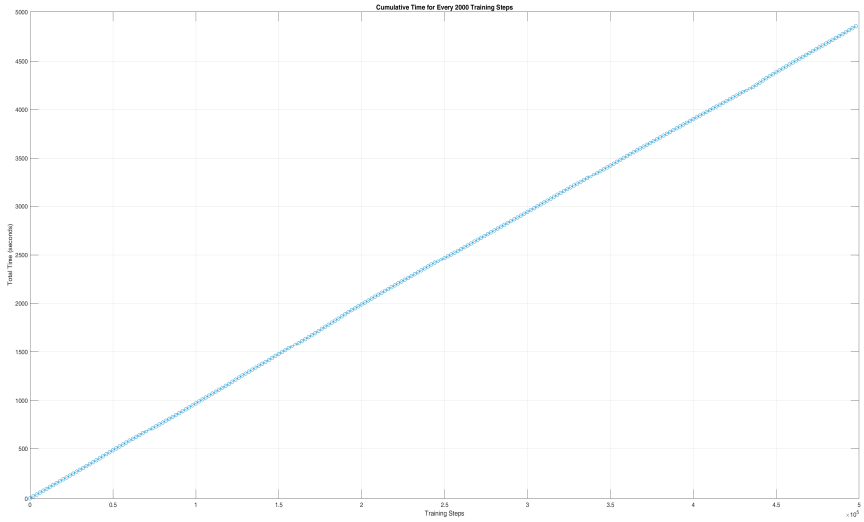

Figure 7. A line graph tracking the increasing cumulative time, in seconds, for every 2000 training steps up to 500,000, showcasing the escalating time commitment as training advances during experimental setup 1.

4.4. Experimental Setup 2

The second experimental setup utilizes "Mish" activation for the hidden layer and "sigmoid" activation for the output layer.

This model, designed for Alice and Bob, adopts a "concatenate and transform" paradigm [27]. The process starts by concatenating the plaintext message with the cryptographic key. This amalgamated input then flows through a fully connected (FC) layer encompassing 64 units. In this FC layer, the output can potentially be a sophisticated blend of the input bits. This design implicitly promotes interaction between the plaintext and the key, allowing the encryption process to naturally use the key's information.

Following this initial transformation, the data is subjected to the Mish activation function—a smooth, non-linear function derived from the combination of the tanh and softplus functions [28,29]. The hyperbolic tangent function, often abbreviated as tanh, is defined as in Equation (9):

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{9}$$

This function maps any real-valued number to the interval $[-1, 1]$. It is zero-cantered, which means that negative inputs will be mapped strongly negative and the zero inputs will be near zero in the output.

The derivative of softplus with respect to x is defined in Equation (10):

$$\frac{d}{dx} softplus(x) = \ln(1 + e^x) \tag{10}$$

which is essentially the sigmoid function.

Given the two functions defined above, the Mish activation function is defined in Equation (11):

$$mish(x) = x * \tanh(softplus(x)) \text{ or } mish(x) = x * \tanh(\ln(1 + e^x)) \tag{11}$$

This choice facilitates a nuanced activation mechanism, making the transformation more expressive and flexible. Post activation, the transformed data are directed to another FC layer, which returns an output of size equivalent to the original message length [30]. This output represents the encrypted (or decrypted, in Bob's case) message. The choice of the sigmoid activation function for this layer ensures the output values are constrained between 0 and 1, fitting the requirements of an encrypted message [31].

The architecture for Eve mirrors the blueprint used for Alice and Bob, with one crucial difference: it does not account for the cryptographic key [32]. Eve's task is to decipher the encrypted message without knowledge of this key. Consequently, her model starts directly with the encrypted message, then proceeds through a series of transformations akin to those in Alice and Bob's networks, sans the concatenation step.

Figure 8 demonstrates the outcomes of a particularly compelling run employing the Mish and sigmoid activation functions, exemplifying the sophistication of the model's architecture. The intensive training spans half a million steps, with milestones echoed every 2000 intervals till 500,000 steps. Throughout this rigorous loop, the system dynamically crafts random messages and keys, processes encryption, and engages both Bob and Eve in decryption, adeptly calculating losses and optimizing the models in response. Both the Alice–Bob and Eve loss values start relatively high, indicating that initially, the models are not performing well. As the steps increase, both losses seem to decrease.

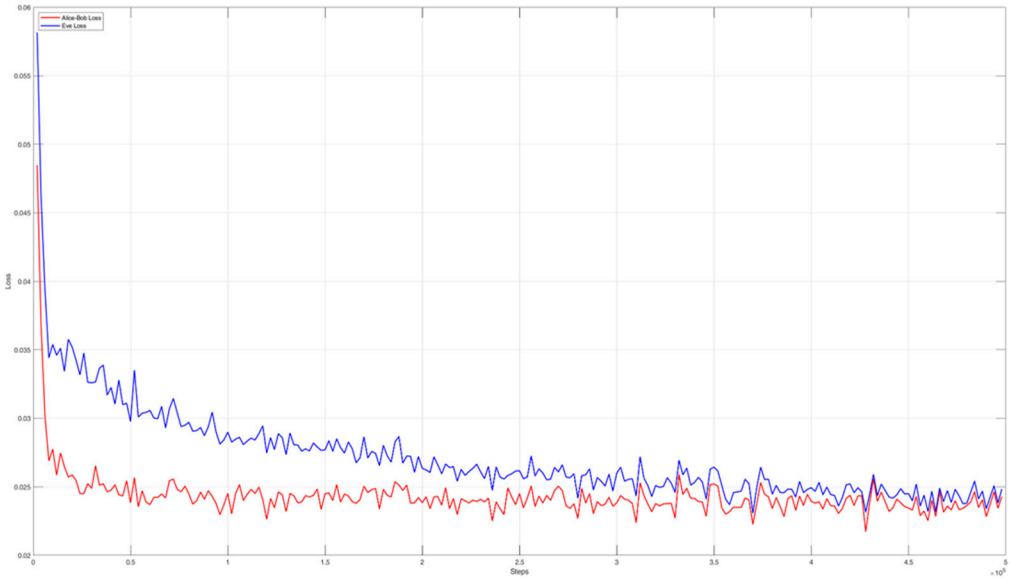

Figure 8. Reconstruction error of Bob and Eve during training during experimental setup 2.

Upon examining the time taken for various computational steps, ranging from 0 to 500,000 in increments of 2000, it is evident that after an initial rapid change, the time duration stabilizes around an average of approximately 20.52 s. The variance in time taken after this initial change is minimal, indicating a consistent computational efficiency across the steps. This consistency suggests that, within the scope of the experiment, the computational task scales linearly with an increase in the number of steps. The plotted graph provides a clear visual representation of this trend, further solidifying the observation of a consistent computational time.

Alice-Bob's loss starts at approximately 0.258 and steadily decreases to around 0.023 by step 126,000. Eve's loss starts at approximately 0.257 and decreases to around 0.027 by step 126,000. Alice–Bob's loss seems to oscillate around the range of 0.022 to 0.025. While it does vary, there is no drastic drop or increase, suggesting the model is stabilizing but not necessarily converging to a very low value. Eve's loss is similarly in the range of 0.024 to 0.027. Eve's eavesdropping capability, based on her loss, evolved from an initial undefined state at step 0, reaching a loss of 0.024 by step 500,000. This showcases Eve's consistent attempts to intercept the encrypted communication, with varying degrees of success. From the onset at step 0 through to step 500,000, Alice and Bob's encrypted communication system has seen consistent refinements, reflected in the steadiness of their loss metrics. Nevertheless, the narrow gap between Alice–Bob's loss and Eve's loss signals shows that, while there have been improvements, the encryption mechanism is not wholly resistant to potential eavesdropping. This emphasizes the necessity for continued advancements and

system enhancement to ensure optimal confidentiality. It appears that as the model trains, Eve is getting better with each increasing training step, Alice-Bob's loss is generally lower than Eve's, but the margin is very nominal at certain steps. This is likely undesirable in this adversarial setup, as we usually want Alice and Bob to communicate securely, making it harder for Eve to intercept and understand their communication.

In Figure 9, upon analyzing the time series data over incremental training steps, it is evident that as the system trains, the time values stabilize and hover around the 20 s mark after an initial anomaly. This suggests that with increased training steps, the system's performance reaches a consistent plateau in terms of execution time.

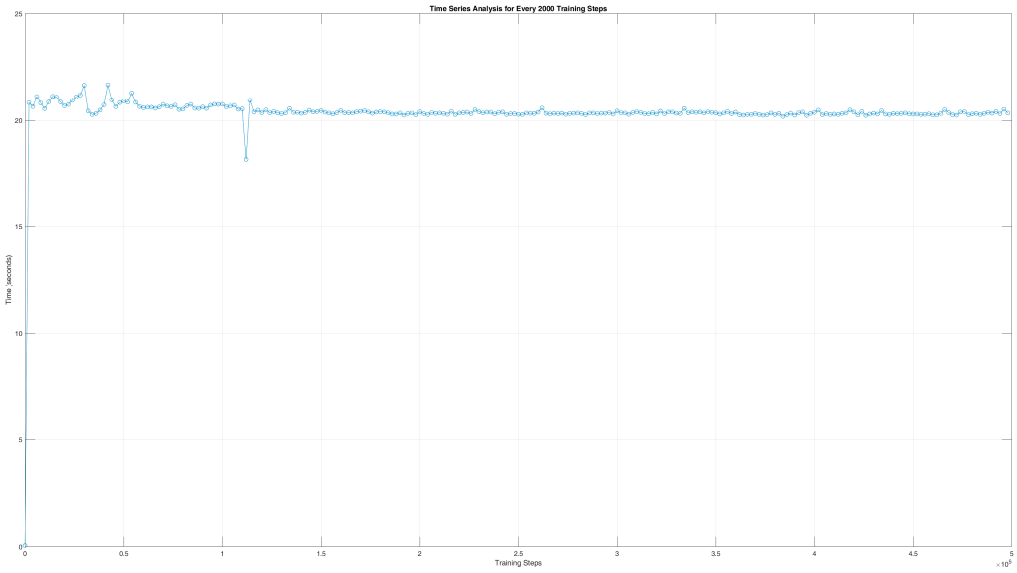

Figure 9. Time series analysis for training during experimental setup 2.

In Figure 10, upon delving into the time metrics of the initial 500,000 training steps, the data underscore a semi-stable pattern in the computation time. The values mainly fluctuate around a mean duration of roughly 20.35 s for every 2000 steps.

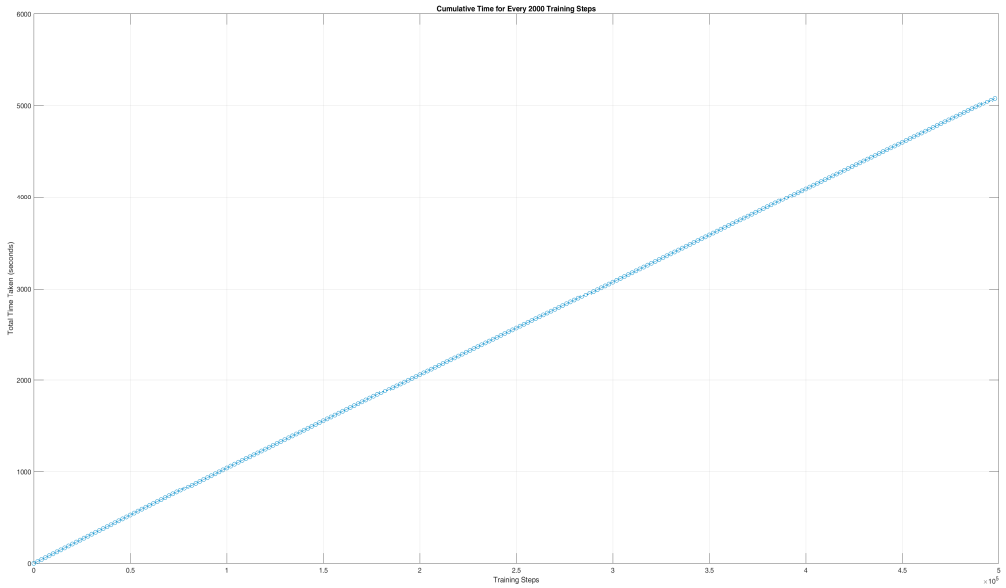

Figure 10. A line graph tracking the increasing cumulative time, in seconds, for every 2000 training steps up to 500,000, showcasing the escalating time commitment as training advances during experimental setup 2.

In essence, this architecture leans on neural networks to organically learn and adapt to the intricacies of encryption, decryption, and eavesdropping, instead of being bound by predetermined cryptographic conventions. The choice of the Mish activation function [33] in the hidden layers allows the model to tap into a richer representational space, enabling the network to discover intricate patterns and relationships within the data. The overarching philosophy is to let the networks learn the art of secure communication organically, rather than prescribing a predefined pathway.

4.5. Experimental Setup 3

The third experimental setup utilizes "ReLU" activation for the hidden layer and "sigmoid" activation for the output layer.

Both Alice and Bob employ a neural architecture that is defined by a "concatenate and transform" pattern. Initially, the plaintext message and the encryption key are concatenated. This merged information stream is then processed by a fully connected (FC) layer consisting of 64 units. Think of this layer as a complex combiner, allowing for potential interplay and integration between the message bits and the key bits.

The activation function [34] utilized post this FC layer is the "ReLU", mathematically represented as shown in Equation (12):

$$f(x) = \max(0, x) \tag{12}$$

The ReLU function introduces non-linearity, ensuring the encrypted output is not just a straightforward combination of input bits.

Following this is another FC layer tasked with outputting the encrypted or decrypted message. For this layer, the "sigmoid" activation function is harnessed. Mathematically, it is described as in Equation (13):

$$f(x) = \frac{1}{1 + e^{-x}} \tag{13}$$

The choice of the sigmoid ensures that the output, whether encrypted or decrypted, remains constrained between 0 and 1, making it ideal for binary message representation.

Eve's architecture mirrors that of Alice and Bob, with a notable exception: Eve lacks the key input. This deliberate omission means Eve must decrypt messages without the corresponding key, a challenge that pushes the neural network to its limits.

The entire system undergoes training for a staggering 500,000 iterations. Each training cycle involves generating random messages and keys [35]. Alice then encrypts these messages, and both Bob and Eve make attempts at decryption. Their performance is gauged through loss functions, which measure the discrepancy between the original messages and their decrypted counterparts. Based on these losses, the weights in all the networks are adjusted using the Adam optimization algorithm [36].

The benchmark for success in this experimental setup is dual-faceted: Alice and Bob should be able to communicate with minimal decryption error, ensuring the integrity and secrecy of their conversation. In stark contrast, Eve's decryption attempts should yield a higher error, indicating the robustness of the encrypted communication against eavesdropping.

Figure 11 shows that after an initial fluctuation with peaks reaching close to 19.77 s, the system's performance shows signs of stabilization within the 16 to 17 s range. As the system progresses further, there is a noticeable trend towards the lower end of the 16 s bracket, even dipping into the 15 s region. These data suggests that while there are sporadic spikes in execution time, the system predominantly operates with consistent efficiency.

From Figures 12 and 13, upon analyzing the time metrics from the initial 500,000 steps of the training process, it is evident that the computation time exhibits quasi-stable behavior, oscillating around for an approximately mean duration of 16.51 s for every 2000 steps.

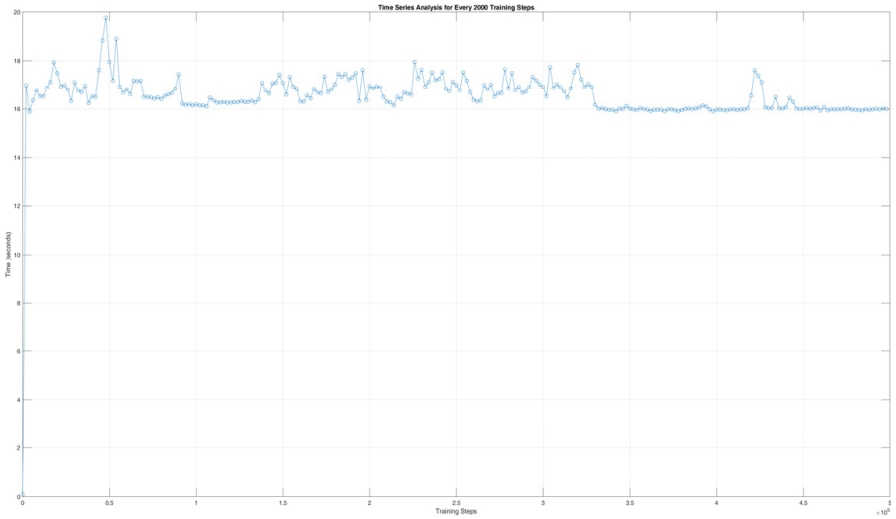

Figure 11. Time series analysis for training during experimental setup 3.

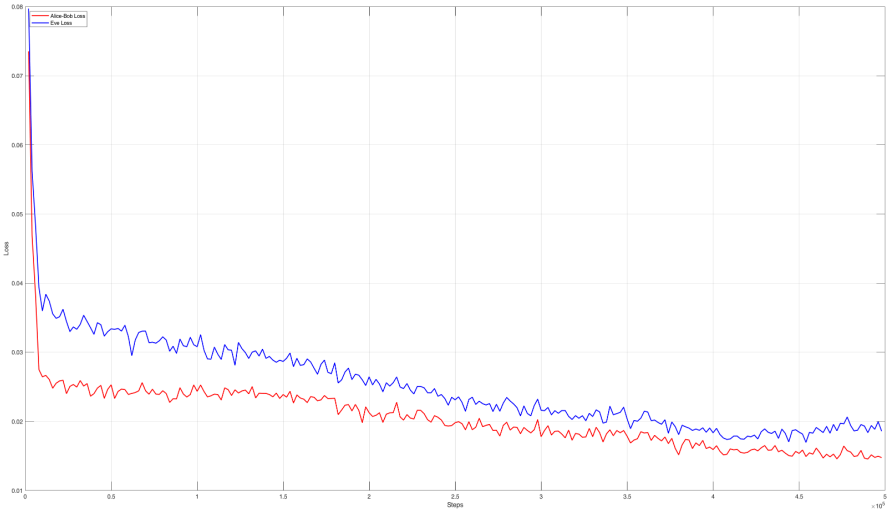

Figure 12. Reconstruction error of Bob and Eve during training during experimental setup 3.

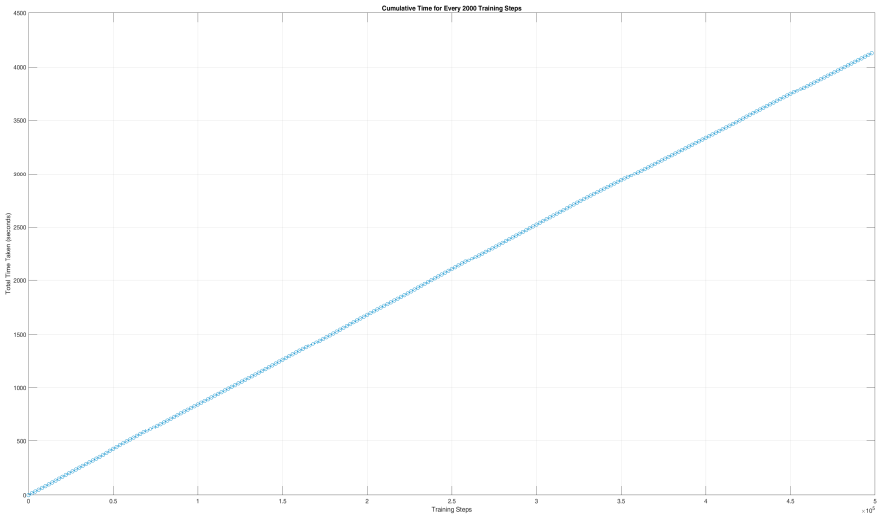

Figure 13. A line graph tracking the increasing cumulative time, in seconds, for every 2000 training steps up to 500,000, showcasing the escalating time commitment as training advances during experimental setup 3.

After training our adversarial neural cryptography models over 500,000 steps, we've observed a consistent and significant improvement in secure communication between Alice and Bob. The loss values, a measure of the accuracy of the encryption and decryption process, have shown a remarkable reduction from initial values of around 0.0735 to 0.0147 for Alice-Bob's communications. Conversely, Eve our eavesdropper, although improving her decryption attempts, has consistently lagged behind in her loss metrics, starting at 0.0797 and ending at 0.0186. This demonstrates the robustness and efficacy of our model, as the encrypted communications between Alice and Bob consistently prove challenging for Eve to decipher accurately.

5. Conclusions

In our cryptographic experiments, we rigorously evaluated the influence of various activation functions on the encryption strength and training efficiency. Our analysis revealed a clear hierarchy in performance. While the models employing "LeakyReLU and tanh" and the innovative "Mish and Sigmoid" delivered mixed results in terms of security and had a more extended training duration, the model utilizing the "ReLU and Sigmoid" combination emerged pre-eminent. With an impressively efficient mean training time of 16.51 s for every 2000 steps as compared to "LeakyReLU and tanh" (19.42 s) and "Mish and Sigmoid (20.35 s)", this outcome underscores the efficacy of the traditional ReLU and Sigmoid activation functions, reiterating their superiority in this cryptographic context. Drawing from our exhaustive experimentation, and the vivid distinction in the loss metrics, we discern a compelling narrative: while innovation and advancements propel us forward, it is the synergistic blend of established methodologies like the "ReLU and Sigmoid" pair that offers an unparalleled defense against adversarial breaches. This study not only highlights the potency of the "ReLU and Sigmoid" pair but also serves as a testament to the enduring relevance of established methodologies amidst rapid advancements.

However, our study is not without limitations. The scope of the activation functions tested was selective and may not encompass all possible combinations that could yield different results. These limitations underscore the need for further research, especially in exploring the integration of adaptive learning with dynamic activation selections, evaluating model robustness under environmental perturbations, and benchmarking against established cryptographic standards. In light of our findings, future endeavors should delve into the nuanced interplay of various activation functions, both within and across layers.

Author Contributions: Conceptualization, P.S., S.D. and P.P.; methodology, P.S.; software, P.S.; validation, P.S., S.D. and P.P.; formal analysis, P.S.; investigation, P.P.; resources, P.S.; data curation, P.S.; writing—original draft preparation, P.S.; writing—review and editing, P.S., S.D. and P.P.; visualization, P.S.; supervision, S.D. and P.P; All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The raw data supporting the conclusions of this article will be made available by the authors on request.

Conflicts of Interest: There is no conflict of interest between the authors.

References

1. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B. Generative adversarial nets. *Adv. Neural Inf. Process. Syst.* **2014**, *27*.
2. Abadi, M.; Andersen, D.G. Learning to protect communications with adversarial neural cryptography. *arXiv* **2016**, arXiv:1610.06918.
3. Zhang, H.; Xu, T.; Li, H.; Zhang, S.; Wang, X.; Huang, X.; Metaxas, D.N. Stackgan++: Realistic image synthesis with stacked generative adversarial networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2018**, *41*, 1947–1962. [CrossRef] [PubMed]

4. Briland, H.; Ateniese, G.; Perez-Cruz, F. Deep models under the GAN: Information leakage from collaborative deep learning. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, Dallas, TX, USA, 30 October–3 November 2017.
5. Júnior, J.V.C.; Gondim, J.J.C. Environment for Threat Intelligence Analysis and Generation using Open Sources. *J. Inf. Secur. Cryptogr.* **2019**, *6*, 9–14. [CrossRef]
6. Wei, Q.; Wang, H.; Zhang, G. A Robust Image Watermarking Approach Using Cycle Variational Autoencoder. *Secur. Commun. Netw.* **2020**, *2020*, 8869096. [CrossRef]
7. Atashpendar, A.; Grévisse, C.; Rothkugel, S. Enhanced Sketchnoting Through Semantic Integration of Learning Material. In *Applied Informatics, Proceedings of the Second International Conference, ICAI 2019, Madrid, Spain, 7–9 November 2019*; Florez, H., Leon, M., Diaz-Nafria, J., Belli, S., Eds.; Springer: Cham, Switzerland, 2019.
8. Papernot, N.; McDaniel, P.; Wu, X.; Jha, S.; Swami, A. Distillation as a Defense to Adversarial Perturbations Against Deep Neural Networks. In Proceedings of the 2016 IEEE Symposium on Security and Privacy (SP), San Jose, CA, USA, 22–26 May 2016; pp. 582–597. [CrossRef]
9. Nicholas, C.; Wagner, D. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (sp)*; IEEE: Piscataway, NJ, USA, 2017.
10. Shor, P.W. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM J. Comput.* **1997**, *26*, 1484–1509. [CrossRef]
11. Gentry, C. *A Fully Homomorphic Encryption Scheme*; Stanford University: Stanford, CA, USA, 2009.
12. Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System; Decentralized business review. 2008. Available online: https://assets.pubpub.org/d8wct41f/31611263538139.pdf (accessed on 31 October 2023).
13. Beaulieu-Jones, B.K.; Wu, Z.S.; Williams, C.; Lee, R.; Bhavnani, S.P.; Byrd, J.B.; Greene, C.S. Privacy-preserving generative deep neural networks support clinical data sharing. *Circ. Cardiovasc. Qual. Outcomes* **2019**, *12*, e005122. [CrossRef] [PubMed]
14. Xi, B. Adversarial machine learning for cybersecurity and computer vision: Current developments and challenges. *Wiley Interdiscip. Rev. Comput. Stat.* **2020**, *12*, e1511. [CrossRef]
15. Radford, A.; Metz, L.; Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv* **2015**, arXiv:1511.06434.
16. Salimans, T.; Goodfellow, I.; Zaremba, W.; Cheung, V.; Radford, A.; Chen, X. Improved techniques for training gans. *Adv. Neural Inf. Process. Syst.* **2016**, *2*.
17. Zhu, H.; Kaneko, T. Comparison of Loss Functions for Training of Deep Neural Networks in Shogi. In Proceedings of the 2018 Conference on Technologies and Applications of Artificial Intelligence (TAAI), Taichung, Taiwan, 30 November–2 December 2018; pp. 18–23. [CrossRef]
18. Zhou, Z.; Li, H.; Wen, S.; Zhang, C. Prediction Model for the DC Flashover Voltage of a Composite Insulator Based on a BP Neural Network. *Energies* **2023**, *16*, 984. [CrossRef]
19. Nhu, V.-H.; Hoa, P.V.; Melgar-García, L.; Tien Bui, D. Comparative Analysis of Deep Learning and Swarm-Optimized Random Forest for Groundwater Spring Potential Identification in Tropical Regions. *Remote Sens.* **2023**, *15*, 4761. [CrossRef]
20. Anshel, I.; Anshel, M.; Goldfeld, D. An algebraic method for public-key cryptography. *Math. Res. Lett.* **1999**, *6*, 287–291. [CrossRef]
21. Zhang, Y.; Lee, J.; Wainwright, M.; Jordan, M.I. On the learnability of fully-connected neural networks. In Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, Fort Lauderdale, FL, USA, 20–22 April 2017; Volume PMLR 54; pp. 83–91.
22. Lu, L.; Shin, Y.; Su, Y.; Karniadakis, G.E. Dying relu and initialization: Theory and numerical examples. *arXiv* **2019**, arXiv:1903.06733. [CrossRef]
23. Xu, B.; Huang, R.; Li, M. Revise saturated activation functions. *arXiv* **2016**, arXiv:1602.05980.
24. Arce, F.; Sossa, H.; Gomez-Flores, W.; Lira, L. Learning an Artificial Neural Network for Discovering Combinations of Bit-Quads to Compute the Euler Characteristic of a 2-D Binary Image. *Comput. Y Sist.* **2022**, *26*, 411–422.
25. Shi, J.; Chen, S.; Lu, Y.; Feng, Y.; Shi, R.; Yang, Y.; Li, J. An approach to cryptography based on continuous-variable quantum neural network. *Sci. Rep.* **2020**, *10*, 2107. [CrossRef]
26. Liu, S.; Song, G.; Huang, W. Real-time transportation prediction correction using reconstruction error in deep learning. *ACM Trans. Knowl. Discov. Data (TKDD)* **2020**, *14*, 17. [CrossRef]
27. Wu, Q.; Wang, F. Concatenate Convolutional Neural Networks for Non-Intrusive Load Monitoring across Complex Background. *Energies* **2019**, *12*, 1572. [CrossRef]
28. Li, H.; Li, J.; Guan, X.; Liang, B.; Lai, Y.; Luo, X. Research on overfitting of deep learning. In Proceedings of the 2019 15th International Conference on Computational Intelligence and Security (CIS), Macau, China, 13–16 December 2019; IEEE: Piscataway, NJ, USA, 2019.
29. Ertam, F.; Aydın, G. Data classification with deep learning using Tensorflow. In Proceedings of the 2017 International Conference on Computer Science and Engineering (UBMK), Antalya, Turkey, 5–8 October 2017; IEEE: Piscataway, NJ, USA, 2017.
30. Akher, F.B.; Shu, Y.; Varga, Z.; Bhaumik, S.; Truhlar, D.G. Post-Activation Function for Managing the Behavior of a Neural Network Potential with a Low-Dimensional Potential. *ChemRxiv* **2023**. [CrossRef]
31. Langer, S. Approximating smooth functions by deep neural networks with sigmoid activation function. *J. Multivar. Anal.* **2021**, *182*, 104696. [CrossRef]

32. Maghrebi, H.; Portigliatti, T.; Prouff, E. Breaking cryptographic implementations using deep learning techniques. In *Security, Privacy, and Applied Cryptography Engineering, Proceedings of the 6th International Conference, SPACE 2016, Hyderabad, India, 14–18 December 2016*; Proceedings 6; Springer International Publishing: Cham, Switzerlan, 2016.
33. Misra, D. Mish: A self-regularized non-monotonic activation function. *arXiv* **2019**, arXiv:1908.08681.
34. Agarap, A.F. Deep learning using rectified linear units (relu). *arXiv* **2018**, arXiv:1803.08375.
35. Acun, B.; Murphy, M.; Wang, X.; Nie, J.; Wu, C.-J.; Hazelwood, K. Understanding training efficiency of deep learning recommendation models at scale. In Proceedings of the 2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA), Seoul, Republic of Korea, 22 April 2021; IEEE: Piscataway, NJ, USA, 2021.
36. Byrd, J.; Lipton, Z. What is the effect of importance weighting in deep learning? In Proceedings of the International Conference on Machine Learning, PMLR, Long Beach, CA, USA, 9–15 June 2019.