McMaster University

Introduction to Neural Networks and ML

GANHide Computer Science Report

Om Patel

Supervisor: Erik S. Sørensen

A report submitted in partial fulfillment for the requirements of
McMaster University for the final project of
DATASCI 3ML3

April 24, 2025

Contents

List of Figures

Chapter 1

Introduction

1.1 Introduction

During my research, Generative Adversarial Networks (GANs) were found to be great for generating realistic data from images to text. GANs consist of a Generator and a Discriminator, where the Generator creates synthetic data while the Discriminator attempts to distinguish it from real data. This process helps produce highly favorable outputs.

The project, *GANHide*, will try to use GANs to create a steganography system. It will train a conditional GAN (cGAN) to embed short binary messages into grayscale images. This will help us achieve a balance between making the hidden messages imperceptible to both human and statistical detection methods, and ensuring that the messages can be reliably extracted. The system will use a Generator to produce stego-images, a Discriminator to ensure that these images are different from originals, and an Extractor to recover the embedded messages.

Finally, this report details the methodology, results, and analysis of *GANHide* by demonstrating its effectiveness through quantitative metrics and visual evidence. This project will highlight the potential of machine learning in solving real-world problems.

Chapter 2

Mathematical Framework for GAN

2.1 Conditional GAN (cGAN)

The first major part of the system is built on a conditional Generative Adversarial Network (cGAN). It consists of the following three parts:

- The **Generator** embeds a secret message into an original image, producing a stego-image.
- The **Discriminator** will then try to distinguish between the original and stego-images.
- The **Extractor** is designed to recover the hidden message from the stego-image.

2.1.1 Generator

The main goal of the generator is to take an original image x and a message m, and output a stego-image $\hat{x} = G(x, m)$ which still tries to visually resemble image x while encoding message m. G is a CNN that modifies x based on m.

2.1.2 Discriminator

Similar to it's name, the discriminator tries to determine whether an image is real (original) or fake (stego-image). D will output a probability $D(x) \in [0, 1]$, where 1 indicates that the image is "real" and 0 for "fake."

2.1.3 Extractor

As the name of this function, the extractor will simply extract the hidden message from a stego-image. E will be a CNN which maps $\hat{x}$ to a predicted message $\hat{m} = E(\hat{x})$. This will display the probabilities for each bit.

2.2 Loss Functions

The project's system is built using three loss functions, where each optimize a specific component.

2.2.1 Discriminator Loss

The formula for the discriminator loss is defined below:

$$\mathcal{L}_D = -\mathbb{E}_{x\sim p_{\text{data}}}[\log D(x)] - \mathbb{E}_{x\sim p_{\text{data}},\ m\sim p_m}[\log(1 - D(G(x, m)))] \tag{2.1}$$

To understand the formula in detail, read the following explanation:

- p_{data} is the distribution of real images from the training dataset.
- p_m is the distribution of the messages that need to be hidden.
- $x \sim p_{\text{data}}$: this means a real image sampled from the dataset.
- $m \sim p_m$: means a message sampled from the message distribution.
- $G(x, m)$: the generator function that embeds message m into image x, producing a stego-image.
- $D(\cdot)$: this is the discriminator function which outputs the probability that the input is a non-stego image.

If we look at the formula, originally, the discriminator's goal is to **maximize** the following objective:

$$\text{maximize} : \mathbb{E}_{x\sim p_{\text{data}}}[\log D(x)] + \mathbb{E}_{x\sim p_{\text{data}},\ m\sim p_m}[\log(1 - D(G(x, m)))] \tag{2.2}$$

However, since most of the optimization problems are made to **minimize** a loss function, I had to multiply the objective by -1 to convert it into a minimization problem. This lead me to the formula:

$$\mathcal{L}_D = -\mathbb{E}_{x\sim p_{\text{data}}}[\log D(x)] - \mathbb{E}_{x\sim p_{\text{data}},\ m\sim p_m}[\log(1 - D(G(x, m)))] \tag{2.3}$$

2.2.2 Generator Loss

$$\mathcal{L}_G = -\mathbb{E}_{x\sim p_{\text{data}},\ m\sim p_m}[\log D(G(x, m))] + \lambda \cdot \mathcal{L}_{\text{extraction}} \tag{2.4}$$

$$\mathcal{L}_{\text{extraction}} = \mathbb{E}_{x\sim p_{\text{data}},\ m\sim p_m}[\text{BCE}(m, E(G(x, m)))] \tag{2.5}$$

To understand the formula, please view the explanation below:

- The adversarial loss (2.4) shows that generator G produces stego-images that the discriminator D classifies as real, $D(G(x, m)) \approx 1$.
- The extraction loss (2.5) is used to ensure that the extractor E can accurately extract the original message m from the stego-image $\hat{x} = G(x, m)$. It uses binary cross-entropy (BCE) loss.
- The formula is seen to also include a scalar λ. It is used to balance the trade-off between adversarial and message.

2.2.3 Extractor Loss

$$\mathcal{L}_E = \mathbb{E}_{x\sim p_{\text{data}},\ m\sim p_m}[\text{BCE}(m, E(G(x, m)))] \tag{2.6}$$

The loss of formula (2.6) minimizes BCE between the true message m and the predicted message $\hat{m} = E(G(x, m))$. This will optimize the extractor E, which will then be able to accurately recover the hidden message from the stego-image.

2.3 Bit-Wise Accuracy

$$\text{Accuracy} = \frac{1}{N \cdot 8} \sum_{i=1}^{N} \sum_{j=1}^{8} \mathbb{I}(m_{i,j} = \hat{m}_{i,j}) \tag{2.7}$$

Taking the formula above, we can see:

- N is the batch size, and $m_{i,j}$ is the j-th bit of the i-th message.
- $\mathbb{I}$ is the indicator function, which returns 1 if $m_{i,j} = \hat{m}_{i,j}$, and 0 otherwise.
- This formula helps measure the extractor's average bit-level accuracy across the batch.

2.4 Summary

Overall, we can see that the Generator embeds messages into images by balancing two goals: maintaining visual realism and ensuring message recover accuracy. The Discriminator is used to detect stego-images but when running the code, it is seen to stabilize around 50% accuracy. This indicates that the Generator is successfully fooling it. Meanwhile, the Extractor accurately recovers hidden messages, via the loss function. If we put all of this together, these components helps ensure both secrecy and reliability in our system.

Chapter 3

Training Progress & Analysis

```
Epoch 1/20
D Loss: 0.6943, D Acc: 0.5042
G Loss: 1.3347
E Loss: 0.6939, E Acc: 0.4956
Val E Acc: 0.4956

Epoch 2/20
D Loss: 0.6985, D Acc: 0.4954
G Loss: 1.3194
E Loss: 0.6932, E Acc: 0.5000
Val E Acc: 0.5081

.
.
.

Epoch 19/20
D Loss: 0.7471, D Acc: 0.4939
G Loss: 0.6205
E Loss: 0.0833, E Acc: 0.9850
Val E Acc: 0.9857

Epoch 20/20
D Loss: 0.7478, D Acc: 0.4939
G Loss: 0.6090
E Loss: 0.0749, E Acc: 0.9863
Val E Acc: 0.9901
```

From the small snippet above, we can see that the model was trained for 20 epochs, with key metrics for each epoch: Discriminator loss and accuracy (D Loss, D Acc), Generator loss (G Loss), and Extractor loss and accuracy (E Loss, E Acc), and validation accuracy (Val E Acc).

3.1 Discriminator Behavior

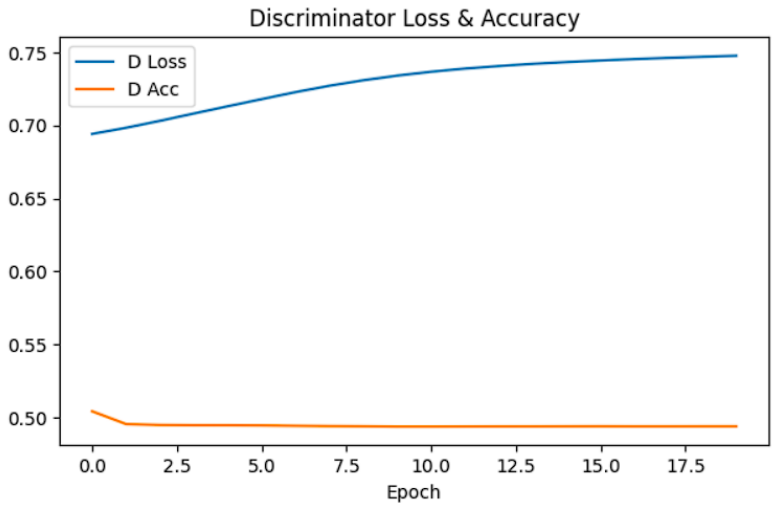

Figure 3.1: Discriminator Metrics

From the figure above, we can see that the Discriminator's accuracy hovered around 49.4% throughout training, and its loss increased slightly. If we look at Epoch 1 D Acc: 50.4% $\rightarrow$ Epoch 20 D Acc: 49.4%, we see that the Discriminator struggled to distinguish between real and stego images. This leads to the conclusion that it performs on a same level as random guessing. This is actually a good result for the GAN, as it implies that the Generator is successfully fooling the Discriminator by generating highly realistic stego-images.

3.2 Generator Progress

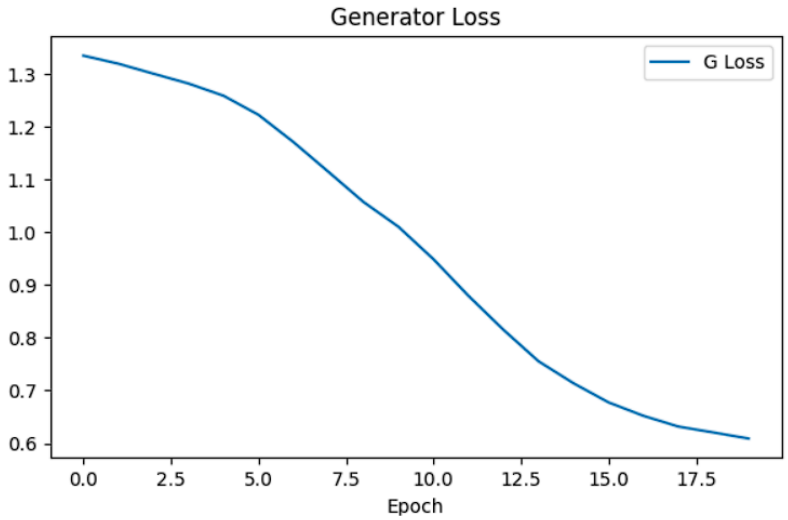

Figure 3.2: Generator Metrics

We see from figure 3.2 that the Generator loss steadily decreases, indicating improved performance in creating both realistic images and recoverable messages. This downward trend

suggests that the Generator is becoming more effective at minimizing both adversarial loss and extraction loss.

3.3 Extractor Behavior

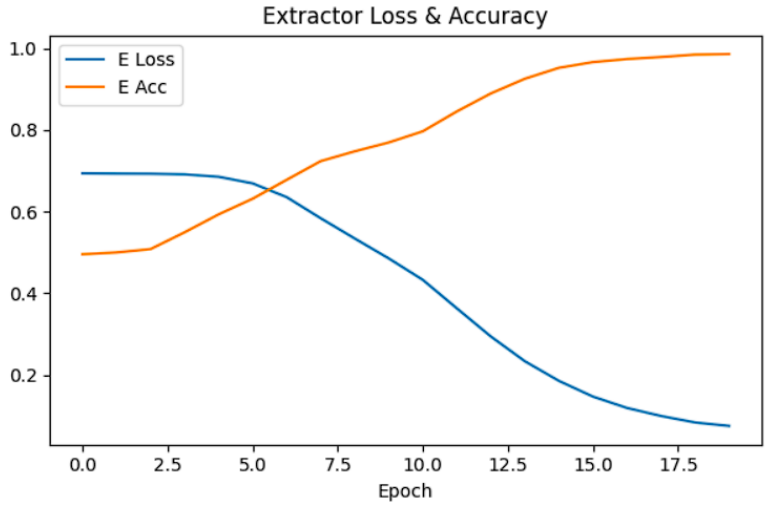

Figure 3.3: Extractor Metrics

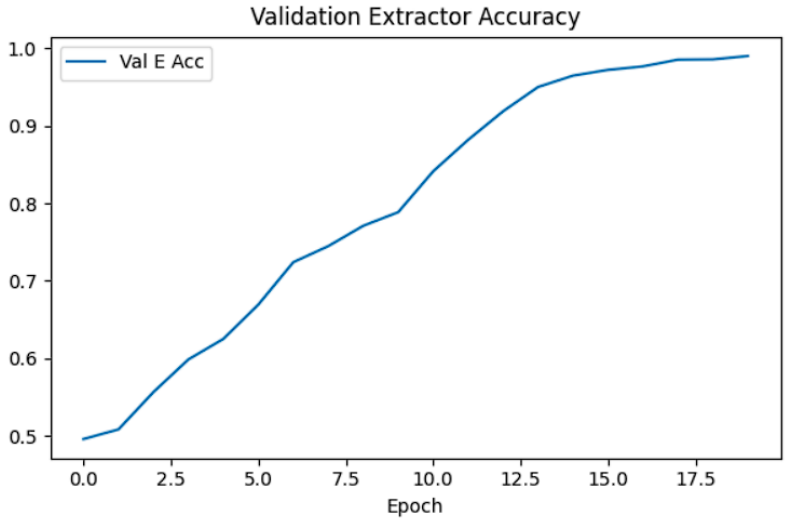

Figure 3.4: Extractor Validation Accuracy

We see that the Extractor improved significantly, where E Loss decreased from 0.6939 to 0.0749 and E Acc increased from ∼49.5% to ∼98.6%. We can also see that the validation accuracy improved from ∼49.6% to ∼99.0%. This improvement in validation accuracy solidifies our model by confirming strong generalization.

3.4 Summary

Overall, we can see that the Generator successfully learned how to produce realistic stego-images that hide messages effectively. We see that the Extractor became highly accurate in recovering messages. The Discriminator was effectively neutralized, helping validate the adversarial progress of the Generator. Laslty, the consistent rise in Val E Acc shows generalization.

Chapter 4

Comparison of Original & Stego Images

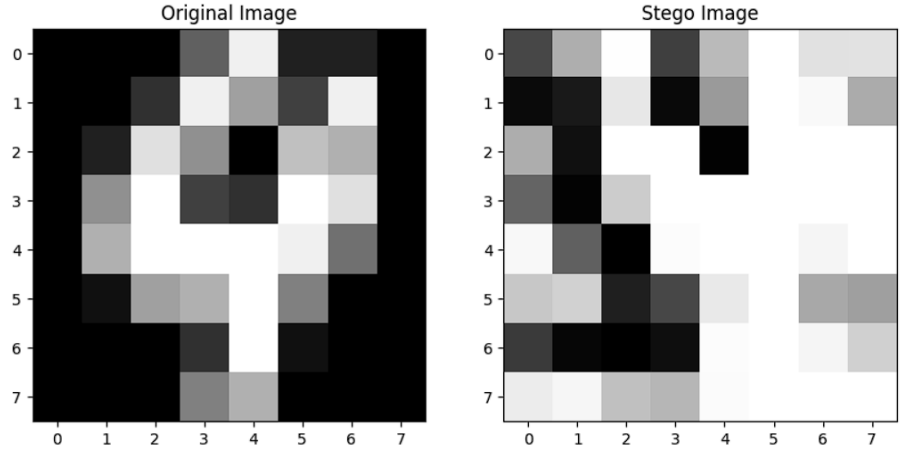

Figure 4.1: Side-by-side comparison of an original image and its stego-image

Taking Figure 4.1 into consideration, we see that it shows a side-by-side comparison of an original image and its corresponding stego image. Although the stego image retains the general structure of the original image, there are clear distortions, particularly in areas where the generator has embedded the secret message. Despite the discrepancy, the embedded image remains interpretable, demonstrating that the model maintains a balance between utility and secrecy.

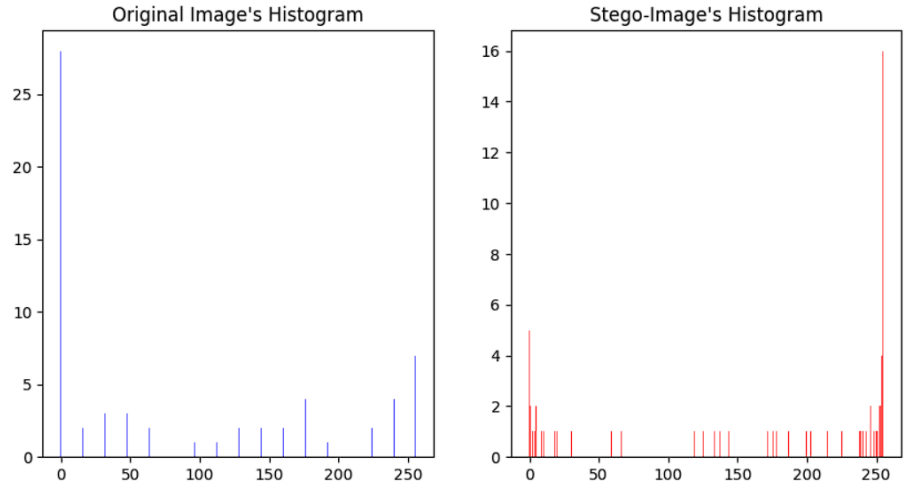

Figure 4.2: Histograms for the original and stego images

To further analyze the impact of the embedding, Figure 4.2 shows histograms of pixel intensity values from both the original and stego images. The original image can be seen to show concentration of pixel values near 0 and a few higher peaks scattered across the intensity range. In contrast, the stego image shows a redistribution of these values, particularly around the ends of the intensity range. This suggests that the embedding process significantly modifies the pixel values to encode the message, even though it tries to remain close to the original distribution.

This difference in the histogram indicate that while the generator is effective at concealing information visually, the embedding process introduces changes at the statistical level. This can somewhat be exploited by techniques to extract the embedded message.

Lastly, the following output from the testing loop confirms that the embedded messages are being successfully extracted:

```
Test 1:
Original Message: 0 1 1 1 1 1 0 0
Extracted Message: 0 1 1 1 1 1 0 0

Test 2:
Original Message: 1 0 1 0 0 1 0 1
Extracted Message: 1 0 1 0 0 1 0 1

Test 3:
Original Message: 0 1 1 1 0 1 1 1
Extracted Message: 0 1 1 1 0 1 1 1

Test 4:
Original Message: 0 1 0 1 1 1 0 0
Extracted Message: 0 1 0 1 1 1 0 0

Test 5:
Original Message: 1 1 1 0 1 0 1 1
Extracted Message: 1 1 1 0 1 0 1 1
```

Chapter 5

Conclusions and Further Takeaways

5.1 Conclusions

This project demonstrated a deep learning-based approach for image steganography using encoder-decoder architectures. The goal of this project was to embed binary messages into images while preserving visual quality and ensuring reliable message extraction.

The results show that the system successfully encodes and decodes messages with high accuracy. Visual inspection revealed that while the stego images remain recognizable, they contain subtle artifacts, mainly due to the embedding process. This drawback does not severely impact the perceptual quality, as evident by the similarity between original and stego images.

A detailed histogram analysis helped me confirm that the pixel intensity distribution shifts noticeably after embedding, especially at the extremes. This statistical discrepancy highlights the importance of developing more advanced techniques to better mimic the original distribution for stronger steganographic security.

Lastly, the consistent extraction of messages across various test cases confirms the reliability of the decoder. The model successfully generalizes across different samples, which suggests that the embedding process is robust and effective.

References

- Mahzaib Khalid, *Understanding Loss Functions in GANs: GAN Training and Impact on Results*. Available: Link
- Or Shulrufer, *Enhancing GANs Training Through Correct Loss Function Formulation*. Available: Link
- Encord Blog, *An Introduction to Cross-Entropy Loss Functions*. Available: Link
- Jeremy Watt, Reza Borhani, and Aggelos K. Katsaggelos, *Machine Learning Refined: Foundations, Algorithms, and Applications*. Cambridge University Press, 2016.