Alternative Approaches to Steganography Using GANs

Aeden Connelly
Ritchie School of Engineering and Computer Science
University of Denver
Denver, CO, United States

***Abstract*—Steganography is a subfield of cryptography which involves the embedding of hidden messages within a different message or object, for instance, storing a text string in an image or an audio file. Cryptoprahy's principal goal is to encode a secret message, and to subsequently prevent that message from being decoded; similarly, this decoding should not lose any information in the process. Steganography, in an effort to circumvent normal cryptography, takes a step in a different direction and seeks instead to totally conceal the presence of the message encoded therein. Given this new framework, the goal of steganographic encryption also differs from cryptography. Like cryptography, it seeks to makes sure that in decoding, it loses as little information as possible, but it also aims to hide the fact that there is hidden information stored in the first place. Steganography has existed for thousands of years, and has recently seen a boom following the advent of machine learning technology. A particularly notable innovation in the field came from the adoption of *generative adversarial networks* in concealing steganographic messages.**

***Index Terms*—machine learning, steganography, generative adversarial networks**

I. Introduction

The primary basis for this project is the 2019 paper published by Zhang et al. titled *SteganoGAN: High Capacity Image Steganography with GANs*. In this project, the researchers were able to implement generative adversarial networks to achieve state of the art embedding rates for steganography which toppled previous approaches by having a near 10x increase in the ability to store data using steganography.

Due to the scope of this project, I do not aim to necessarily hit these benchmarks or revolutionize steganography in the same way that SteganoGAN did, but merely to take inspiration from the original and apply their thinking in a unique way. My project's deviation from the original does mean that a detailed cross-analysis between the two implementations is not entirely possible, however, my hope is that it has the potential to open doors for new modes of inquiry into the field of steganography.

Aside from subtle novelties in writing the code that makes up the foundation of this project to prevent a total parroting of the SteganoGAN code - for instance, I implemented a data loader which was not the one that the original implemented - this project differs from SteganoGAN in several ways:

- I trained and validated my model using a portion of a different dataset than SteganoGAN, known as IStego100k. This dataset is detailed in the paper *IStego100K: Large-scale Image Steganalysis Dataset* (Yang, et al., 2019)
- I changed the activation function from the LeakyReLU activation function found in the original (and several tangential works) to another one called Mish, which is detailed in the paper *Mish: A Self Regularized Non-Monotonic Activation Function* (Misra, 2020)
- While the original implemented and trained the data on a basic, dense, and residual encoder and decoder (which I implemented but did not conduct training on), it only implemented a basic critic. I implemented and tested not only a basic critic, but also a densely connected and residual one.

As I noted before, I do not aspire in this paper to revolutionize the study of steganography or even necessarily to be compared with prior approaches. My greatest hope in this paper is that it opens the door for further inquiries, either by myself or by others. It is also for this reason that I have implemented code which goes above and beyond my training, so that it may be taken further and a more nuanced analysis may take place at a later time.

The contents of this paper will be as follows. Section 2 is dedicated to a concise overview of the project and its main components. Section 3 reviews the related works on which I have based this paper and from which I have taken inspiration; this may seem counter intuitive, but I feel it is necessary given the similarity between the source material and my own. Section 4 goes on to explain the unique aspects of my own research methodology and how they deviate from the original: the datasets employed, the different activation functions explored, and the new approaches I introduce to the discriminator in the GAN framework. Section 5 discusses the loss functions that I used to train this network, as well as the metrics I took and the hyperparameters and other details of my experimentation. Subsequently, Section 6 examines my results. Section 7 concludes the paper by noting some of the potential ways forward from here.

II. Project

In standard, more conventional approaches to cryptography, it is known that the secret message has been encoded in some way, and the goal is then to decode it. Steganographic encoding is an alternative means of concealing these hidden messages: taking a step in a different direction, these encoders store their messages in objects of a different type so as to conceal their presence entirely. While there is nothing which outright prevents steganography from further encoding its message using normal cryptography, its main goal is to use this concealment method to make sure that the presence of its message is undetectable, and that no information is lost in translation. Early implementations of steganography can be traced back to ancient Greece, and have had a myriad of applications over time, such as invisible ink or music ciphers.

Generative adversarial networks - henceforth *GANs* - are a class of neural networks that operate through a dynamic interplay between two components: a generator and a discriminator." GANs consist of two main parts: a generator and a discriminator. The generator produces data, such as images, which is evaluated by a discriminator against real samples to judge its authenticity. This process continues in a sort of feedback loop where the discriminator is constantly evaluating - and ideally improving - the generator. The goal of the generator is to fool the discriminator by appearing entirely authentic. GANs have become a cornerstone in machine learning, particularly in areas like image generation, due to their ability to produce highly realistic data.

The primary challenge addressed by this project is enhancing the security and imperceptibility of hidden messages using advanced steganographic techniques combined with generative adversarial networks (GANs). Given the strength of GANs in image generation, this experiment seeks to leverage their capabilities to enhance the security and imperceptibility of steganographic techniques. The application of GANs in steganography is then to embed a secret message in some other kind of media. In this case, as in SteganoGAN, that consists of images: the GAN framework seeks to take a cover image C and embed in it a secret message M (in this case they are randomly-generated binary vectors), thereby producing a steganographically-encoded image S. The goal is to embed the message into the cover image in such a way that it is totally undetectable, but that when the message is decoded, no information is lost.

Specifically, this work explores the implementation of different critic architectures, including Basic, Dense, and Residual designs, within the GAN framework. By analyzing the performance of these architectures in terms of image quality, embedding efficiency, and robustness against detection, this research aims to contribute to the development of more sophisticated methods for securely embedding and transmitting hidden messages.

III. Related Works

A. SteganoGAN: High Capacity Image Steganography with GANs

This is by far the largest, most famous, and most impactful source of inspiration that I had in making this project. Until now, both traditional and machine learning approaches had not been able to make significant strides in steganography: traditional approaches, for instance, would often just try to embed images inside of other images rather than vectors, and machine learning approaches had imposed limitations on the size of the images that could be encoded. Additionally, machine learning methods had not done much to overcome traditional ones, and though their rates were competitive, they were by no means dominant. By introducing the use of a GAN framework, SteganoGAN was able to revolutionize the field of steganography, making images in which it was significantly harder to detect the presence of a secret message and where the secret message could be accurately decoded by the decoding agent. These figures were in the figure of a 10x improvement over the previous machine learning approaches. It is the essential structure of SteganoGAN which inspired this project as a whole, and while it is not a 1:1 copy, it is very close.

B. Implementation of SteganoGAN: High Capacity Image Steganography for Image and Audio Input with GANs

Following up on the approach of steganography with GANs, I found the paper above, by Garg and Rossetti, 2020. In it, they do something very similar to what I did here. This paper, as the name would suggest, also tested audio steganography, which I did not implement in my own project but which could potentially be deployed using similar methods to my own. The main idea I drew from this paper was to find a novel approach to implementing the original. Additionally, in this paper, the authors employed a more fragmented algorithm for the critic, which could allow for more modularity when modifying the project later.

C. Mish: A Self Regularized Non-Monotonic Activation Function

The paper above, published by Misra in 2020 details an activation function known as Mish. Mish is approximated as $x\tanh(softplus(x))$, and it was discovered as a means of improving upon the existing activation functions which have been dominant in the field of machine learning, particularly, ReLU, LeakyReLU, and Swish. Its invention was largely a means of addressing problems that were present in each of the functions, which we will discuss later. It outperforms all three on virtually all fronts, including in tests done on DenseNet and ResNet architectures, as well as tests in image recognition. The implementation of dense and residual connections on the encoder, decoder, and critic in my project made use of these libraries, and steganography, though not exactly the same as image recognition, does similar things, such as image feature detection.

IV. Methodology

A. Data

SteganoGAN and [2] both worked with the same datasets: The Diverse 2k Resolution Image Dataset (*DIV2K*), and the Microsoft Common Objects in Context dataset (*MSCOCO*).

These datasets are designed for general image analysis. The DIV2k dataset contains a smaller number of images which have a higher resolution. To be exact, the image resolution is 2k ($2048 * 1080$), and there are 800 images in the training set, 100 in the validation set, and 100 in the test set. MSCOCO is made up of lower resolution images, with $330,000$ total images: $140,000$ for training, $5,000$ for validation, and $40,000$ for testing.

These datasets have been around for a very long time - DIV2k was introduced in 2017 and MSCOCO in 2014 - and they have been used in a number of other studies for various kinds of tasks: object detection, image classification, and many more. The image detail of the DIV2k dataset make it good for making higher quality images, even if there are less of them, and the size and vastness of the MSCOCO dataset make it ideal for diversity, even if the images are not of the highest possible quality. Each one compensates where the other is weak, so these two complement each other very well.

The dataset I used is known as *IStego100k*, and is specifically designed for steganalysis. The dataset was introduced in 2019, more recently than either of the others. In total, it contains $208,104$ images. $200,000$ of these are reserved for training, split into one dataset made up of cover images and another made of steganographically-encoded ones, both of size $100,000$. The remaining $8,104$ are left for testing. Each of the images in IStego100k is $1024 * 1024$, so in terms of its level detail it lies somewhere between DIV2k and MSCOCO.

I chose to use this dataset for several reasons, mostly to stay consistent to the goals that I had in making this project in the first place: to remain different from the original and to open up new potential avenues for inquiry on this topic. The first of these should be fairly rhetorical and ubiquitous throughout my project at all levels. The second reason has to do with several other factors. For one, it is obviously advantageous to have a newer dataset which was designed specifically with steganalysis in mind. While the others may lend themselves to a more manifold set of uses, this one is tailored toward my body of research. Second, the dataset is more detailed than the MSCOCO dataset and larger than the DIV2k dataset, addressing the respective weaknesses in each at a happy medium, and providing a unique palette on which to test steganography. Thirdly, this dataset contains a set of already steganographically encoded images, so if one were to focus on any particular aspect of the project, they could do so by choosing the data they wanted to analyze - for instance, you could focus on the critic and the decoder by training on both the given cover images and the given stego images, thereby skipping the encoding phase in exchange for a more focused decoding and critique analysis.

For my research, I chose a very small sample of the IStego100k dataset and chose 800 cover images for training and 200 for validation. Since I was handling both encoding and decoding, which I will explain later, I did not need to use any of the pre-encoded images. While this smaller number does limit my testing and results, it also keeps open the possibility of using a much larger number of samples to evaluate the problem.

One key aspect to note about this dataset, and similar research, is the absence of a test dataset. Most machine learning models utilize a training set, a validation set, and a test set, as seen with the DIV2K and MSCOCO datasets. However, in the IStego100k dataset and my own testing (as well as in [2]), only two stages are present. This is because, in the context of Steganography with GANs, a test set is not strictly necessary. The primary objective is to accurately encode information rather than make predictions. In this sense, the validation phase effectively serves as the testing phase. The adversarial training process is designed to continuously challenge itself to avoid overfitting, which can allow the model to bypass the traditional validation stage.

B. Activation Function

Previous implementations of this framework have relied upon the LeakyReLU activation function, which is an expansion upon the regular ReLU (rectified linear unit) activation function, defined as $f(x) = max(0, x)$. The problem with the regular ReLU function is that it "dies" over time if the neural network continually outputs zero. This happens because once a neuron's output is zero, the gradient also becomes zero and the function does not update.

LeakyReLU's solution to the "dying ReLU" problem is to allow a small, non-zero gradient (typically 0.01) when the input is negative. This ensures that there is a consistent, albeit small, activation of the neurons at any given point, thereby reducing the likelihood that the activation signal "dies.". LeakyReLU is defined by the logic $f(x) = x$ if $f \geq 0$, else $f(x) = \alpha x$ if $x < 0$, where α represents the nonzero gradient. Although it is piecewise linear, the accumulation of these pieces introduces non-linearity by introducing different slopes when the system is provided different inputs. This offers significant improvements over ReLU. In spite of this, LeakyReLU is still monotonic, meaning that no matter what, it only increases in the positive direction. To introduce non-monotonic behavior, the Swish function was introduced. Swish is approximated as $f(x) = x * sigmoid(x)$. It makes use of the sigmoid function $\frac{1}{1+e^{-x}}$ to construct a non-linear, non-monotonic activation function which empirically outperforms both ReLU and LeakyReLU.

Mish, the activation function that I used in my project, is defined as $f(x) = \tanh(softplus(x))$, where $\tanh$ is the hyperbolic tangent function $\frac{e^x - e^{-x}}{e^x + e^{-x}}$ and $softplus$ is a smooth approximation of the ReLU activation function, defined as $\ln(1 + e^x)$. Both of these functions are non-linear and non-monotonic, and the result of multiplying them against each other is a function which permits even more nuanced behavior than Swish, and, as discovered in [3], it outperforms Swish, and thus also LeakyReLU and ReLU.

I employed use of the Mish activation function in my project to test what happens when a non-linear non-monotonous activation function. I decided upon Mish over Swish on the basis for a few reasons. The first is its improvement in performance over Swish. Additionally, it is new and novel enough to provide insights which may not have been researched before, and that there has not been much significant research into its applications. Additionally, I was intrigued by the benchmarks in [3], where Mish's improvement over swish was tested against Densenet and Resnet architectures, which are both employed in this project to create dense and residual architecture for encoding.

C. Critic Architecture

In SteganoGAN, the encoder's role is to embed the secret message into the cover image while minimizing the distortion of the cover image to ensure that the resulting steganographic image is visually indistinguishable from the original. SteganoGAN offers three types of encoders, which employ three different neural network architectures:

- **Basic:** Consists of a series convolutional layers followed by an activation function. Layers do not have additional complex behaviors, they simply pass the information along sequentially to focus on straightforward feature extraction. This is the simplest and most straightforward method of feature extraction.
- **Dense:** Based on the DenseNet architecture, a densely-connected network is one where every layer connects every layer to every other layer. The implication of this as it moves forward is that each layer receives input from all the other layers before it, allowing for more complex behavior at a greater computational cost.
- **Residual:** Based on the ResNet architecture, residual connections allow for the system to bypass one or more layers, potentially allowing for more complex relationships among existing layers. Allowing the bypassing of layers means that it can be easier to optimize. However, like dense encoding, this comes at a greater computational cost.

The original SteganoGAN project only implements a basic critic, despite implementing the encoder and decoder in these ways. In an adversarial network like this, it is crucial that the critic is able to keep up with the other agents in the network, the encoder and decoder. As such, I thought it would be interesting to try and implement the critic in the same way that the original implemented the encoder and decoder. My encoders, aside from the change in activation function from LeakyReLU to Mish, are identical to those in SteganoGAN. The only agent to which the changes were made was the critic.

I will discuss this further in the experimentation section, but, in my own research, I tested the various critics against only the basic encoder and decoder. While the other encoders were implemented, I wanted to keep the research concise and begin the examination into the topic, making for a more stable comparative analysis of the different critic architectures.

V. Experiment

In the GAN framework implemented for this project, the loss functions play a crucial role in guiding both the training and the validation of the generator (encoder-decoder network) and the discriminator (critic network). To remain consistent with the original, I implemented the same training and validation loss functions as SteganoGAN.

A. Training: Loss Functions

1) Encoder-Decoder Loss: The primary objective of the encoder-decoder network is to generate a stego image that is visually indistinguishable from the cover image while also accurately embedding the secret message. The loss function for the encoder-decoder is a combination of two key components:

- **Encoder MSE (Mean Squared Error):** This metrics measures the difference between the cover images and the stego images. The aim is to have a lower MSE, which indicates less distortion. Simply put, this means that it aims to make the stego image as similar as possible to the cover image, making the insertion of the message imperceptible.
- **Decoder Loss:** This metric evaluates the model's ability to accurately reconstruct the payload embedded by the encoder. It is optimized using cross-entropy loss, which quantifies the difference between the predicted and actual outputs of the decoder. The objective is to minimize the difference between the original message and the decoded message. It is complementary but not identical to the **decoder accuracy**, which directly measures the proportion of correctly decoded bits or data from the stego image. Decoder accuracy is measured by the system and is used in later metrics for validation.

2) Critic Loss:: This loss is calculated as the difference between the critic's scores for the cover and stego images. The objective is for the critic to output high scores for cover images and low scores for stego images, thus forcing the generator to create more realistic stego images that can "fool" the critic. It is calculated using the mean scores of the critic's evaluation.

B. Validation: Metrics

The validation stage of this experiment involved evaluating several key metrics to assess how well the model learned. These metrics were derived from both the loss functions used in training and additional metrics specifically relevant to steganography, which were the main benchmarks for SteganoGAN:

- **Structural Similarity (SSIM):** SSIM is used to measure the structural similarity between the cover and stego images. Higher SSIM values indicate that the stego image is more visually similar to the cover image. This metric is crucial because it correlates more with human perception than MSE or PSNR alone.
- **Peak Signal-to-Noise Ratio (PSNR):** This metric quantifies the reconstruction quality of the steganographic image in terms of the logarithmic ratio between the maximum possible power of a signal and the power of the noise. The higher the PSNR, the lower the distortion, which indicates that your method maintains image quality.
- **Reed-Solomon Bits Per Pixel (BPP):** This metric measures the payload capacity of the steganographic algorithm, indicating how many bits of information can be hidden per pixel while maintaining image quality.

These metrics provide a comprehensive evaluation of the model's performance, offering insights into both the visual quality and the effectiveness of the steganographic embedding process.

Additionally, the model also took as its validation metrics the aforementioned **decoder accuracy** as well as a *cover score* and a **generated score** which each evaluate the critic's ability to distinguish cover images from generated ones. A higher score means that the critic is better at evaluating whether or not the image is real. A higher cover score, for instance, indicates that the critic is improving at detecting real images.

C. Hyperparameters

For this project, I conducted my training on the aforementioned dataset of **800 images** and my validation on **200 images** randomly selected from IStego100k. Similarly to SteganoGAN, I used the Adam optimizer with a learning rate of **1e-4** to optimize the model.

Training was conducted over **16 epochs** with a **batch size of 16**. Initially, I intended to conduct the training over 32 epochs with larger batch sizes, but I encountered issues with memory allocation, likely due to the computational demands of the dense and residual critics. This meant that I had to reduce the size of my training accordingly.

D. Hardware

This experiment was conducted using Python notebooks in Google Colab. I utilized the A100 GPU, which is well-suited for image analysis and particularly effective for GAN applications due to its high memory capacity and processing power.

E. Equations

- **Encoder MSE:**

$$\frac{1}{n}\sum_{i=1}^{n}(G_i - C_i)^2$$

Where G is the generated image, C is the cover image, and n is the number of pixels.

- **Decoder Loss:**

$$-\left[P \cdot \log(D) + (1-P)\cdot \log(1-D)\right]$$

Where P is the payload and D is the decoded data.

- **Decoder Accuracy:**

$$\frac{\sum_{i=1}^{n}\mathbb{I}(\hat{P}_i = P_i)}{n}$$

Where $\hat{P}$ is the predicted payload, P is the actual payload, and n is the total number of bits.

- **Generated Score:**

$$\frac{1}{n}\sum_{i=1}^{n}\text{Critic}(G_i)$$

Where G is the generated image.

- **Cover Score:**

$$\frac{1}{n}\sum_{i=1}^{n}\text{Critic}(C_i)$$

Where C is the cover image.

- **SSIM:**

$$\text{SSIM}(C, G)$$

Where C is the cover image, and G is the generated image.

- **PSNR:**

$$10 \cdot \log_{10}\left(\frac{4}{\text{encoder\_mse}}\right)$$

Where 4 represents the maximum possible pixel value for normalized images (ranging from -1 to 1).

- **BPP:**

$$\text{bpp\_score} = \text{data\_depth} \cdot (2 \cdot \text{decoder\_acc} - 1)$$

Where data_depth is the depth of the data being embedded.

VI. Results

My results can be displayed in the following graphs:

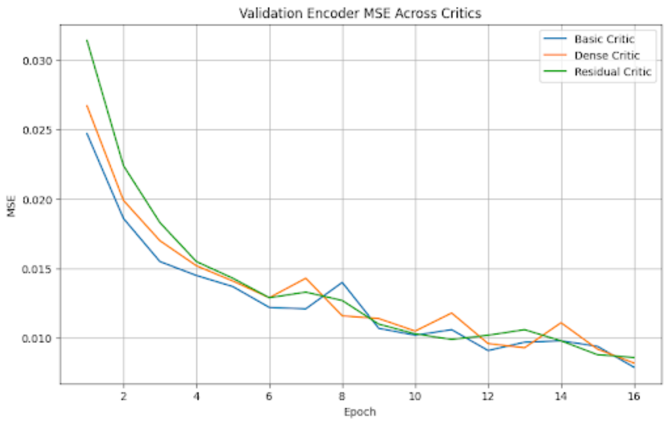

Fig. 1. The MSE decreases steadily across all critics as training progresses, with the Basic Critic showing a slightly lower MSE by the end

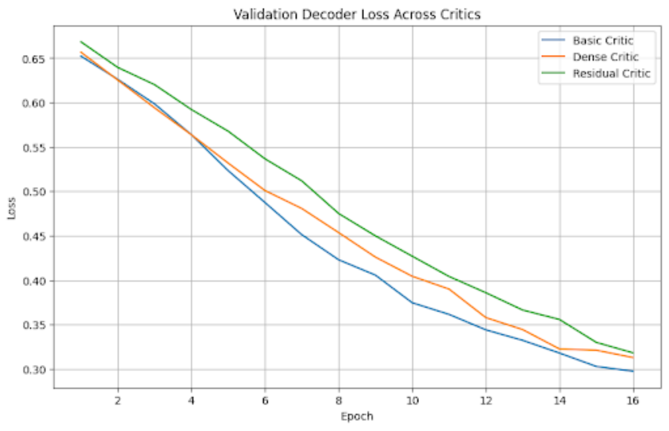

Fig. 2. The Decoder Loss decreases smoothly across all critics, with the Basic Critic achieving the lowest loss

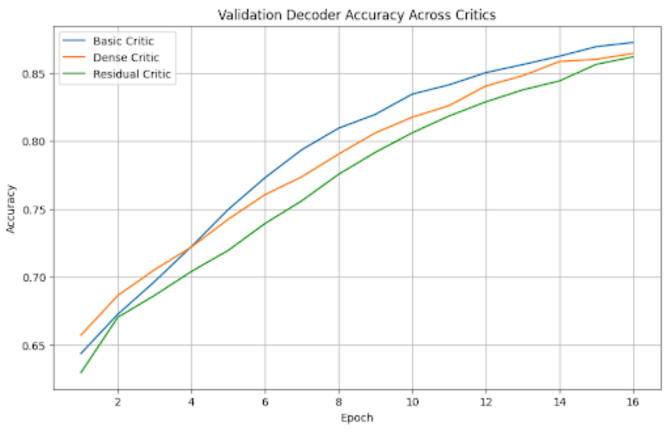

Fig. 3. Decoder Accuracy increases steadily across all critic types, with the Basic Critic slightly outperforming the others

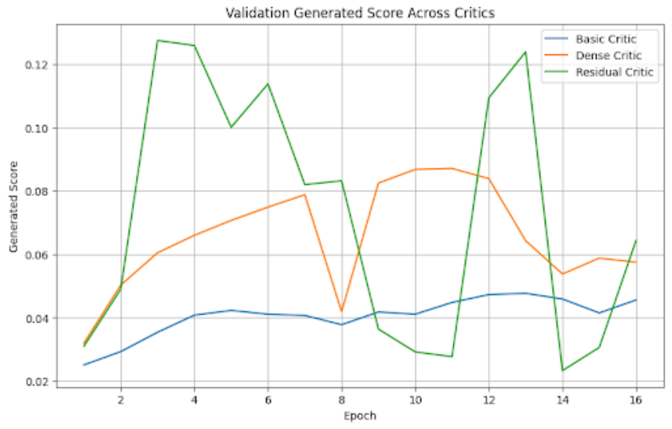

Fig. 4. There is more fluctuation in the scores for the Residual Critic, indicating that it might have a harder time consistently evaluating the quality of the generated images. The Dense Critic shows the most stable performance.

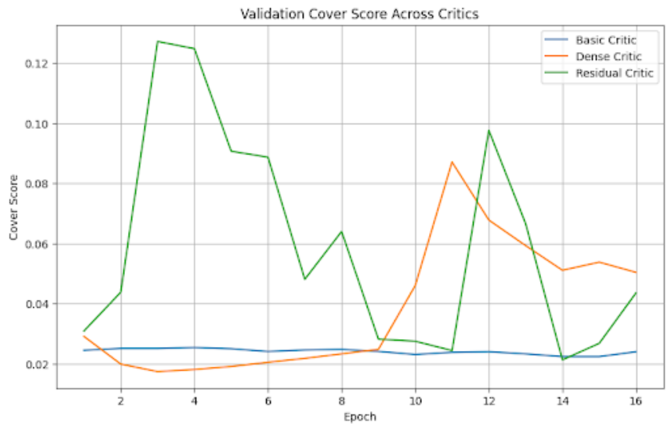

Fig. 5. The cover scores for the Residual Critic fluctuate more, suggesting it may struggle with consistency in evaluating the authenticity of cover images.

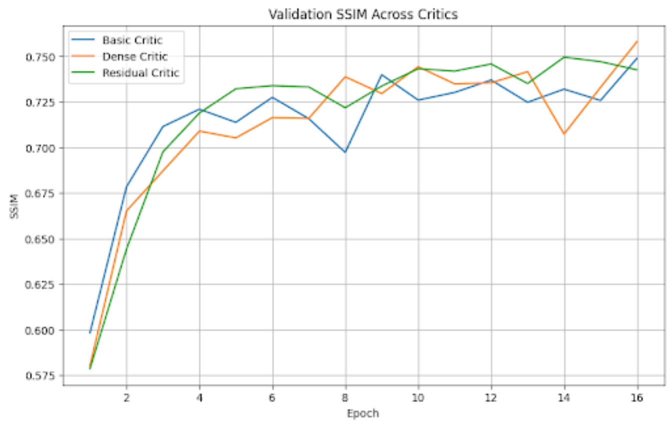

Fig. 6. The SSIM values show that all critics are capable of maintaining high visual similarity between the cover and stego images, with the Residual Critic slightly outperforming the others in the later stages of training

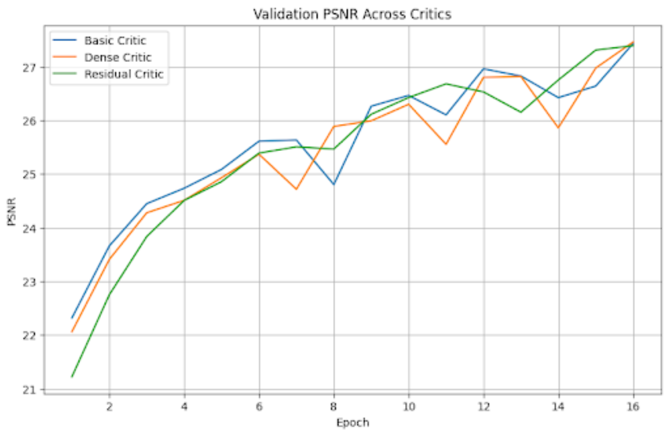

Fig. 7. The PSNR (Peak Signal-to-Noise Ratio) graph shows an overall improvement, with the dense critic performing slightly better.

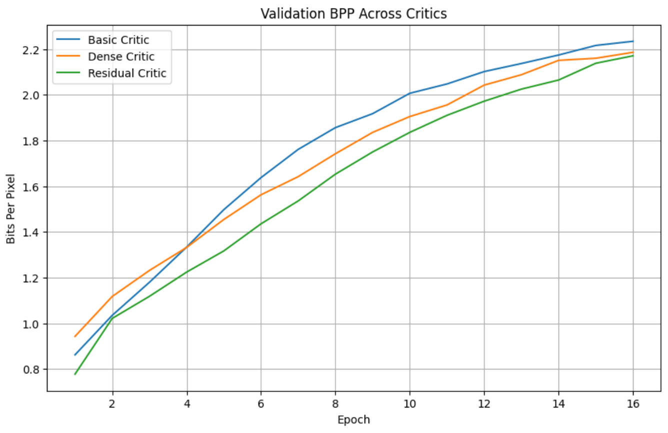

Fig. 8. The BPP (Bits Per Pixel) graph demonstrates a steady increase, with the basic critic achieving the highest BPP.

VII. Conclusion

The exploration of different critic architectures—basic, dense, and residual—has provided valuable insights into their impact on the performance of the steganographic GAN model. Among these, the basic critic consistently demonstrates superior performance in key metrics like decoder accuracy, PSNR, and BPP, positioning it as the most effective for the goals of this project. The residual critic, while more sensitive, often produced volatile results, particularly in the generated and cover scores. This increased sensitivity might be beneficial for tasks requiring detailed detection, though it presents challenges in maintaining stability during training.

For future experimentation, it would be beneficial to consider implementing significant changes to the hyperparameters. Utilizing larger batch sizes or extending the number of epochs could help in further stabilizing the model and refining its performance. Additionally, it may be advantageous to leverage the IStego100k dataset more extensively, either by selecting larger datasets or by modifying the data augmentation strategies to improve robustness and generalizability.

References

[1] K. A. Zhang, A. Cuesta-Infante, L. Xu, and K. Veeramachaneni, “SteganoGAN: High Capacity Image Steganography with GANs,” 2019, arXiv. doi: 10.48550/ARXIV.1901.03892.

[2] A. Garg, S. Rossetti, ”Implementation of SteganoGAN: High Capacity Image Steganography for Image and Audio Input with GANs,” 2020

[3] D. Misra, “Mish: A Self Regularized Non-Monotonic Activation Function,” 2019, arXiv. doi: 10.48550/ARXIV.1908.08681