1 2 9 0

UNIVERSIDADE Đ
COIMBRA

Telmo Dias Cunha

Noise simulation for the improvement of training deep neural networks for printer-proof steganography

Dissertação no âmbito do Mestrado em Engenharia Eletrotécnica e de Computadores, no ramo de Computadores orientada pelo Professor Doutor Nuno Miguel Mendonça da Silva Gonçalves e apresentada no Departamento de Engenharia Eletrotécnica e de Computadores da Faculdade de Ciências e Tecnologia da Universidade de Coimbra.

Setembro de 2023

FCTUC FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Noise simulation for the improvement of training deep neural networks for printer-proof steganography

Telmo Dias Cunha

Coimbra, Setember 2023

Noise simulation for the improvement of training deep neural networks for printer-proof steganography

Supervisor:
Prof. Nuno Gonçalves

Co-Supervisor:
Prof. Luiz Schirmer

Jury:
Prof. Teresa Gomes
Prof. Jorge Lobo
Prof. Nuno Gonçalves

Dissertation submitted in partial fulfillment for the degree of Master of Science in Electrical and Computer Engineering.

Coimbra, Setember 2023

Acknowledgements

I want to express my sincere gratitude to my dissertation supervisor, Professor Ph.D. Nuno Gonçalves, and co-supervisor, Ph.D. Luiz Schirmer, for their unwavering support and guidance throughout research and development.

I also would like to thank my colleagues and friends who provided support and encouragement throughout the research process, as well as appreciate the help provided by my colleagues from the VIS Team of Institute of Systems and Robotics (University of Coimbra) for the useful discussions of the ideas and constructive criticism of the proposed methods.

Finally, I am grateful to my family and loved ones, who have been a constant source of motivation and inspiration during my graduate studies. Thank you to everyone who has contributed to my personal and academic growth during this journey.

Thank you,

Telmo

Resumo

Nos dias de hoje, imagens surgiram como ferramentas valiosas para ocultar informações, dando a origem a métodos inovadores como marca de água e a esteganografia, no qual nos últimos anos surgiram soluções de esteganografia de ponta a ponta. No entanto, estes novos métodos apresentam alguns problemas relativamente à mensagem oculta e à diminuição da qualidade das imagens. Além disso, a maioria dos métodos de esteganografia e marca de água são aplicáveis apenas a imagens digitais, uma vez que a impressão ou qualquer outra transformação física destruirá os dados ocultos. Este projeto de dissertação investiga a eficácia dos métodos de simulação de ruído e dos métodos de aprendizagem profunda para melhorar a resistência da esteganografia à impressão. A investigação desenvolve uma solução de esteganografia à prova de impressão de ponta a ponta, com um foco particular no desenvolvimento de um módulo de simulação de ruído, capaz de superar as distorções causadas pela transmissão do meio impressão e digitalização. Durante o desenvolvimento, são implementadas várias abordagens, desde a combinação de várias fontes de ruído presentes no ambiente físico, durante o processo de impressão e captura de imagem, como também a introdução de técnicas de Data Augmentation e Self-Supervising Learning com o intuito de melhorar e estabilizar a resistência da rede neuronal da solução de esteganografia. Através de uma experimentação rigorosa, obteve-se um aumento significativo da robustez da rede neuronal ao adicionar combinações de ruído, enquanto se manteve o desempenho da rede neuronal. Assim, através destas experiências, provou-se que a simulação de ruído pode fornecer um método robusto e eficiente para melhorar a esteganografia à prova de impressão.

Palavras Chave: Esteganografia resistente à impressão, Simulação de Ruído; Aprendizagem Profunda; GAN.

Abstract

In the modern era, images have emerged as powerful tools for concealing information, giving rise to innovative methods like watermarking and steganography, with end-to-end steganography solutions emerging in recent years. However, these new methods presented some issues regarding the hidden message and the decreased quality of images. Additionally, the majority steganography and watermarking methods are only applicable to digital images, since printing or any other physical transformation will destroy the hidden data. This dissertation project investigates the efficacy of noise simulation methods and deep learning methods to improve the resistance of steganography to printing. The research develops an end-to-end printer-proof steganography solution, with a particular focus on the development of a noise simulation module capable of overcoming distortions caused by the transmission of the print-scan medium. Through the development, several approaches are employed, from combining several sources of noise present in the physical environment during printing and capture by image sensors to the introduction of data augmentation techniques and Self-Supervising learning to improve and stabilize the resistance of the steganography solution neural network. Through rigorous experimentation, a significant increase in the robustness of the neural network was obtained by adding noise combinations while maintaining the performance of the neural network. Thereby, through these experiments, it was proven that noise simulation can provide a robust and efficient method to improve printer-proof steganography.

Keywords: Printer-proof steganography; Noise Simulation, Deep Learning; GAN.

"We can know only that we know nothing. And that is the highest degree of human wisdom"

— Lev Tolstói, *War and Peace*

Contents

List of Acronyms

CGANs	Conditional Generative Adversarial Networks
CNN	Convolutional Neural Network
DCT	Discrete Cosine Transform
DWT	Discrete Wavelet Transform
DNN	Deep Neural Network
GANs	Generative Adversarial Networks
GELU	Gaussian Error Linear Unit
IDs	Identity Documents
LPIPS	Learned Perceptual Image Patch Similarity
LSB	Least Significant Bit
ML	Machine Learning
MRTD	Machine Readable Travel Documents
MSE	Mean Square Error
NST	Neural Style Transfer
PSNR	Peak Signal-to-Noise Ratio
ReLU	Rectified Linear Unit
STN	Spatial Transformer Network
SSIM	Structural Similarity Index
SSL	Self-Supervising Learning

List of Figures

List of Tables

1 Introduction

Nowadays, the presence of images across diverse applications has transformed the way we communicate, learn, and interact with the digital world. From social media platforms to scientific research endeavors, images have emerged as potent conveyors of information and knowledge. One notable use of images is their innate capacity to carry concealed information, an interesting characteristic for both researchers and industries. This concept has led to the development of several applications aimed at increasing the utility of images across various aspects of society, including security measures. Two prominent techniques that leverage images for information embedding are watermarking and steganography, which represent essential contributions to the dynamic field of image manipulation.

Watermarking is a technique used to embed visible or invisible information, called a watermark, into digital media. On the other hand, steganography consists of hiding information in digital media, like audio, image, or video, while guaranteeing a low distinctiveness of encoded and cover samples (original). In recent years, researchers have developed complex algorithms based on deep learning networks (DNN) and steganography, which are dominated by end-to-end steganography solutions. This approach has the main concept of using Convolutional Neural Networks (CNNs) or other complex networks, such as GANs [9, 10, 22], to embed hidden messages into images and retrieve those concealed messages. The concept of GANs, which is composed of generator and discriminator models, is to use the generator learning process to produce a desired output while the discriminator learns to distinguish true data from the output of the generator. These new processes allow for a more robust concealed message and also improve the quality of the images with the concealed message.

Another important aspect to be mentioned about these new techniques is the use of noise simulation, which is one of the aspects that can influence the performance of detecting the hidden message. Noise simulation consists of recreating the different types of noise present in the surrounding environment, in this case during the process of printing and scanning the hidden information, to optimize the resistance of the architectures under the influence of noise as well as to assess the effectiveness of the steganography process.

In this dissertation study, the work developed consists of three aspects: noise simulation, deep learning networks, and steganography. Noise simulation allows for improvements in the resistance and reliability of the architectures. DNNs enhance the steganography process by creating more realistic images that are more resistant to steganalysis systems. By combining these three methods, it is possible to provide a robust and effective way to improve end-to-end steganography solutions.

1.1 Motivation

In the modern era, the addition of information to an image brings several benefits to society since it can be used to perform various applications, such as research purposes and security measures. Researchers and industries have developed several techniques, where the use of methods based on watermarking and steganography can be highlighted.

Watermarking, primarily focused on safeguarding copyrights, brings measure to protect the creator's work and allows for the verification of the authenticity and integrity of the content. It can also trace the distribution and usage of the digital content. On the other hand, steganography is a process that hides information within a common object. This method allows to hide sensitive information from unauthorized access, ensuring confidentiality as well as covering communication. Examples of techniques that use this method are steganography end-to-end solutions, which can be underlined by the following methodologies: HiDDeN [40], SteganoGAN [37], and StegaStamp [31], and CodeFace [29]. These new technologies provide a robust and effective method for hiding messages within digital images. One example of the application of these steganography solutions is exemplified in figure 1.1. In this example, the steganography concept is used to improve the security of documents. It is possible to observe that, concealing a hidden message within a facial portrait on an ID or passport, along with the use of specific tools and databases, can guarantee the documents authenticity.

Furthermore, these methodologies use noise simulation to train their networks. This module develops the robustness of the architecture; however, it presents some limitations regarding noise in the print-scan environment as well as when subjected to extreme compression. Nevertheless, this approach offers a potential path to improve the robustness of end-to-end solutions by enhancing specific points present in their models.

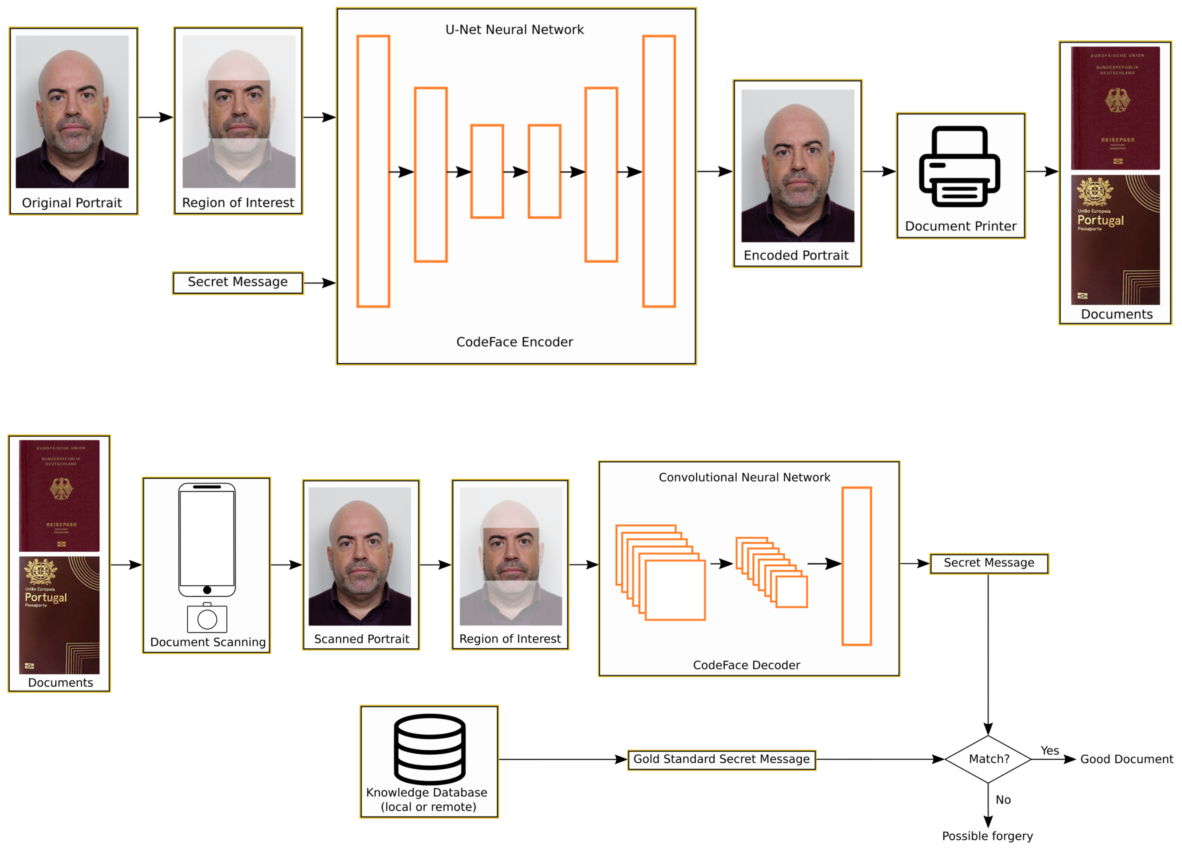


Figure 1.1: Novel approach of an end-to-end steganography solution called CodeFace [29]. This solution is performed in two pipelines, implemented using independent networks, namely the encoder and decoder.

1.2 Problem definition

Steganography end-to-end solutions such as StegaStamp [31], HiDDeN [40], SteganoGAN [37], and CodeFace [29] combine DNNs with steganography. The architectures of these end-to-end solutions follow a similar constitution, in which we can represent the overall architecture of end-to-end solutions in the following figure 1.2.

As can be seen, the architecture of an end-to-end application is composed of four components: the encoder, decoder, noise simulation module, and loss functions. The main goal of the encoder is to hide messages in facial images. The decoder is designed to recover a message that is encoded in a facial image. The noise simulation module allows for the simulation of the physical environment conditions of printing and capturing by image sensors to train the network because these conditions affect the appearance of encoded images and decoder performance. For the last element, loss functions, which evaluate the behavior of your model, are composed of several pre-defined network components and additional loss functions that preserve the appearance of the encoded face and message during training.

In an end-to-end solution, the noise simulation module is one of the components that presents

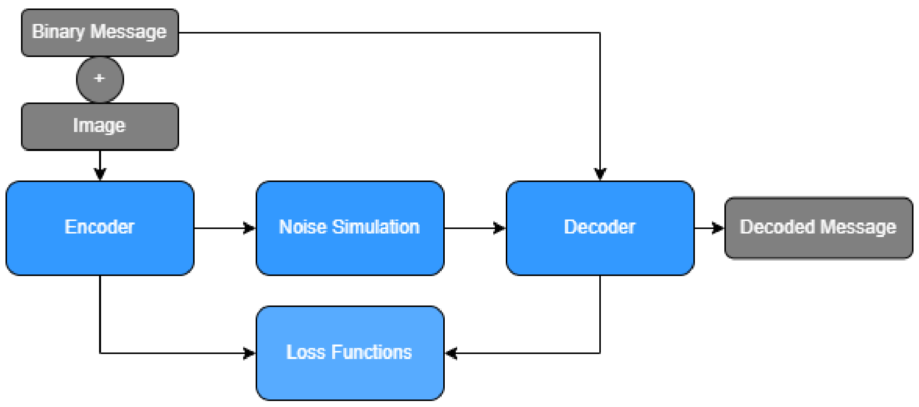


Figure 1.2: Overall architecture of the end-to-end Steganography solution.

limitations, which is the principal problem to be addressed in this dissertation. In this component, it is important to consider a set of different noise sources that occur during the printing and scanning processes since it affects critical points of the model and thus affect the overall functionalities of the solution. One example of this issue is related to color manipulation noticeable during the printing process since printers have a limited gamut compared to the full RGB color space present in displays, and cameras, by using exposure controls, white balance, and a color correction, alter the output. Another example of distortion is the noise introduced by camera systems during the scanning process, which includes photon noise, dark noise, and shot noise. Another important source of perturbation is related to JPEG compression, which affects the image when it is stored in a lossy format such as JPEG.

Another limitation that can be highlighted is the issue that stems from the incompatibility of printers and image capturing devices, such as smartphones. Both devices can cause distortion and loss of quality in the original image, caused by the inadequate reproduction of the hidden message and the resolution of the captured image, consequently affecting the decoding process of the solution.

1.3 Objectives

In order to address the mentioned limitations, it is necessary to focus on two approaches: noise simulation and deep learning methods. Combining these two topics allows for the development of resistance to critical points present in the architecture, which consists mainly of an end-to-end Generative Adversarial Network. This is achieved through the simulation of noise from several sources present in the physical environment and image sensors. Therefore, the main goal of this dissertation is to develop and improve printer-proof steganography through the realization of noise simulation.

Additionally, two partial goals were defined. The first goal consists of achieving a high decoding success rate. The second goal is to increase the robustness of the model while maintaining the performance of the algorithm, since the addition of noise to the model can affect its performance.

1.4 Contributions

This dissertation contributed to the field of steganography with new insights and advancements regarding the use of noise simulation. These contributions can be described as follows:

- Through rigorous experimentation and analysis, this research provides new insights into the behavior and limitations of noise simulation;
- Improves the robustness of steganography solutions models against distortions present in print-scan environments;
- The introduction of Self-Supervising Learning, which is on initial stages, constitutes a new approach in the end-to-end steganography solutions.
- The research conducted has been submitted as an article for publication in the ICPRAM (International Conference on Pattern Recognition Aplications and Methods) conference.

1.5 Structure of the document

The introduction of this dissertation presents the contextualization of an end-to-end steganography solutions, as well as the motivation and objectives of this research. The summary of this dissertation project is presented as follows:

1. (Chapter 2) State-of-the-Art: Summarize description of the different methods reviewed for the project. It includes discussions on Watermarking, Steganography and Printer-Proof Steganography, Noise Simulation, and GANs;
2. (Chapter 3) Methods: In this chapter, the ideas and methodology that were used to develop and implemented this dissertation project are presented;
3. (Chapter 4) Experiments: Description of the used dataset as well as the performed tests for the several hypotheses presented;
4. (Chapter 5) Results and Discussion: Presentation and discussion of the obtained results, as well as the analysis of the hypothesis presented.
5. (Chapter 6) Conclusions.

2 State-of-Art

2.1 Generative Adversial Network (GAN)

Generative Adversial Networks [9, 10, 22] are deep learning architectures with the main purpose of generating realistic data such as images, videos, and audio. In recent times, GANs have revealed promising results in a variety of fields, including image and video synthesis, data augmentation, image translation, and style transfer. GAN is composed of two parts, a generator(G) and a discriminator(D) networks, based on an adversarial process. The generator captures an input (a random signal) and produces a new fake image. The discriminator assesses these generated images, assigning a probability score that represents how distinguishable they are from the original, labeled images. In the last decade, several new methods based on GANs have been developed, of which we can point out the following: Conditional GANs and CycleGANs.

Firstly, the Conditional Generative Adversarial Network (CGAN) [23] was introduced as a novel way to train generative models. In this method, the authors show the ability of the model to learn a multi-modal model and provide preliminary examples of an application to image tagging, where this model approach generates descriptive tags that are not part of the training labels. This method consists of two "adversarial" models: a generative model that captures the data distribution and a discriminative model that estimates the probability that a sample came from the training data rather than the generator. In both models, is added a conditional information (y) that can be any kind of auxiliary information such as class labels or data from other modalities, as represented in figure 2.1. These networks are used to learn the mapping between an input image and an output image using a training set of aligned images by the following loss function:

$$\min_{G}\max_{D} l(G, D) = E_{x \in p_x}[\log(1 - D(G(x|y)))] + E_{z \in p_z}[\log(D(z|y))] \tag{2.1}$$

where x, z , and y represent the inputs of the generator and discriminator model, and p_x and p_z are the image distributions. The generator tries to create images $G(x|y)$ that look similar to real images with the condition y, while $D(z|y)$ aims to distinguish between produced samples and real samples. The generator tries to minimize this objective against an adversary discriminator that tries to maximize it. The method teaches the model mapping $p_x \rightarrow p_y$ by training a generator network.

In conclusion, CGAN, which uses conditioning input instead of solely relying on random

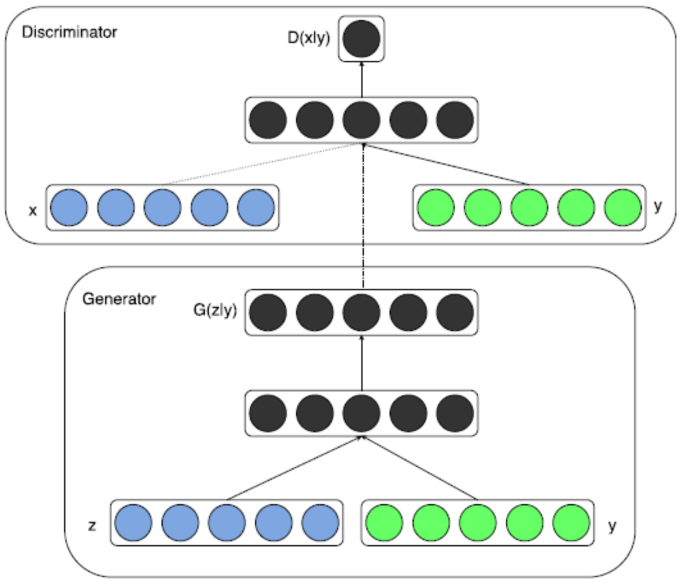


Figure 2.1: Representation of a simple conditional adversarial network. The generator receives as input a noise distribution (z) and a conditional information (y). The discriminator receives as input the real data (x), the created images from the generator, and the conditional information y. Figure from [23]

aversarial noise, offers advantages in approximating many intractable probabilistic computations when compared to traditional GANs. Furthermore, this method is considered the basis for most of the next generation image-to-image translation networks. Beside this, the CGANs models need a dataset of paired images and obtaining a paired example is not always feasible, which becomes an issue.

Image-to-image networks have the goal of learning the mapping between an input image and an output image using a training set of aligned pairs; one example of this method is CGAN, explained previously. CycleGAN [41] is an image-to-image network that tries to learn this generated image mapping without requiring paired input-output images by using a cycle-consistent adversarial network. This method overcomes the issue mentioned in the previous method, CGAN. In order to achieve this, this method introduces two techniques: adversarial losses for matching the distribution of generated images to the data distribution in the target domain and cycle consistency losses to prevent the learned mappings from contradicting each other.

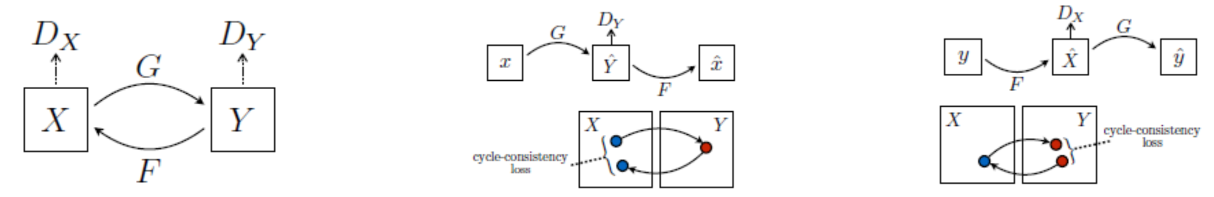
For the mapping functions $G : X \rightarrow Y$ and the function $F : Y \rightarrow X$ and its discriminator D_Y and D_X, adversarial losses are applied according to equation 2.1, where the generator tries to generate images $G(x)$ that look similar to images from domain Y, while D_Y aims to distinguish between translated samples G_y and real samples y. However, adversarial losses alone cannot guarantee that the learned function can map an individual input distribution that matches the target existence, because with enough capacity, a network can map the same set of input images to any random permutation of images in the target domain, where any of the learned mappings can induce an output distribution that matches the target distribution. In this way, in order to reduce the space of possible mappings, it introduces cycle consistency loss, where it enforces forward cycle consistency, $F(G(x)) \approx x$, and backward cycle consistency, $G(F(y)) \approx y$, shown in figure 2.2. Mathematically, G and F have the same structures for learning the inverse mapping for each other. Both mappings G and F are trained simultaneously, and a cycle consistency loss, represented below, is applied to the cycle-network.

$$l_{cycle}(G,F) = E_{x \approx p_{data}(x)}[\|F(G(x)) - x\|_1] + E_{y \approx p_{data}(y)}[\|G(F(y)) - y\|_1] \tag{2.2}$$

The full loss function object introduced in this method is presented as follows:

$$L(G,F,D_X,D_Y) = l_{GAN}(G,D_Y,X,Y) + l_{GAN}(F,D_X,Y,X) + \lambda * l_{cycle}(G,F) \tag{2.3}$$

In conclusion, this method has made contributions to image-to-image networks, resolving one of the issues present in CGANs. Although the method achieves compelling results in many cases, it is important to mention that their performance may not always be uniformly positive and can exhibit failure cases. Nevertheless, one notable aspect is its ability to work with entirely unpaired dataset in various scenarios, pushing the boundaries of what is achievable in this field.



(a) (b) (c)

Figure 2.2: CycleGAN losses functions. (a) Represents the Adversarial loss of this model. The model contains two mapping functions: $G : X \to Y$ and $F : Y \to X$, and associated adversarial discriminators D_Y and D_X. (b) Forward cycle consistency loss: $F(G(x)) \approx x$. (c) Backward cycle consistency loss: $G(F(y)) \approx y$. Figure from [41].

2.2 Watermarking

Watermarking [32, 18] is a technique used to embed hidden information, called a watermark, into digital media such as images, audio, and video. The watermark is usually a digital code, logo, or message that is added to the media in such a way that it is difficult to remove or alter without affecting the quality of the original media. Watermarking is used to protect the copyright of digital media by allowing the owner to prove ownership and prevent unauthorized use or distribution. For this method to be effective, it must be imperceptible and resistant to common image manipulations such as compression, filtering, rotation, scaling, and conventional attacks. Digital image watermarking techniques have been divided into two categories in recent decades: spatial and frequency domain watermarking techniques.

Regarding spatial domain watermarking [19], it is a technique that embeds the watermark directly into the pixel values in the original image. This is done by manipulating the least significant bits (LSB) [6] of the pixel values, which is called LSB insertion. The process of spatial domain watermarking starts by dividing the image into small blocks or regions, and then the LSB of each pixel value in the block is replaced with the watermark information for all the blocks in the image. The watermark information is encoded in the LSB of the image in a way that it can be retrieved and decoded by using the appropriate decoding algorithm. Spatial watermarking is a simple technique; however, it is vulnerable to image processing and compression attacks.

Frequency domain watermarking [17], on the other hand, is a technique that transforms the original image into the frequency domain using techniques such as Discrete Cosine Transform (DCT) [13] or Discrete Wavelet Transform (DWT) [3]. Regarding DCT it is a lossy compression technique in the frequency domain where data is lost when the original image is reconstructed from the compressed image. Discrete Wavelet Transform is a lossless compression technique. In this method, the cover image is divided into sub-frequency bands, where each level represents different frequency bands that can contain low-frequency information or high-frequency information. Like the previous method, the image is divided into smaller blocks or regions, and then each block is transformed into the frequency domain using DCT or DWT. Frequency domain watermarking is a technique more robust to image processing and compression attacks, but at the same time, it is computationally more complex than spatial domain watermarking.

In recent years, watermarking has witnessed developments in the field of machine learning . One example of the use of machine learning is an automated image watermarking system using deep convolutional neural networks [39]. The authors proposed a system that is compatible with unsupervised deep autoencoders, in which an input space can be transformed into a latent space (a multi-dimensional representation where data points share similarities or patterns) containing the most representative features. The developed architecture is divided into two groups: embedder network and extractor network. The embedder network takes two inputs, the watermark and the cover image, and transforms both inputs into a marked image. Instead of assigning some unnoticeable portions of the visual components as the watermark, the embedder network learns to replace the visual appearance of the feature space (the input watermark space is mapped to one of its latent spaces) with the cover image while maintaining the characteristics of the feature space. On the other hand, the extractor network takes in a transformation of the marked image and learns to separate and reconstruct the feature space and the watermark.

Overall, the method transforms two input spaces into a desired latent space and reconstructs one of the inputs from the latent space. The recovery ability of the autoencoders, which ensures an exact reconstruction of the input with appropriate features extracted by the deep neural networks, can secure the feasibility of the proposed method structure.

In conclusion, watermarking is an active field of research with many promising developments. Advancements in robustness, security, and machine learning are improving the effectiveness of watermarking.

2.3 Steganography

The image steganography methods consist in hiding the existence of a secret message, audio, image, or video into a cover image in such a way that the encoded and cover images are not distinguishable from each other. This field has seen significant advancements in recent years, with new techniques and methods being developed to improve the robustness and security of steganography. The primary challenges of this method consist in invisibility, print-proof, capacity of the secret message, and security. In this section, we briefly review the existing image steganography models in two groups: the traditional methods and deep learning methods.

2.3.1 Traditional methods

In the modern era, traditional steganography consists in utilization of several methods, like Least Significant Bit (LSB) and transform domain techniques. As mentioned in the last section, LSB, is a method that involves replacing the least significant bits of the cover image's pixels with the hidden message. Regarding transform domains, which include methods that involve Discrete Cosine Transform (DCT) or Discrete Wavelet Transform (DWT), the cover image is transformed into the frequency domain, where the hidden message is embedded into the frequency coefficients of the image. These referred methods don't rely on the use of deep learning. In recent years, based on the LSB method, a K-LSB-based method [6] using k least bits to hide the image was proposed, where the addition of relative global histogram stretching (RGHS), which is used to improve the perception of the encoded images, prevented the image quality from decreasing. This method, by involving k least significant bits of each pixel, allows to increase the amount of data that can be hidden in image and provide more robustness against steganalysis attacks.

2.3.2 Deep learning methods

The use of traditional approaches to image steganography is only effective up to a relative payload of around 0.4 bits per pixel. Beyond this point, they tend to introduce artifacts that can be easily detected by automated steganalysis tools and, in extreme cases, by the human eye. In recent decades, image steganography with the approach of deep learning has brought a disruptive change in its capabilities and applications, being an area with very fast growth. These approaches use neural networks as either a component in traditional algorithms, for example, using deep learning to identify spatial locations suitable for embedding data, or as an end-to-end solution, which takes in a cover image and a secret message and combines them into a steganography image, that is, using an encoder and decoder process for the information. Examples of this last technique are HIDDeN and SteganoGANs.

Regarding HiDDeN [40], the author presents a method that consists of an end-to-end trainable framework for data hiding that can be applied to both steganography and watermarking. This architecture can be described in four components: an encoder, a parameterless noise layer, a decoder, and an adversarial discriminator, as we can see in figure 2.3. The encoder receives as input a cover image and a binary message that are incorporated in such a way that the encoder can easily learn to embed parts of it at any spatial location of the output. For that reason, it is necessary to replicate the message spatially and concatenate the message volume to the encoder's intermediary representation. This ensures that each convolutional filter in the next layer has access to the entire message as it convolves across each spatial location. As a result, the encoder produces an encoded image that is visually indistinguishable. The noise layer takes as input the cover image and the encoded image in order to produce a noisy image. This component primary goal is to enhance the robustness of images against various types of distortions. It achieves this by applying six different noise layers, each introducing specific image distortions, including JPEG compression, pixel-wise dropout, cropping, and Gaussian blur. The decoder main goal is to recover the message from the encoded image. To accomplish this, several covolutional layers are used to generate feature channels in the intermediate representation. Then, global spatial average pooling is applied to produce a vector of the same size as the image, and finally, a single linear layer is applied to produce the predicted message. Last, the adversarial discriminator has a similar structure to the decoder, but its output is a binary classification.

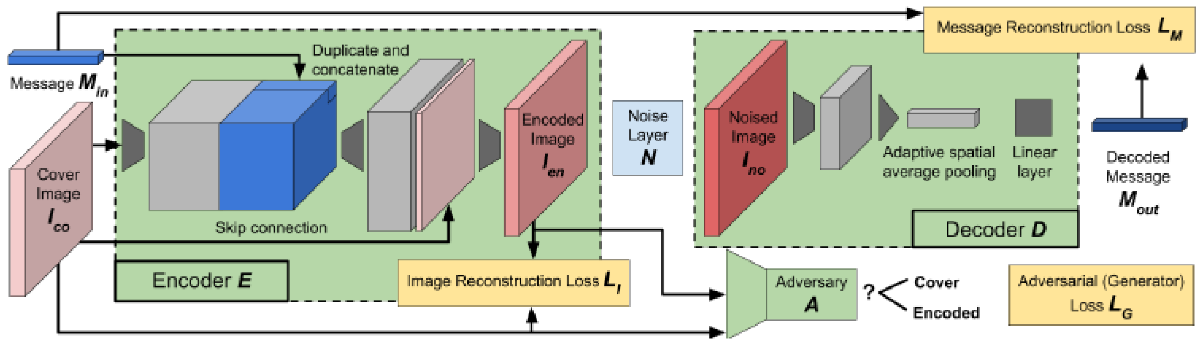

Message M_{in}
Duplicate and concatenate
Cover Image I_{co}
Skip connection
Encoded Image I_{en}
Encoder E
Noise Layer N
Noised Image I_{no}
Adaptive spatial average pooling
Linear layer
Message Reconstruction Loss L_M
Decoded Message M_{out}
Decoder D
Image Reconstruction Loss L_I
Adversary A
?
Cover
Encoded
Adversarial (Generator) Loss L_G

Figure 2.3: The model architecture of HiDDeN. The encoder network is composed of four Conv-BN-ReLU blocks with a final convolutional layer. The decoder network is composed of seven Conv-BN-ReLU blocks. Figure from [40]

When compared to traditional data hiding methods, the end-to-end method HiDDeN allows for flexible trading of capacity, secrecy, and robustness to different types of noise by varying parameters or noise layers during training. In comparison with deep learning methods for steganography, it demonstrate improvements in quantitative and qualitative performance. Overall, the end-to-end solution HiDDeN enhances data hiding robustness through the incorporation of new distortions during the training process.

In SteganoGAN [37], the authors propose a novel technique for hiding arbitrary binary data in images using Generative Adversial Networks which allow for optimizing the perceptual quality of the images produced by the model. The architecture followed by the authors, which we can see in figure 2.4, is divided into three models: an encoder that takes a cover image and a data tensor and produces a steganography image; a decoder that takes the steganography image and attempts to recover the data tensor; and a critic network that evaluates the quality of the cover image and steganography images. The encoder network takes an image and a message as inputs, where the encoder architecture is composed of three variants: basic, residual, and dense. In the first variant, two convolutional blocks are applied to generate the steganography image. The second variant uses the residual connections to improve the model stability and convergence, which therefore improves the quality of the steganography image. The third variant introduces additional connections between the convolutional blocks so that the feature maps generated by the earlier blocks are concatenated with the feature maps generated by the later blocks, which mitigates the vanishing gradient problem (which occurs when gradients become too small during training, affecting the learning and convergence of the model). The decoder takes as input the generated steganography image and attempts to recover the data tensor or message. Regarding the last component, the critic part, where the network consists of three convolutional blocks followed by a convolutional layer, has the aim of providing feedback on the performance of the encoder and generating higher quality images.

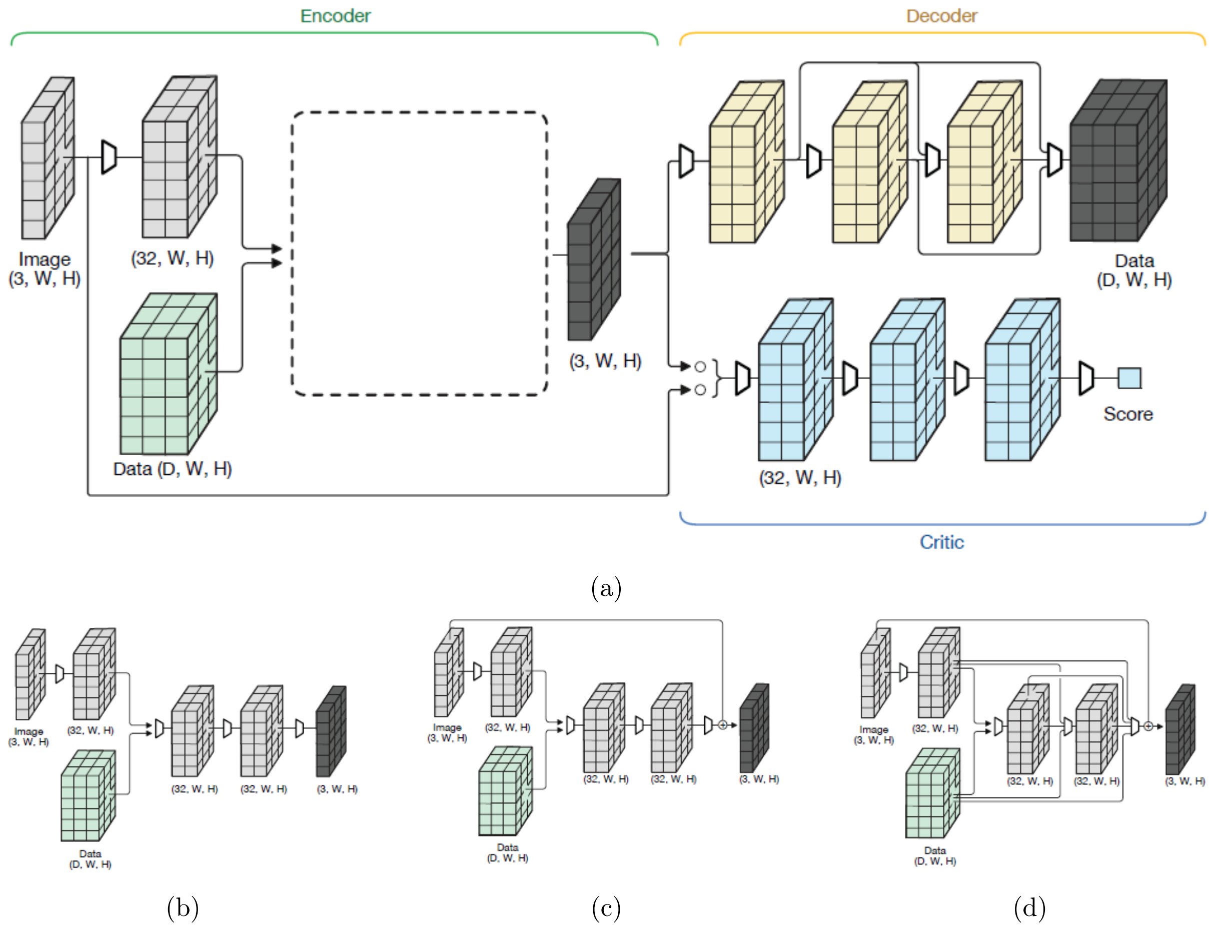


Figure 2.4: (a) The model architecture of SteganoGAN, with Encoder, Decoder, and Critic. The blank rectangle representing the Encoder can be any of the following:(b) Basic encoder,(c) Residual encoder and (d) Dense encoder. The trapezoids represent convolutionals blocks, two or more arrows merging represent concatenation operations, and the curly bracket represents a batching operation. Figure from [37]

The SteganoGAN has brought several innovations that can be highlighted. Firstly, this method introduces a new approach to image steganography that support cover images and arbitrary data of different sizes. Secondly, this end-to-end solution proposes a new metric for evaluating the performance of deep-learning-based steganographic systems that allows for direct comparison with traditional steganography algorithms.

Furthermore, there are other methods that can be highlighted, such as StegaStamp and CodeFace. StegaStamp [31] was the first successful example of steganography with printed images, demonstrating a robust decoding message under physical transmission. The authors propose a learned steganographic algorithm to enable robust encoding and decoding of arbitrary hyperlink bitstrings into photos in a manner that approaches perceptual invisibility, as shown in figure 2.5. This method comprises a deep neural network, in this case, a Generative Adversial Network (GAN), that learns an encoding and decoding algorithm robust to image perturbations, approximating the space of distortions resulting from real printing and photography.

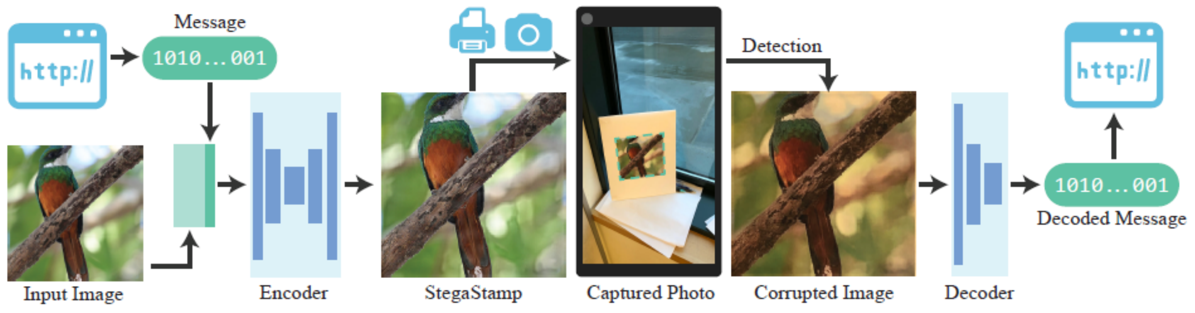


Figure 2.5: Pipeline of the end-to-end solution StegaStamp. First, an encoder network processes the input image and a hidden message. Then the encoded image is printed and captured by a camera. A detection network localizes and rectifies the encoded image before passing it to the decoder network. After the hidden message is recovered and the error corrected, it is possible to follow the hyperlink. Figure from [31].

The architecture of this end-to-end solution, explained in more detail in Chapter 3, is composed of four components: an encoder, a noise simulation, a decoder, and a detector. The encoder has the main objective of embedding a message into an image while minimizing perceptual differences between the input and encoded images. For that, the network of the encoder is based on U-Net architecture [28]. The decoder has the main goal to recover the hidden message from the encoded image. To achieve this, the architecture of the decoder is composed of a spatial transformer network (STN) [16], to develop robustness against small perspective changes that are introduced while capturing and rectifying the encoded image, followed by a CNN. The detector has the main objective to aid the decoder since the decoder network alone is not designed to handle full detection within a much larger image. For that, the detector uses a BiSeNet network [35] to segment the areas of image that may contain the hidden message. Regarding the last component, the noise simulation, represents a set of differential image corruptions added between the encoder and decoder that successfully approximate the space of distortions resulting from the physical transmission.

Nevertheless, StegaStamp has some limitations in its application as a security element to verify the integrity of documents. One of the issues is the relatively large printed images used to demonstrate the technology, which make unsuitable for small facial photo applications as well for identification documents. Another issue is regarding the excessive noise present in the encoded image when compared to the original image. Despite these limitations, StegaStamp demonstrates a robust performance in the decoding process on a variety of printer, screen, and camera combinations. Furthermore, this method proves to be stable enough to be deployed in society as a replacement for existing barcodes, as it is less intrusive and more aesthetically pleasing.

In CodeFace [29], the authors introduce a novel deep learning printer-proof steganography approach for document security systems. This new approach was inspired in the StegaStamp model and introduces a new security system for encoding and decoding facial images that are printed on common IDs and MRTDs.

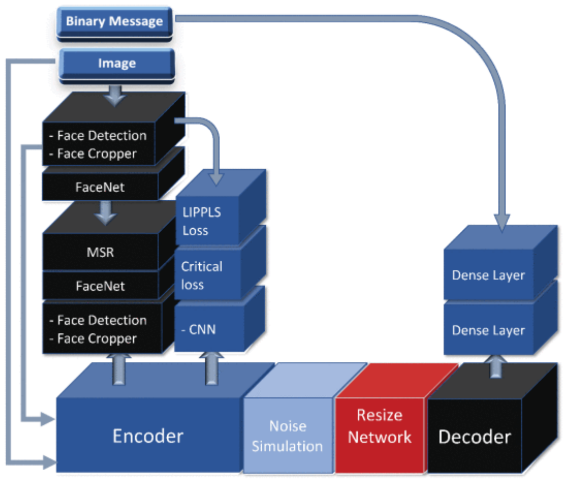


Figure 2.6: Architecture of the end-to-end Generative Adversarial Network, called CodeFace. The generator is composed of an encoder, a decoder and a noise simulation module. The discriminator is a combination of face detection, alignment and cropping systems, a CNN and a Simple Dense layer (fast forward network). Figure from [29].

The architecture of this approach, presented in figure 2.6, uses a GAN that is composed of four components: CNNs for the encoder and the decoder, a noise simulation module, and loss functions. The encoder network is the first part of the generator and has the aim to optimize the trade-off between its ability to restore perceptual properties of the input images and the decoder performance to extract the hidden message. The encoder architecture is based on U-nets; however, the pooling layers were removed to preserve the information of the secret message that may otherwise be lost during network training. Thus, it receives a $400 * 400 * 3$ image and a random binary message as input and generates an encoded image of the same size. The decoder network is incorporated into the whole architecture after applying the noise to the images and is designed to recover the hidden message. The architecture of the decoder is composed of a STN followed by a CNN and receives a $100 * 100 * 3$ image as an input. The STN simplifies the steganography decoding task and improves the performance since it crops the appropriate region and normalizes its scale. The noise simulation module has the aim to simulate perturbations that occur from printers and digital cameras. Furthermore, a resize network (using down-sampling) is added to decrease the size of the decoder input, boosting the performance of the decoder. To maintain the appearance of encoded images and messages during the training process, this algorithm is composed of several loss functions, each of which has its own role. The loss functions are the weighted sum of the five image loss terms,

$$Loss = FL_F + PL_P + WL_W + RL_R + BL_B \tag{2.4}$$

where L_F, L_P, L_W, L_B represent the FaceNet loss, LPIPS (Learned Perceptual Image Patch Similarity) perceptual loss, Wasserstein loss, Residual regularization, and Cross Entropy message loss, respectively. The F, P, W, R, B are the weights for each loss function component.

In conclusion, this new end-to-end solution has made several contributions in the field of steganography. In comparison with other end-to-end solutions mentioned in this topic, it improved the perceptual quality of the encoded image and its compliance with modern facial recognition systems and document issuing requirements. Furthermore, CodeFace presents an innovative technology that can be easily implemented in real world document validation systems and applied to IDs and passports as a security measure.

Table 2.1: Summary of the papers cited in the Section 2.3 relative to end-to-end Steganography solutions.

End-to-End Solution	Important Contributions
HiDDeN [40]	- Incorporates new distortions during training to improve data hiding robustness.
SteganoGAn [37]	- Supports cover images and arbitrary data of varying sizes.
StegaStamp [31]	- Robust decoding of messages even under physical transmission.
CodeFace [29]	- Significant improvement in perceptual quality of the encoded image.

2.4 Noise simulation

Noise simulation is a widely used method by researchers to assess image processing algorithms under realistic conditions in various fields of research. It involves introducing artificial noise into digital images or signals to simulate real-world noise effects. This approach evaluates algorithm resilience to varied environmental conditions in image processing, and it is also used in deep learning for enhancing neural network robustness. In this section, we will review methods used for noise simulation, including traditional techniques and Planckian Jitter.

2.4.1 Traditional methods

Traditional methods of noise simulation encompass established techniques that have long been used to replicate real-world noise or degradation in digital images and signals. These methods are essential tools for assessing algorithm robustness, optimizing image processing approaches, and examining the effects of noise on several systems. Gaussian Blur and JPEG Compression are two methods that can be highlighted. Gaussian Blur [24] involves the application of a convolutional filter to an image with a gaussian kernel, introducing a blurring effect into the image. The amount of blurring is dependent on the kernel size, with bigger kernels causing more blur. The Gaussian kernel in 2-D form is expressed as:

$$G_{2D}xy\sigma = \frac{1}{2\pi\sigma^2}e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2.5}$$

where σ is the standard deviation of the distribution, which controls the variance around a mean value of the gaussian distribution and determines the extent of the blurring effect around a pixel, and x and y are the location indices.

On the other hand, JPEG Compression uses lossy compression algorithms to reduce image file sizes, simulating the artifacts and quality degradation of compressed images. One way to implement JPEG Compression is through the use of Discrete Cosine Transformation [27]. This method can be divided into two stages: the transformation stage and the quantization stage. In the first stage, the image is transformed from its original image representation in the spatial domain to a new representation in the frequency domain using Discrete Cosine Transform(DCT), following the next equation:

$$F(w) = \frac{a(u)}{2}\sum_{n=0}^{N-1} f(n)\cos\frac{(2n+1)w\pi}{16} \tag{2.6}$$

where, $a(u)$ represents the scaling factor for u, N represents the number of samples, and $f(n)$ represent the pixel value of the input signal. The value 16 comes from the division of the block size, which ensures that the DCT coefficients are properly scaled for an $8 * 8$ block. The DCT originates a set of coefficients that represent the image of its frequency components, where these coefficients are sorted in order to decrease the magnitude and sorted in a "zig-zag" pattern, since this is a compact representation that reduces the amount of data required to store the image. In the second stage, the DCT coefficients are divided by a set of quantization values, where these values are chosen based on the desired level of compression and the quality of the image that is required. Then these values are encoded using a variable-length coding algorithm, which assigns shorter codes to coefficients with smaller absolute values and longer codes to coefficients with larger absolute values, allowing for more efficient storage of the image data.

2.4.2 Planckian Jitter

Zenit et al [42] analyze the problem of how the traditionally used color jitter negatively impacts the quality of the color features in the learned feature representation. To resolve this issue, the authors recommend replacing this technique with physics-based color augmentation, designated Planckian Jitter, which creates realistic variations in chromaticity, producing a model robust to illumination changes that can be commonly observed in real life while maintaining the ability to discriminate the image content based on color information.

To perform this new image manipulation, the method exploits the physical description of a black body radiator to re-illuminate the training images within a realistic distribution that allows for augmentations to be more realistic. To achieve this, the method receives an RGB image as input and applies a chromatic adaptation transform that simulates realistic variations in the illumination under four conditions. The first condition is to create a new illumination spectrum $\sigma_T(\lambda)$ from the distribution of black body simulator by the implementation of the following equation:

$$\sigma_T(\lambda) = \frac{2\pi hc^2}{\lambda^5(e^{\frac{hc}{kT\lambda}} - 1)} W/m^3 \tag{2.7}$$

where $c = 2,99792458 * 10^8 m/s$ is the speed of light, $h = 6.6261760 * 10^{-34} Js$ is the Planck's constant, $k = 1.380662 * 10^{-23} J/K$ represents Boltzmann's constant, and λ represents the wavelength in the interval between $400mm$ and $700mm$. The represented equation is known as Planck's Law, where we can retrieve the temperature of the radiation of the black body. The second condition consists of transforming the obtained sample into its sRGB representation. The third condition consists of creating a jittered image by reilluminating with the sample obtained in the previous step. This is done by the application of the Von-Kries-like transform, where it is assumed the original scene illuminant is white, as we can see in the next equation:

$$I'^{R,G,B} = I^{R,G,B} \cdot \frac{\{R, G, B\}}{\{1, 1, 1\}} \tag{2.8}$$

where $\{1, 1, 1\}$ represents the original scene illumination, which is white. The fourth condition introduces brightness and contrast variation to the image obtained, forming the Planckian-jittered image. This process is made possible by the following equation:

$$I'' = c_B \cdot c_C \cdot I' + (1 - c_C) \cdot \mu(c_B \cdot I') \tag{2.9}$$

where c_B and c_C, which respectively represent the brightness and contrast coefficients, and μ is the spatial average function.

Overall, the Planckian Jitter method enhances robustness to real-world illumination variations on images while retaining the capacity to differentiate image content through color details. It shows positive results in tasks prioritizing object color (reflectance-related) for discrimination, regardless of varying illumination sources.

3 Methods

In this chapter, the main methods and algorithms used to achieve the main objective of this dissertation, which is the improvement of printer-proof steganography through the enhancement of noise simulations, are presented. This chapter starts by introducing the base algorithm used in this research, which is a modified version of the steganography solution StegaStamp [31]. This algorithm serves as the foundation upon which several techniques are implemented to enhance the model resistance and performance against the distortions encountered in print-scan environments. Following the explanation of the base algorithm, the added noises to the noise simulation module are described. This section presents comprehensive research to identify noise types capable of increasing the model robustness. Alongside the explanation of the different noises incorporated in the noise module, the chapter covers the implementation of additional deep learning methods. These additions consist of the use of Data Augmentation, Self-Supervising Learning, and the addition of a new loss function, the Dual Contrastive Loss. Lastly, is to be mention that the overall algorithm was implemented using the Python language.

3.1 Baseline

3.1.1 StegaStamp

StegaStamp [31] was the first successful example of steganography with printed images, demonstrating a robust decoding of messages under physical transmission. This method enables robust encoding and decoding of arbitrary hyperlink bit strings into photos in a manner that approaches perceptual invisibility. So, this algorithm is considered the starting point to improve printer-proof steganography. StegaStamp is comprised of a deep neural network, a Generative Adversarial Network (GAN), that learns an encoding and decoding algorithm robust to image perturbations that approximates the space of distortions resulting from real printing and photography.

Architecture

The architecture of this steganography solution can be described by the following elements: an encoder, a decoder, a spatial transformer network and a discriminator. The encoder has the main objective to embed a message into an image while minimizing perceptual differences between the input and encoded image. For this, the network of the encoder is based on a U-Net style architecture [28] that receives a $400 * 400 * 3$ pixel cover image and a 100 bits secret

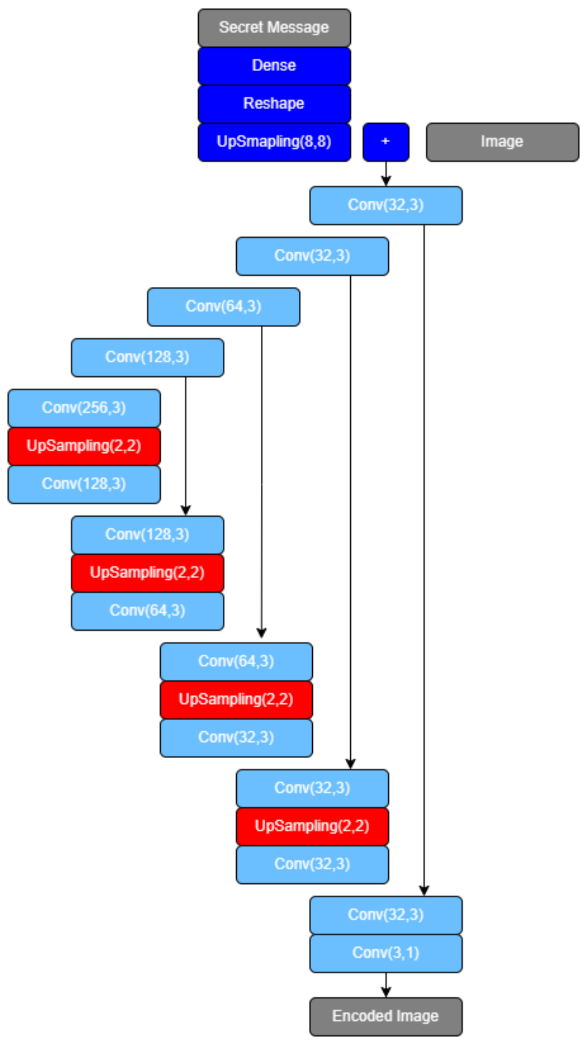


Figure 3.1: The encoder network is based on a U-Net network with no pooling layers (pooling layers reduces the spatial dimension of an input, such as an image or a feature map, while retaining the essential information).

message as input and generates an encoded residual image at the output, as shown in figure 3.1. The U-net style architecture is, as the name says, a convolutional neural network (CNN) with a "U" shape that was specifically designed for segmentation tasks. The purpose of this method is to supplement an usual contracting network with successive layers where pooling operators are replaced by upsampling operators, increasing the resolution of the output. The "U" shape network used in the network of the encoder consists of a contracting path and an expansive path. The contracting path follows the typical architecture of a convolutional network. It is composed of the repeated application of $3 * 3$ convolutional layers, each followed by the activation function Gaussian Error Linear Unit (GELU). In the expansive path, every step consists of an upsampling of the feature map followed by $3*3$ convolutional layers with a correspondent activation function GELU. The cropping is necessary due to the loss of border pixels in every convolution. At the final layer, there is a $1 * 1$ convolutional layer with a Sigmoid activation function, where the output is modified to the range of $[-5; 5]$. This change was made in accordance with an issue that appeared when the algorithm was tested and is explained in detail in Chapter 4. In total, the encoder network has 14 convolutional layers and one dense layer, which is used for the secret message. Then, the encoded residual image is added back to the original to produce the final RGB encoded image.

Before explaining the decoder network, we will explain the spatial transformer network (STN) [16] that will be necessary for the decoder network. The STN allows the spatial manipulation of data within the network. The spatial transformation module is a dynamic mechanism that can actively spatially transform an image or feature map by producing an appropriate transformation for each input sample. The transformation is then performed on the entire feature map (non-locally) and can include scaling, cropping, rotations, and non-rigid deformations. This allows networks that include spatial transformers to not only select regions of an image that are most relevant but also transform those regions to a canonical, expected pose to simplify recognition in the following layers. The spatial transform network, as illustrated in figure 3.2 can be divided into three parts: the localization network, the grid generator, and the sampler.

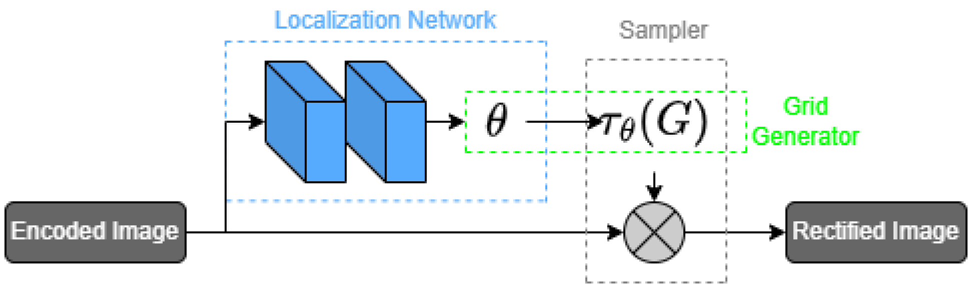


Figure 3.2: Architecture of the Spatial Transformer Network. The input is passed to a localization network which regresses the transformation parameters θ. The regular spatial grid G is transformed to the sampling grid $\tau_\theta(G)$, which is applied to the input, producing the output.

The localization network receives the feature map and, through a number of hidden layers, outputs the parameters of the spatial transformation that should be applied to the feature map, thus giving a transformation conditional to the input. This part of the STN is composed of three $3*3$ convolutional layers, each followed by the activation function GELU, a flatten layer, a dense layer followed by the activation function GELU, and lastly, a regression layer, in this case a linear layer, in order to produce the transformation parameter θ. The second part of STN, the grid generator, creates a sampling grid with the predicted transformation parameters, which is a set of points where the input map should be sampled to produce the transformed output. To perform warping on the input feature map, spatial transformations are applied to modify its geometric arrangement. This process computes each output pixel by using a sampling kernel centered at a particular location in the feature map. The output pixels are defined to lie on a regular grid as $G = \tau_I(G)$, which is expressed by the following equation:

$$G = \tau_\theta(G_i) = A_\theta \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix} = \begin{bmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \end{bmatrix} \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix} \tag{3.1}$$

where (x_i^t, x_i^t) are the target coordinates of regular grid in the output feature map, and A_θ is the affine transformation. The τ_θ allows cropping, translation, and isotropic scaling. The transformation can also be more general, such as a plane transformation with more parameters, a piece-wise affine, or a thin plate spline. The transformation can have any parameterized form, provided that it is differentiable with respect to the parameters; this crucially allows gradients to be backpropagated from the sample points to the localization network output θ. If the transformation is parameterized in a structured, low-dimensional way, this reduces the complexity of the task assigned to the localization network. The final part of STN is the sampler, which has the purpose of creating the output map sampled from the feature map and the sample grid. To perform a spatial transformation of the input feature map, a sampler must take the set of sampling points $\tau_\theta(G)$, along with the input feature map U and produce the sample output feature map V. Each coordinate present in $\tau_\theta(G)$ defines the spatial location in the input where the sampling kernel is applied to get the value at a particular pixel in the output V, which can be written as follows:

$$V_i^c = \sum_n^H \sum_m^W U_{nm}^c k(x_i^s - m; \phi_x) k(y_i^s - n; \phi_y) \forall_i \in [1 \ldots H'W'] \forall_c \in [1 \ldots C] \tag{3.2}$$

where ϕ_x and ϕ_y are the parameters of a generic sampling kernel $k()$ that defines the image interpolation, U_{nm}^c is the value location (n, m) in channel c of the input, and V_i^c is the output value for pixel (x_i^t, y_i^t) in channel c. The sampling is done identically for each channel of the input, so every channel is transformed in an identical way, preserving the spatial consistency between channels. The Spatial Transformer Network is a module that is computationally very fast, does not impair the training speed, and can also increase the speed of attentive models due to subsequent downsampling that can be applied to the output of the transformer.

The decoder has the main goal to recover the hidden message from the encoded image. As mentioned earlier, a spatial transformer network is used to develop robustness against small perspective changes that are introduced while capturing and rectifying the encoded image. The network of the decoder receives a transformed image, which is fed through a series of convolutional and dense layers. The decoder network, shown in figure 3.3 is constituted by seven convolutional layers with a kernel size of three, where each one is followed by the activation function GELU, a flatten layer, and two dense layers, where the first one has the GELU activation function and the second one does not have one. At the end of the decoder network, the sigmoid activation function is introduced to produce a final output with the same length as the hidden message. Also, it is to be mentioned that in the original version, the activation function used in the overall architecture of the StegaStamp was Rectified Linear Unit (ReLU), which was changed to GELU since this activation function allows for better convergence and improves the performance of the model because of the fact that this activation function has a saturation behavior, where it produces non-zero activations for both positive and negative inputs.

Lastly, the discriminator, as the name implies, has the objective of distinguishing between real and fake images. The discriminator network is composed of five convolutional layers with

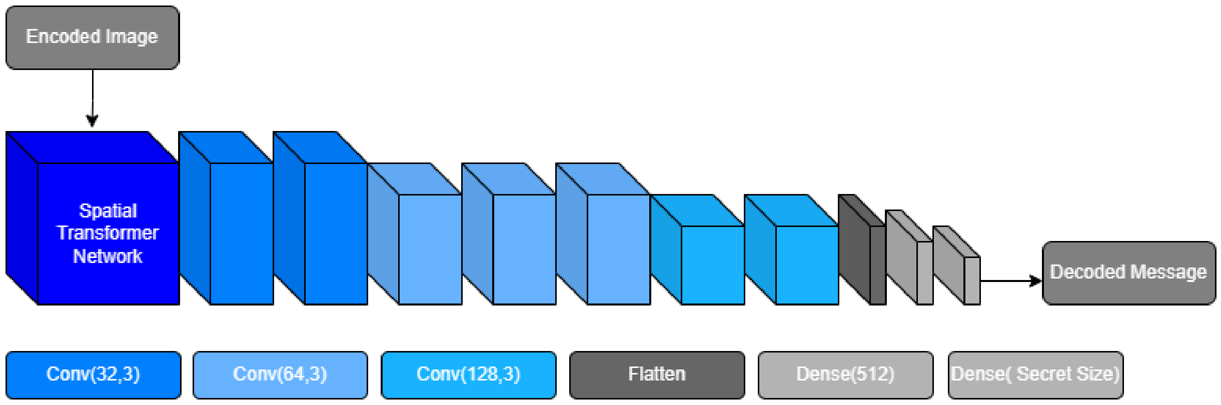


Figure 3.3: The decoder network is composed of an STN module followed by a CNN.

a kernel size of three, where each one is followed by the ReLU activation function with the exception of the final layer. Furthermore, with the addition of the Dual Contrasctive loss, a final linear layer was added, since the addition of this loss function, requires the use of contrastive learning with adversarial learning.

Loss functions and Metrics

As for all of the deep learning systems, the loss function constitutes an important component to evaluate the performance of the model. This algorithm is composed of several loss functions where each one has its own role, which are: image loss, perceptual loss, secret loss, training loss, generator loss, and discriminator loss. The image loss has the objective of evaluating the image compression quality of the generated image by the encoder, where it verifies the preservation of visual details and the minimization of perceptual artifacts. For this, the image loss uses three components: YUV color space, L2 regularization, and Mean Square Error. The YUV color space, which stands for luminance (Y) and chrominance (U and V), is commonly used for image processing to enable efficient image compression. The use of the YUV color space has the purpose of evaluating the color-related distortions between the original image and the encoded image. The L2 regularization has the role of reducing the noise and artifacts in the reconstructed image that can originate from the encoding process by smoothing and regularizing the luminance variations, preserving the clarity and fidelity of the image. The last component, the MSE, is used as the loss metric, weighted for each channel of the YUV color space. The LPIPS perceptual loss [38], as well as the image loss, has the objective to evaluate the perceptual similarity between the original image and the encoded image by comparing the feature representations of both images. The secret loss has the role to evaluate the decoder by comparing the original message with the retrieved message with the use of the binary cross entropy loss function. With the combination of the mentioned losses, the training loss is calculated, where it is equal to the weighted sum of these loss components, as follows:

$$L = \lambda_I L_I + \lambda_P L_P + \lambda_S L_S \tag{3.3}$$

where $(\lambda_I, \lambda_P, \lambda_S)$ represents the loss weights of image loss, LPIPS loss, and secret loss. To evaluate the generator, which is also used to evaluate the encoder, the Wasserstein loss function [1] is used, where the loss of the generator is equal to the mean of the discriminator output. To evaluate the discriminator, it is also used the Wasserstein loss, and, depending on the performed study, can be used the Dual Contrastive loss [36].

To further evaluate the model, StegaStamp uses two metrics: bit accuracy and string accuracy. As the name implies, the first metric is used to measure the proportion of bits that were correctly predicted, and the second metric measures the proportion of characters that were correctly predicted.

Noise module

During the training process, a set of different image perturbations is applied between the encoder and decoder processes to approximate the distortions caused by physically displaying and imaging. The initial version of the noise module present in StegaStamp is composed of five components: perspective warp, Gaussian blur, color manipulation, Gaussian noise, and JPEG compression. The perspective warp generates a random homography to simulate the effect of a camera that is not precisely aligned with the encoded image marker. The Gaussian blur, as the name suggests, simulates the blurring effect on images. The color manipulation represents the limited gamut in printers and displays compared to the full RGB color space, and to simulate these perturbations, a series of random affine color transformations such as saturation and brightness are applied. The last, JPEG compression, simulates the issue of camera images being stored in a lossy format such as JPEG.

3.2 Developed Techniques

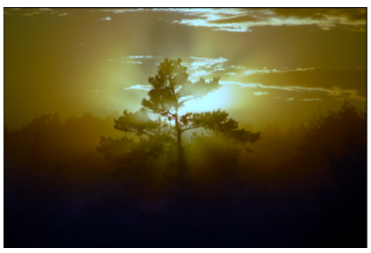
3.2.1 Noise simulation module

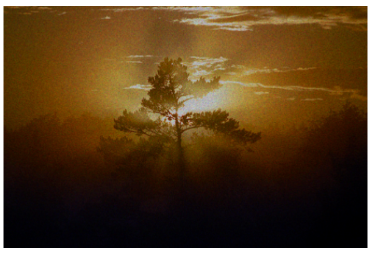
For simplicity reasons, the following figure 3.4 represents examples of different noises sources implemented in the noise simulation module.

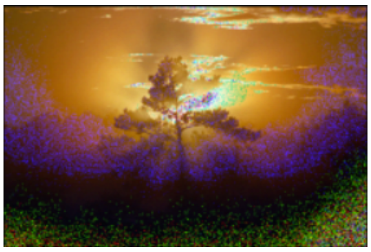
Planckian Jitter

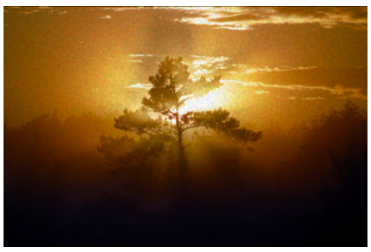
As mentioned in Chapter 2, Planckian Jitter [42], represented in figure 3.4b, has the main aim to simulate the thermal noise that can occur in image systems, allowing the model to be more robust to illumination changes. This method focuses on the variations of the illumination conditions to implement a more accurate variation of the hue intensity and saturation. To perform this, the method exploits the Planck law, where it describes the spectral radiance of an ideal black body at a given temperature to create an illumination spectrum by following the equation 2.7. Using the Planck laws, this method creates new images by following four steps. First, it calculates a new illuminant spectrum $\sigma_T(\lambda)$ from the distribution of the black body radiator. The second step consists of transforming the sampled spectrum to the sRGB representation. The third

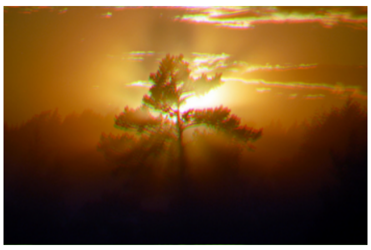
(a) Original image

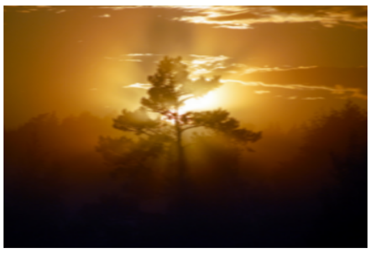

(b) Planckian Jitter

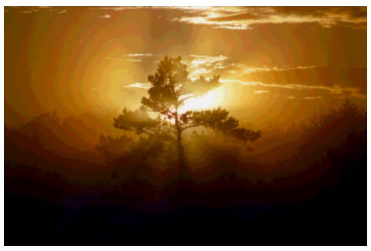

(c) Poisson noise


(d) Dark noise


(e) Speckle noise


(f) Misregistration noise


(g) Motion Blur


(h) Posterization

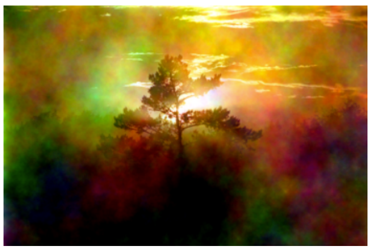

(i) Plasma Brightness

Figure 3.4: Examples of the different sources of noise integrated in the noise simulation module.

step creates a jittered image by re-illuminating the original image with the transform of the second step. Lastly, it introduces brightness and contrast variation to the image, obtaining the final image. The last two steps follow the equations 2.8 and 2.9, respectively. This image manipulation, allows to simulate the illumination changes that can occur in real-world images while maintaining the ability to discriminate the image content based on color information.

Poisson noise

Poisson noise [11], also known as Photon noise, is a basic form of uncertainty associated with the measurement of light, inherent to the quantized nature of light and the independence of photon detection, shown in figure 3.4c. Its expected magnitude is signal-dependent and constitutes the dominant source of image noise in light conditions, with the exception of low-light environments. This noise is caused by the independence of random individual photon arrivals, a signal-dependent form of uncertainty that is a property of the underlying signal itself. The image sensor measures scene irradiance by counting the number of discrete photons incident on the sensor over a given time interval. In a digital sensor, the photoelectric effect is used to convert photons into electrons, whereas film-based sensors rely on photo-sensitivity chemical reactions.

In order to implement photon noise, it is necessary to simulate the independent events that follow a random temporal distribution, which are performed by the individual photons. Thus, it is necessary to follow a process that is able to count photons, known as the classic Poisson process. The number of photons N measured by a given sensor element over a given interval t is described by the the discrete probability distribution:

$$Pr(N = k) = \frac{e^{-\lambda t}(\lambda t)^k}{k!} \tag{3.4}$$

where λ is the expected number of photons per unit time interval, which is proportional to the incident scene irradiance. This equation represents the standard Poisson distribution with a rate parameter λt that corresponds to the expected incident photon count. For small photon counts, photon noise is generally dominated by other signal-independent sources of noise, and for larger counts, the central limit theorem ensures that the Poisson distribution approaches a Gaussian. Furthermore, as we know, photon noise is derived from the nature of the signal itself, it provides a lower bound on the uncertainty of measuring light. Even under ideal conditions, free from all other sensor-based sources of noise, such as read noise, any measurement would still be subject to photon noise. When photon noise is the only significant source of uncertainty, as commonly occurs in bright photon-rich environments, imaging is considered photon-limited.

Dark noise

The dark noise [14], shown in figure 3.4d, can be defined as a random variation of the dark current signal since it results from statistical fluctuations in the number of thermally generated electrons, which contributes to the uncertainty in the dark current value at a given pixel location. The dark current refers to the electric current that flows through a semiconductor device, such as a CCD or CMOS sensor (used in digital cameras and smartphones), even in the absence of light. The cause of dark current can be assigned to two main sources: thermal noise and generation-recombination noise. Thermal noise, also known as Johnson-Nyquist noise, is a type of noise that arises from the random motion of charge carriers, that is, electrons and holes, in a conductor or semiconductor at finite temperatures. The spectral density of the thermal noise is independent of the frequency and is commonly referred to as white noise. The thermal noise follows Gaussian statistics and can be most conveniently characterized by its power spectral density, as follows:

$$\sigma_{th}^2 = \frac{4kT}{R_L}, [W/Hz] \tag{3.5}$$

where R_L is the load resistance, k is the Boltzmann's constant, and T is the absolute temperature. The second cause, generation-recombination noise (GR noise), is a specific noise process in semiconductors that involves fluctuations in the number of free carriers inside a two terminal sample associated with random transitions of charge carriers between states in different energy bands. The essence of GR noise can be defined by considering the simple system provided by a macroscopic semiconducting resistor with resistance R in which the instantaneous number of free electrons $N(t)$, taken as majority carriers, fluctuates between two levels, that is, the conduction band and the donor impurities. Within a relaxation time approximation, GR noise can be related to the resistance R (conductance $G = \frac{1}{R}$) fluctuations with spectral densities given by:

$$\frac{S_R(w)}{R^2} = \frac{S_G(w)}{G^2} = \frac{S_N(w)}{N_0^2} = \frac{\delta\bar{N}^2}{N_0^2}\frac{4\tau_N}{1+(w\tau_N)^2} \tag{3.6}$$

with $S_R(w), S_G(w), S_N(w)$ the spectral density of resistance, conductance, and carrier number, respectively. $\delta\bar{N}^2$ is the variance of carrier number fluctuations, and τ_N the carrier lifetime. The presented equation is of Lorentzian type with two parameters: the relative variance of the number of fluctuations and the lifetime of charge carriers to be determined. The dark current noise plays a crucial role in contributing to the overall dark noise in imaging sensors, having a large influence on image quality in low-light conditions.

Speckle noise

Speckle noise [2], presented in figure 3.4e, is a granular noise texture that degrades the quality of an image as a consequence of the interference among wavefronts in imaging systems. The speckle effect is a result of interference of many waves of the same frequency with different phases and amplitudes, which add together to give a resultant wave whose amplitude and therefore intensity vary randomly. Unlike other types of noise, speckle noise causes uneven pixel distribution. Speckle noise is modeled as the combination of multiplicative and additive noise by the mathematical equation as follows:

$$G(i,j) = g(i,j) * v(i,j) + \eta(i,j) \tag{3.7}$$

where $G(i,j)$ is the obtained image, $v(i,j)$ is the multiplicative noise, $g(i,j)$ is the noise-free image, and $\eta(i,j)$ is the additive noise. The additive noise part refers to the noise that is added directly to the pixel values of an image. It has a consistent variance across the picture and is independent of pixel intensity. The additive component in the context of speckle noise denotes the addition of a constant quantity of noise to the pixel values, which raises the total noise level. On the other hand, the multiplicative part is the noise that scales or multiplies the values of the pixels by a random factor. In this instance, the noise magnitude is proportional to the underlying pixel intensity. The multiplicative noise component results when coherent waves interact with rough surfaces because the noise that is generated can scale with the pixel values.

Misregistration noise

The misregistration noise [33] simulates the noise that arises from the misalignment of image channels, as can be seen in figure 3.4f. This type of noise can occur for several reasons and factors in the registration process or printing process, since perfect alignment may not be achieved. One of the causes could be camera movement, such as small movements and vibrations during image capture, or variations in focal lengths and camera settings. Another factor stems from issues during the printing process. The printing devices have mechanical tolerances, which can result in slight variations in the positioning and alignment of printing plates or rollers. These tolerances can lead to misalignment between different color or printing layers, causing misregistration noise.

Another issue comes from color-to-color trapping, which aims to minimize the impact of this noise. Color-to-color trapping involves intentionally overlapping adjacent colors to create a buffer zone, which helps reduce the visibility of this noise. However, if this technique is not well controlled, it can result in misalignment of the channels image.

Motion blur

Motion blur [20, 21], shown in figure 3.4g, is a well-known noise and is one of the most significant reasons for image quality to decrease. Motion blur is a common optical effect in photographs and videos that occurs when the positions of objects change with respect to the camera point of view during the interval in time where the camera shutter is open. If the objects are moving rapidly or the shutter interval is long enough, then the objects leave a blurred streak in the direction of motion. The motion blur can be divided into several types since it occurs in accordance with the direction and speed of the device or the subject movement. The most common are linear motion blur, radial motion blur, and rotational motion blur. Overall, the motion blur noise can be described by the following equation:

$$g(x, y) = \iint f(x-u, y-v) * h(u, v) du dv \tag{3.8}$$

where $g(x, y)$ is the blurred image intensity at pixel coordinate (x, y) and $f(x, y)$ is the original image. $h(u, v)$ is the Point Spread Function (PSF) or the blur kernel, which represents the intensity distribution of light spread due to motion or other optical effects. In this equation, the blurred image is calculated as the convolution of the true image with the kernel over all possible spatial shifts. This convolution simulates the effect of motion blur by spreading the intensity of each pixel in the original image to adjacent pixels based on the kernel. Motion blur is a difficult noise to work with since it is very difficult to estimate the motion blur effect on the image, since it follows a random distribution based on the exposure time and the characteristics of the device and the individual.

Posterization

Image posterization has the aim of converting an image that has a large number of tones into an image with distinct flat areas and a reduced number of tones, giving it a simplified and graphical appearance, similar to a poster or painting, as can be seen in figure 3.4h. The image posterization process can be divided into two main stages: color quantization and color mapping.

In color quantization [4], the goal is to reduce the number of colors in the image while preserving essential visual information. The pixels in images could have associated 24-bits containing at most $2^{24} = 16.777.216$ different colors. These colors are represented as three dimensional vectors, with each vector element having an 8-bit dynamic range, allowing $2^8 = 256$ different values. These vectors are called RGB triplets, and a smaller representative of the colors of the image is called the color palette. The color quantization process involves building a proper palette that will be used to represent the reduced colors in the posterized image, since the quantity of colors in the palette influences the reduction of the color level in the image. In the color mapping stage, the aim is to map each pixel color value in the original image to the closest color value from the reduced color palette obtained in the previous step, resulting in the posterized image. This step is crucial in creating the final posterized image, as it ensures that the original colors are replaced with the closest matching colors from the palette, resulting in flat regions of distinct colors in the output image.

Plasma Brightness

Plasma Brightness noise refers to a random brightness variation that affects an image, resulting in an image with random and visually patterns with a particular appearance, as shown in figure 3.4i. The plasma brightness effect is a form of synthetic noise that resembles organic and dynamic brightness variations, similar to how plasma display or monitors displays.

To implementation of this effect is based on an old algorithm called the diamond-square algorithm [25]. This algorithm can be used to generate cloud-looking fractals called plasma fractals, where the algorithm requires sparse memory access and square images of size $(2^n + 1) + (2^n + 1)$ for any $n > 2$. The algorithm uses convolutions and can be defined into two steps. The first one is applying a diamond and square step cascade to an existing image and some weighted random pixels, effectively quadrupling its resolution. The second step invokes the first step recursively in order to grow the plasma fractal to an arbitrary size. Furthermore, an important parameter, designated as the roughness parameter, controls the ratio between the existing pixels and random pixels added, controlling the fractal generation and also controlling the frequency range will dominate, whether low or high frequencies.

3.2.2 Dual Contrastive Loss

Dual Contrastive Loss is a loss function used in contrastive learning, which is a self-supervising learning approach. In self-supervising learning, the model is taught to discover usable representations from unlabeled data by developing surrogate tasks that produce synthetic supervisory signals. Contrastive learning aims to map similar samples closer together and dissimilar samples farther apart in the embedding space. In recent years, a new dual contrastive loss [36] has been developed as an extension of the standard constrastive loss. This new loss function is designed to be used in adversarial training, aiming to generalize representations more effectively. Its primary goal is to improve the discriminators ability to distinguish between real and fake samples while incentivizing higher-quality image generation.

As mentioned in Chapter 2, adversarial training relies on the discriminator ability to distinguish between real and fake, and as in other classification tasks, it can be prone to overfitting. Thus, this new method improves the discriminator by incentivizing generation via contrastive learning. Contrastive learning associates data points and their positive examples and disassociates the other points within the dataset, which are referred to as negative examples. With adversarial learning, contrastive learning is used as an auxiliary task and can also be directly coupled with the main adversarial training.

The Dual Contrastive loss has two key components used during training: positive pairs and negative pairs. The positive pairs consist of anchor samples and positive samples. In this case, with the use of a function, the real loss and the fake loss are calculated from the real and fake samples from the same batch, representing the positive pairs. Then the aim of the function is to encourage these positive pairs to have high similarity scores, which, in turn, would lead to mapping the samples close together in the embedding space. Similarly, the negative pairs are also constructed using the same function; however, in this case, the negative samples are created by negating the fake loss and the real loss. Additionally, the function seeks to reduce the similarity scores between these negative pairs so that they are pushed apart in the embedding space. The dual contrastive function can be defined in the next equation as follows:

$$DualContrastiveLoss(Lr, Lf) = -\sum_i Lr_i * \log\left(\frac{e^{Lf_i}}{\sum_i e^{Lf_i}}\right) + -\sum_i -Lf_i * \log\left(\frac{e^{-Lr_i}}{\sum_i e^{-Lr_i}}\right) \tag{3.9}$$

where (Lr, Lf) represents the real loss and the fake loss, respectively. The first part of the equation represents the positive pairs, and the second part represents the negative parts. $\log\left(\frac{e^{Lf_i}}{\sum_i e^{Lf_i}}\right)$ represents the softmax cross-entropy formulation.

With the use of dual contrastive loss, the discriminator can learn more generalized and distinguishable representations, making it easier to distinguish between real and fake samples and incentivizing better generation quality.

3.2.3 Data augmentation

Data augmentation is a well-known technique used in the field of machine learning with the aim of enhancing the quality and diversity of the training data, enabling the model to generalize better to unseen data, and improving its overall performance. There are several types of data augmentation techniques: conventional methods, such as geometric transformations and color space augmentations, are the most simple and widely used, and more advanced techniques that are based on GANs or another machine learning method. In this dissertation, we used two different approaches for data augmentation: one that followed the more conventional way and the other that followed the neural style transfer.

In conventional data augmentation methods [30], a variety of transformations are applied to the dataset, and these transformations are specifically designed to preserve the semantic information of the data while altering its appearance. One important aspect of conventional methods is the necessity of finding a good combination of transformations that maintains the balance between introducing useful variations and maintaining the integrity of the original data. If the augmentations implemented are too aggressive or inappropriate, may lead to unrealistic data samples, which consequently could adversely affect the model ability to generalize to unseen data. On the other hand, if the augmentations added are insufficient, the model is not exposed to a diverse set of examples, limiting the potential of its performance. Conventional data augmentation can be divided into different groups, such as geometric transformations, color transformations, and spatial transformations. The geometric transformations group, as the name implies, is a group that applies geometric operations to the data, especially for image data. Color transformations focus on altering the color and brightness of the images. The spatial transformation group includes methods that rearrange the spatial arrangement of the data. By combining different methods from these groups, the conventional data augmentation methods play an important role in enhancing the robustness and effectiveness of the machine learning methods.

Neural Style Transfer [8] is an interesting and innovative technique of data augmentation where, with the use of deep learning, such as CNN, it is possible to create new images by merging the content of one image with the style of another image, where style refers to the patterns or texture of the image. The base idea behind the neural style transfer is to extract the content and style representations of two different images using a pre-trained deep neural network, such as VGG-19. Furthermore, it is important to notice that in this method, there is no training of a neural network, but instead weights and bias constants are used to update the image; that is, this method optimizes the cost function by changing the pixel values of the image. Thus, the cost function is an important part of this method. The cost function can be divided into two terms: a style loss and a content loss term.

In the style loss method, which has the aim of capturing the patterns of the image, the lower layers of the model are used, and the gram matrix method, which is used to capture the statistics of the lower layers, is used to find the low-level features. The style loss can be defined by the following equation:

$$\begin{gathered} E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} \left(G_{ij}^l - A_{ij}^l \right)^2 \\ L_{style}(\vec{a}, \vec{x}) = \sum_{l=0}^{L} w_l E_l \end{gathered} \tag{3.10}$$

where A^l is the representation of the original image and G^l is the representation of the generated image in layer l. N_l is the number of feature maps and M_l is the size of the flattened feature map in layer l. w_l is the weight given to the style loss of layer l. The content loss is based on the intuition that images with similar content will have similar representations in the higher layers of the network. This loss is calculated as the MSE (Mean Square Error) between the features maps of a selected layer in the neural network. This layer represents the content layer and is chosen based on the level of detail and semantic content desired in the final image, ensuring that the created image captures the same structural elements and objects as the content image. The content loss can be formulated as follows:

$$L_{content}(\vec{p}, \vec{x}, l) = \frac{1}{2} \sum_{i,j} \left(F_{ij}^l - P_{ij}^l \right)^2 \tag{3.11}$$

where P^l is the representation of the original image and F^l is the representation of the generated image in the feature maps of layer l. The neural style transfer process involves optimizing the synthesised image by minimizing both the content loss and the style loss simultaneously, since the final image content and style must be similar to the content of the content image and the style of the style image. The total loss of the neural style transfer is defined as follows:

$$L_{total}(\vec{p}, \vec{a}, \vec{x}) = \alpha L_{content}(\vec{p}, \vec{x}) + \beta L_{style}(\vec{a}, \vec{x}) \tag{3.12}$$

where p, a, x are the content image, style image, and the base image input. α and β are weights for content and style, respectively, defining the adjustments necessary to the intended final result. In conclusion, Neural Style Transfer uses DNN to combine the content of one image with the style of another, creating a unique image by optimizing content and style losses.

3.2.4 Self-supervising learning

Self-supervised learning (SSL) [7] is a promising approach within deep learning since it is one of the most promising ways to build background knowledge and approximate a form of common sense in artificial intelligence systems. Self-supervised learning is a subcategory of unsupervised learning because it leverages unlabeled data. The key idea is to allow the model to learn the data representation without manual labels. Once the model learns how to represent data, it can be used for downstream tasks with a smaller amount of labeled data to achieve similar or better performance than the models without self-supervising learning. SSL obtains supervisory signals from the data itself, often leveraging the underlying structure in the data. For example, as it is common in natural language processing (NLP), SSl can involve concealing part of a sentence and predicting the hidden words using the remaining words, or predicting past or future video frames (hidden data) from current frames (observed data).

In this way, we used self-supervised learning in order to improve the model performance. For that, we use this method to improve two principal components of the model, the encoder and decoder processes. In order to achieve that, in the first step, we have to define the task that will be used to perform the self-supervising problem. As the encoder consists of a U-net network, we can use an information restoration problem in order to improve the network. In this type of problem, the SSL task involves restoring some information that is deliberately removed or modified from the input data, as for example, filling in a missing part of an image or removing the noise from an image. In this case, we implement this problem through the restoration of the original image from its transformed image; that is, the model attempts to obtain the original image from an image with geometric or color transformations. Furthermore, to evaluate the behavior of this approach, image loss with L2 regularization is used, as described in the baseline of this Chapter 3. As for the metrics, the MSE and the PSNR (Peak Signal-to-Noise Ratio) methods are used. Thus, the principal purpose of self-supervising in the encoder network is to improve the U-net network performance without the process of encoding a hidden message on the image but simply to restore the original image from its transformation. As for the decoder process, the problem defined involves a self-supervised learning approach aimed at enhancing the decoding process using a dataset with hidden messages. To evaluate the model, the same loss and metrics described for the decoder process in the baseline of this Chapter 3 are used. The principal purpose is to train the decoder network to decode hidden messages in another environment. Here, the SSL framework leverages the inherent structure and patterns present in the dataset with hidden messages.

3.2.5 Metrics: SSIM, PSNR and Decoding rate

Metrics used to assess images play an important role in evaluating the quality and effectiveness of various image processing tasks. Image metrics provide objective measures that help to quantify aspects like fidelity, sharpness, and overall visual similarity to ground truth or reference images. In this way, to perform a better evaluation of the model performance, we added two metrics in order to evaluate the quality of images during the process of generating an image with a hidden message. For this, we chose two widely used techniques: the Structural Similarity Index (SSIM) and Peak Signal-to-Noise Ratio (PSNR).

SSIM [34, 26] evaluates the structural information, luminance, and contrast similarities between the reference image and the distorted image. One important aspect of this metric is that instead of quantifying error between two images, such as the difference in the values of the corresponding pixels, it follows the idea of the human visual perception of identifying structural information from a scene and identifying the differences from the extracted information. Thus, to implement this metric, the method starts by calculating the gaussian window based on a 1D Gaussian Kernel, that will be used in the convolutional operations. After calculating the gaussian window, the images go through a convolutional operation with the guassian filter window to calculate the sample means and variances necessary to compute the parameters of the SSIM equation, as we can see in the following operation:

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{\left(\mu_x^2 + \mu_y^2 + C_1\right)\left(\sigma_x^2 + \sigma_y^2 + C_2\right)} \tag{3.13}$$

where x and y are the inputs of the images. Also, μ_x and μ_y are the sample means of the input images, σ_x^2 and σ_y^2 are the sample variances, and σ_{xy} is the sample cross-covariance between x and y. The constants C_1, C_2 stabilize SSIM when the means and variances become small.

Peak Signal-to-Noise Ratio (PSNR) [12] evaluates the quality of an image by comparing its pixels to those of a reference image, and it provides insights into the amount of noise or distortion present in the image. The PSNR is calculated by the following equation:

$$\begin{aligned} PSNR(f,g) &= 10\log_{10}\left(\frac{255^2}{MSE(f,g)}\right) \\ MSE(f,g) &= \frac{1}{MN}\sum_{i=1}^{M}\sum_{i=1}^{N}(f_{ij} - g_{ij})^2 \end{aligned} \tag{3.14}$$

where f and g represent the reference image and the testing image, respectively. The value 255 corresponds to the maximum possible pixel value of the image, and lastly, the MSE represents the mean square error between the reference image and the testing image.

The decoding rate is a simple metric created exclusively to quantify the number of encoded images that were successfully decoded, thereby measuring the efficacy of the hidden message retrieval process.

4 Experiments

In this chapter, we provide an overview of the methodical process formulated to drive the development of the dissertation project with the overall purpose of achieving the goals stated in the Chapter 1. The chapter starts by succinctly summarizing the datasets necessary for the research. Following this, it delves into the simulation procedure, thoroughly describing the sequential steps undertaken to meet the predetermined goals. Furthermore, it provides a comprehensive explanation of the strategy developed for implementing each step.

4.1 Datasets

Throughout the development of the dissertation, we used the MIRFLICKR dataset [15] to train the StegaStamp model. The MIRFLICKR dataset, which is composed of 25.000 images, is a dataset with a wide-range of diversity, covering a variety of categories including humans, animals, urban landscapes, and more. The dataset images are each tagged with descriptive annotations to increase their usefulness for tasks like object detection, picture classification, and content-based retrieval.

Furthermore, in order to apply Self-Supervising Learning to the model, specifically to create a task for the decoder network, we use the JMiPOD dataset [5]. This dataset is designed for steganalysis, which, in short, is the process of detecting hidden information within digital media. This dataset is composed of modified JPEG images where the compression technique has been changed to incorporate hidden data.

4.2 Simulation Procedure

To implement and define the different hypotheses throughout this dissertation, a systematic approach was adopted, consisting of several steps, as shown in figure 4.1. The initial phase consists of rigorous research to identify potential noise sources within the environment that could impact the steganography process. Subsequently, a noise ensemble was performed to assess the model behavior towards a group of noise. The third stage, data augmentation is added, boosting the dataset diversity and scale. Lastly, to further increase the model performance, Self-Supervising learning was used.

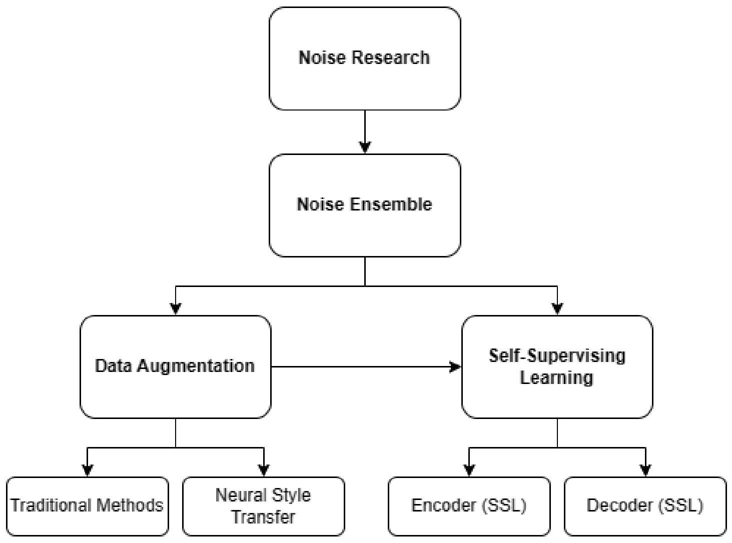


Figure 4.1: Procedural plan developed to implement and evaluate the proposed hypotheses.

4.2.1 Noise research

In the first stage of the plan simulation, a comprehensive study is conducted to assess and improve the robustness of the model against several noise sources, as explained in Chapter 3. Noise, often an inherent part of data acquisition and processing, has the potential to significantly impact the accuracy and robustness of the models. Therefore, delving into noise research allows us to cover diverse sources and manifestations of noise that can distort images and information. It is crucial to identify the moments in the steganography process that are vulnerable to noise. The steganography process is most affected by noise during the decoding process, particularly when the image is captured by a camera and during the printing process.

The environment in which image systems operate is replete with several noise sources that impact the integrity of the image. The most common types of noise are often caused by illumination changes, which can be referred to as the following noises: Planckian Jitter, Poisson noise, Dark noise, and Speckle noise. Another source of distortion is the spontaneous movement of the device or trembling hands during the image capture process, which can result in Motion Blur. Regarding the printing process, one cause of distortion is mechanical and manufacturing inconsistencies that can lead to misregistration noise. Additionally, one of the significant sources of distortions is the limited color gamut of printer devices. To simulate this limitation, Image Posterization and Plasma Brightness noise are used.

4.2.2 Noise ensemble

With the possible noise sources that affect the steganography process defined, we step into the next stage. The noise ensemble encapsulates a diverse set of unwanted fluctuations, distortions, or variations that affect an image. With this approach, it is possible to compute a comprehensive assessment of how well a model can handle real-world scenarios and perceive the performance and accuracy of the model in noisy environments.

Following this approach, we first start to verify the nature of each individual noise and determine its appropriate group assignment. This division by group helps establish the sequence for implementing noise in the noise simulation module. Since there are several noises with different properties, it is quite difficult to define the correct order to add noise to an image. Noise related to camera distortions has more impact than noise originating from the printing process. Therefore, it is necessary to categorize the different types of noise into four distinct groups. The first group is centered around the posterization process, focusing solely on this technique. The second group is composed of noises that specifically originate from the printing process. The third group encompasses noises associated with cameras and sensors. Lastly, the fourth group, similar to the first, features a single noise type, which is JPEG compression.

Having the noise groups defined, the next phase involves a systematic trial-and-error approach to identify optimal noise combinations from these several groups. This iterative process not only unveils the most effective combinations but also refines the impact of each noise on the image through tuning the parameters of the noise. To facilitate this process, a dedicated algorithm was developed. This algorithm aids in the process of generating new noise combination groups by leveraging SSIM and PSNR metrics to quantitatively assess the noise influence on the image.

4.2.3 Data augmentation

Knowing the behavior of the model towards several combinations of noise, we can further improve the model with the use of data augmentation. This methodology applies a variety of transformations and modifications to the dataset, generating augmented versions while preserving the intrinsic characteristics of the original content. In this manner, we can assess the behavior of the network on a large dataset.

Continuing with this new approach, two distinct data augmentation techniques were incorporated. The first technique aligns with the conventional data augmentation approach, incorporating geometric transformations encompassing vertical and horizontal rotations, along with spatial transformations, such as cropping. While color transformations were barely used due to the reason that color transformations mainly focused on manipulating the pixel values intensity of the image, the conversion to gray scale was one of the integrated transformations. The implementation of conventional data augmentation encompassed several stages of dataset expansion. Initially, the dataset size was doubled from 25.000 to 50.000 using a single combination, where it was used to achieve a good combination of transformations. Following this augmentation, the dataset was further expanded to 100.000 images, leveraging two distinct combinations of transformations to ensure diversity. Subsequently, after assessing the model behavior toward a dataset of 100.000 images, it was further increased to 200.000 images, still utilizing the same two chosen combinations.

The second approach uses neural style transfer as a data augmentation method. This technique takes advantage of using two different images to create a new image, where the new image has the contents of one image and the style of the other image. This process enables the generation of diverse variations of images by merging styles and content. This approach not only increases the dataset size but also introduces unique diversity. Initially, the second step focuses on doubling the dataset size by generating new images that combine the content of the original dataset with the style or textures of another dataset. This yields a dataset of 50.000 samples, all of which remains unaltered, that is, without any transformations, providing a foundation for further augmentation. Subsequently, conventional augmentation techniques can be applied to the dataset by using the same combinations techniques from the initial approach. This enables the dataset to expand to 100.000 and 200.000 samples, minimizing the risk of compromising diversity while enhancing the model performance. Furthermore, it is possible to further increase the dataset using neural style transfer, however, it is worth noting that increasing the dataset to a high number of sample counts may become impractical due to the parameter adjustments and increased computational time required for image generation.

In essence, this third step embraces two distinct data augmentation techniques to further increase the performance and robustness of the model. This third step encompasses a step-by-step process in the increase of dataset while keeping attention to the behavior of the model towards the increase of the dataset.

4.2.4 Self-Supervising Learning

With the application of data augmentation, the model performance improves to a certain extent, allowing for further improvement. To further boost the model capabilities, we turn to a transfer learning approach, specifically self-supervised learning. In SSL, the model is trained on a pretext task using unlabeled data, and the knowledge gained from this pretext task is then transferred to improve the performance on a downstream task.

Following this concept, it becomes necessary to first identify the network that could be improved through the implementation of SSL in the multi-network model. The model is composed of an encoder, decoder, spatial transformer network, and discriminator, all of which play essential roles. However, it is worth highlighting that the encoder and decoder play a more important role since they perform essential tasks. Hence, these two network represent our main focus for applying SSL.

After selecting the networks, we proceeded to define the task for the self-supervised learning approach. The encoder network primary function is to embed a secret message within an image. Given that the encoder is composed of a U-net architecture, we opted for an information restoration task by reconstructing the original image from its transformed version. In contrast, the decoder network is responsible for extracting the concealed message. Thus, a self-supervised learning approach is employed to enhance the decoding process, utilizing a dataset containing hidden messages.

In conclusion, this fourth step implements self-supervising learning in the model, with the aim of enhancing the encoder and decoder processes. This approach was implemented with the notion of reinforcing the model robustness but also to evaluate the value of self-supervising techniques in the model.

4.2.5 Simulation conditions

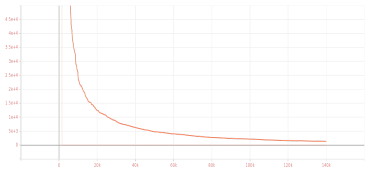
To assess the various hypotheses, it is essential to consider the evaluation conditions. The overall steganography process was conducted within a digital environment, encompassing training, encoding, and decoding operations on digital devices. During the training process, a NVIDIA GeForce RTX 2080 Ti GPU was used, while for encoding and decoding, a NVIDIA GeForce GTX 1650 Ti GPU was used.

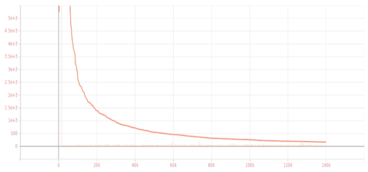
The images used for encoding and decoding maintained the same dimensions as those employed in the training phase, measuring 400*400 pixels. However, it is worth noting that the resolutions differed. During training and encoding stages, input images had resolutions varying from 72 to 300 dpi (dots per inch). On the other hand, the input image for the decoding process consistently had a resolution of 96 dpi, obtained through the encoding process. This resolution is suitable for use in digital and print applications.

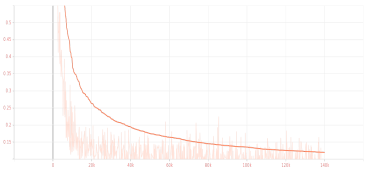
4.3 Setbacks

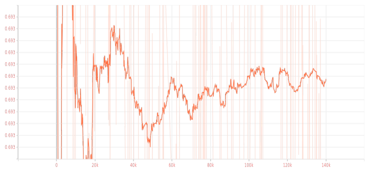
In this section, it will be addressed some setbacks that occurred during the dissertation project. These setbacks encompass some obstacles and difficulties that affect the project timeline and desired outcomes.

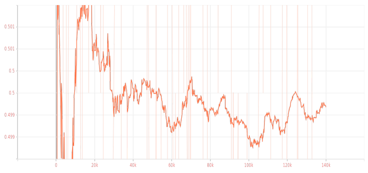
The first hurdle appeared in the initial steps of the noise research implementation, manifesting as an issue affecting the system functionalities. The normal behavior expected of the model is that all its components perform their purpose, while the loss functions and metrics exhibit standard behavior for these elements. However, in this instance, the model exhibited unexpected behavior. In the following loss functions, we can observe the unusual behavior of the loss function, the metrics, and the expected behavior of the same elements.


(a) Image loss


(b) Training loss


(c) Perceptual loss


(d) Secret loss


(e) Bit accuracy

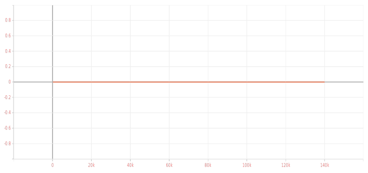
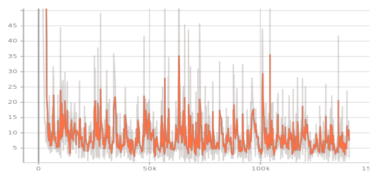
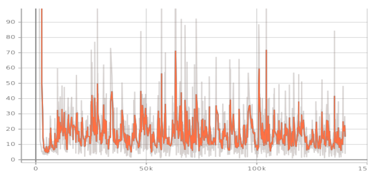
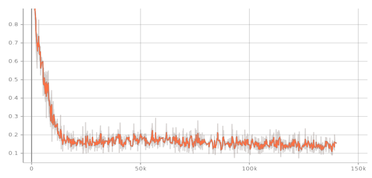
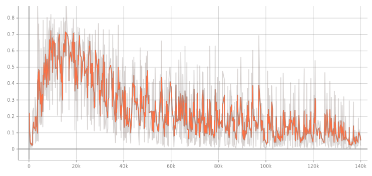

(f) String accuracy

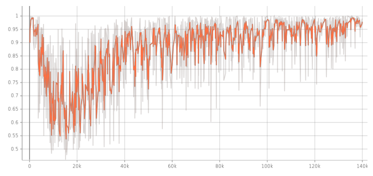
Figure 4.2: These loss functions and metrics represent the execution of the model in its original version. This results is a representation of the model behavior with the issue described.


(a) Image loss


(b) Training loss


(c) Perceptual loss


(d) Secret loss


(e) Bit accuracy

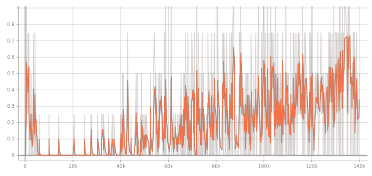

(f) String accuracy

Figure 4.3: These loss functions and metrics represent the execution of the model in its original version.

As it is evident from the analysis of figures 4.2 and 4.3, the loss pertaining to the secret message and the metrics measuring the similarity between the original and decoded messages are confined within a narrow range, which is a very different behavior from the expected outcome. Thus, in a first approach, it was assumed that the problem could come from the decoder network, since the loss of the secret message is a reflection of the decoder performance. Thus, after performing some debugging, it was noticed that if we did not use the decoder network with the spatial transformer network, the model would perform the expected behavior. For this reason, some small modifications were made to the network of the decoder and the spatial transformer network to stabilize the model. These adjustments encompassed altering the activation function and the optimization function of the model. Despite these alterations, the issue still persisted. As a result, two approaches were formulated: the issue potentially stemmed from a different part of the model or exploring an alternative to the STN while, in the meantime, proceeding with the initial phase of the plan without its utilization.

Subsequently, under the first procedure, it was questioned if the problem might stem from the encoder network itself, since its role is to generate an image with hidden messages. Taking into consideration the architecture of the encoder, it was hypothesized that the problem could potentially lie within the final layer, since the output of the layer consists of unprocessed values. Consequently, it was applied activation functions to limit the output value. In a first attempt, the Tanh function and the Sigmoid function were added, but still did not resolve the problem. In a second attempt, we altered the output of the Sigmoid activation function from the interval of $[-1; 1]$ to $[-5; 5]$ and in this case, it solved the issue. By performing this adjustment, it is effectively scaling the output of the encoder, allowing the output to be more manageable to work in downstream task. Furthermore, with this modification, it helps to regularize the output of the network from becoming excessively confident or unstable in its predictions.

The second obstacle emerged during the transition from the noise ensemble to the integration of data augmentation. Analogous to the initial issue, this issue also affected the performance and functionalities of the model. It also occurred within the encoder, where, in the initial stages of the process, an anomaly was observed. The encoder generated an image consisting of a single solid color instead of the expected image. This behavior deviated from the normal occurrence, as illustrated in figure 4.4.

(a) Expected image (b) Image with the issue

Figure 4.4: These images show the anticipated image during the normal training process of the model, and the image appears during this second issue, in the early epochs of the training process.

In instances when this behavior occurred, the model deviated from its intended functioning. The behavior of the model resembles that depicted in figure 4.2 above. As we can see, the possible cause of this problem is the model divergence. One of the solutions to this problem is to add a new layer between the last layer and the penultimate layer of the encoder network, however, it proved to be ineffective. The alternative solution encompassed modifying the activation function of the other layers, excluding the last layer, where the GELU activation function was added. With this alteration, the encoder architecture would display more stability, effectively mitigating the issue occurrence.

5 Results and Discussion

In this chapter, we delve into a comprehensive analysis of the results obtained throughout the course of this dissertation. The chapter starts by showcasing the outcomes of each step outlined in the previously defined plan procedure, explaining in detail the aspects of each result. Subsequently, a meticulous and detailed analysis of the obtained results is presented, accompanied by thorough explanations that aim to provide a deeper understanding of the obtained results.

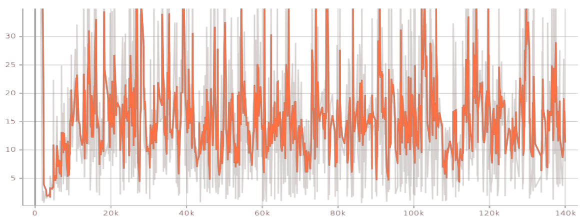
5.1 Results

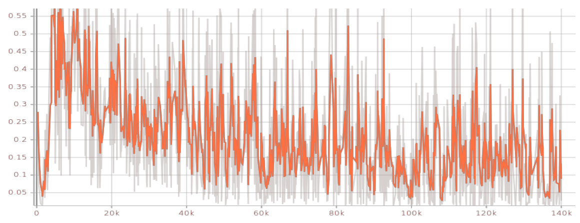
The results obtained during the realization of this project will be presented following the plan procedure mentioned in the previous chapter. Firstly, the base result of the performance of the original code will be mentioned in order to provide a basis for comparison with the obtained results of the following stages of the procedure plan. Additionally, it is important to highlight some aspects:

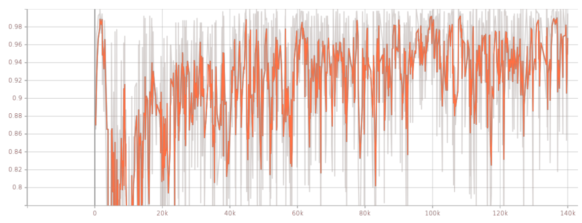
- The scale of the graphic losses is approximated to a smaller scale, since it was intended to give a better notion of how the model converges and its performance. If the graphic loss had its original scale, it would be impossible or very difficult to notice the convergence and performance of the loss;
- The SSIM metric is represented by the mean value of several images, which ranges between 0 and 1;
- The PSNR metric is represented by the mean value of several images, which ranges between 0 and 100 decibels.

5.1.1 Baseline

Before presenting the results of the implemented hypothesis, the results of the base execution of the model will be presented. During in the initial phase of the noise research stage, we encountered a hurdle, resulting in two sets of base execution results, present in table 5.1: one with all the components of the base model in execution, nominated as base 1 (figure 5.2), and the other with all base components except for the Spatial Transformer Network, designated as base 2 (figure 5.1). Furthermore, it should be noted that the noise simulation module, in its base version, already incorporates certain predefined noises, as mentioned in Chapter 3.


(a) Training loss


(b) Secret loss


(c) Bit accuracy

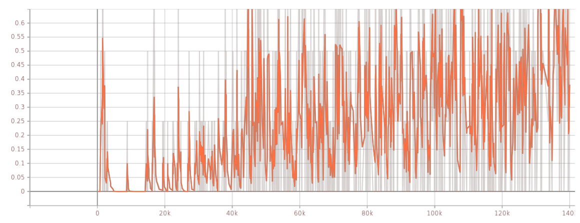
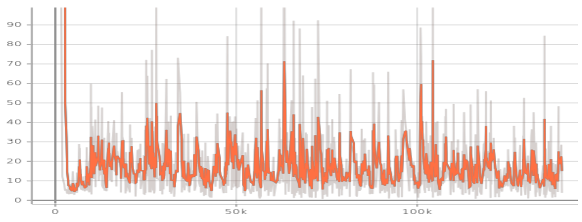
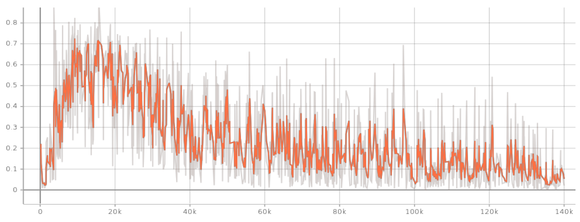
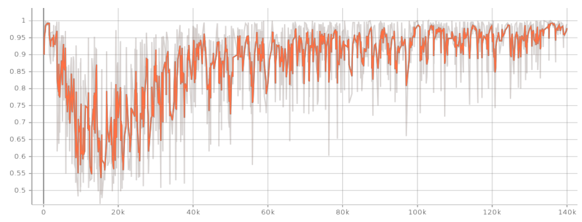

(d) String accuracy

Figure 5.1: These loss functions and metrics represent the base execution of the model in its original version without the use of the spatial transformer network component.


(a) Training loss


(b) Secret loss


(c) Bit accuracy

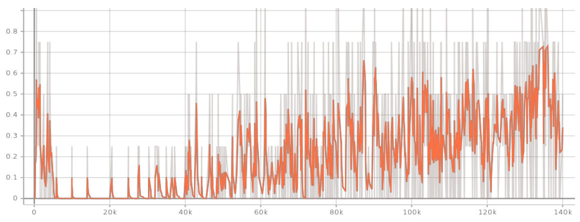

(d) String accuracy

Figure 5.2: These loss functions and metrics represent the execution of the model in its original version, with the use of the spatial transformer network.

Table 5.1: Result of the metric decoding rate for both of the baseline results.

Test	Epochs	Decoding rate
Base 1	$140,000$	70.3%
Base 2	$140,000$	80.4%

5.1.2 Noise research

As the development of this stage was performed under the context of the first hurdle mentioned in the Chapter 4, we are presenting two results for the same individual noise. These results showcase the execution with and without the alterations performed during this particular obstacle.

For the initial results of the noise research, shown in table 5.2, the research and implementation of the noise were conducted within the algorithm, where not all functions were operational; specifically, the spatial transformer network was not included. Furthermore, the testing of the individual noises was carried out using the original parameters of the model. The parameter values were as follows:

- Epochs: For all executions involving individuals noises, it was used 140.000 epochs;
- Loss weights: Image loss(2.0); LPIPS loss(1.50); Secret loss(1.5); Secret loss ramp(1.0); and Generator loss(0.5);
- Activation function: The activation functions used in the encoder and decoder were Rectified Linear Unit (ReLU).

Table 5.2: Decoding rate of each individual noise without using the STN module and percentage points difference with Base 2.

Noise					
Planckian Jitter	Poisson	Speckle	Dark	Plasma Brightness	Misregistration
83,8%	84,1%	80,5%	80,0%	76,5%	81,1%
↑ 3,4 pp	↑ 3,7 pp	↑ 0,1 pp	↓ 0,4 pp	↓ 3,9 pp	↑ 0,7 pp

For the second set of results (table 5.3), each outcome was derived from running the model with all functions performing their purpose. Within this set of results, some of the parameters were modified. The update parameter values are as follows:

- Epochs: Similarly to the first results, all individual noise was tested with 140.000 epochs;
- Loss weights: Image loss(2.0); LPIPS loss(1.50); Secret loss(1.75); Secret loss ramp(1.25); and Generator loss(0.5);
- Activation function: The activation functions used in the encoder, decoder, and STN were GeLU.

5.1.3 Noise ensemble

Following the noise research, we proceeded to the ensemble noise stage. Similar to the previous stage, a portion of the development of this stage was influenced by the first hurdle mentioned earlier. As a result, we will be presenting results from both scenarios.

Table 5.3: Metrics of each individual noise with intended model functioning.

	Noise			
	Posterization	Planckian Jitter	Poisson	Misregistration
Decoding rate	75,1%	78,7%	79,9%	84,6%
ΔBase 1	↑ 4,8 pp	↑ 8,4 pp	↑ 9,6 pp	↑ 14,3 pp
SSIM	0,71	0,72	0,68	0,69
PNSR	52,0	52,3	51,6	51,8
	Noise			
	Dark	Plasma Brightness	Motion Blur	Speckle
Decoding rate	77,1%	74,8%	78,7%	80,0%
ΔBase 1	↑ 6,8 pp	↑ 4,5 pp	↑ 8,4 pp	↑ 9,7 pp
SSIM	0,69	0,68	0,70	0,69
PNSR	63,7	63,5	64,1	63,9

For the initial results of the noise ensemble, presented in table 5.4, the noise combination was evaluated under the algorithm where not all functions were operational, as previously stated, such as the absence of the STN model. In this first set of results, it can be considered an experimentation phase where noise was added in accordance with the obtained values of the individual noise. Additionally, the parameter values employed for the subsequent executions remained consistent with the original values, aligning with the parameters used in the first result displayed in the previous section. The first set of results can be enumerated as follows:

- Test 1: Planckian Jitter and Poisson noise;
- Test 2: Posterization and Misregistration noise;
- Test 3: Planckian Jitter, Poisson noise, and Dark noise;
- Test 4: Planckian Jitter, Dark noise, and Speckle noise;
- Test 5: Planckian Jitter, Poisson noise, and Posterization;
- Test 6: Planckian Jitter, Poisson noise, Dark noise, and Speckle noise.

In the context of the second set of outcomes (table 5.5) from the noise combination stage, each combination was obtained by executing the model with all functions operational. Within this set of results, the majority of the test follows the parameters defined in the first set of outcomes. However, a notable alteration was made to the activation functions employed in the decoder and STN network, wherein they were replaced with the GeLU activation function. The second set of results can be enumerated as follows:

- Test 7: Planckian Jitter and Poisson noise;

Table 5.4: Decoding rate of the first set of noise combinations and percentage points difference with Base 2.

Noise combination					
Test 1	Test 2	Test 3	Test 4	Test 5	Test 6
85,5%	67,0 %	80,6%	79,9%	74,6%	76,5%
↑ 5,1 pp	↓ 13,4 pp	↑ 0,2 pp	↓ 0,5 pp	↓ 5,8 pp	↓ 3,9 pp

- Test 8: Planckian Jitter, Poisson noise, and Dark noise;
- Test 9: Planckian Jitter and Poisson noise with different loss weights. In this case was set as: Image loss(2.50), LPIPS loss(2.00), Secret loss(2.0), and Generator loss(1.0);
- Test 10: Planckian Jitter, Poisson noise, and JPEG compression;
- Test 11: Motion Blur , Planckian Jitter, and Phoisson noise;
- Test 12: Posterization, Planckian Jitter, Plasma Brightness, Misregistration, and Poisson noise, with the total of 160.000 epochs.

Table 5.5: Decoding rate of the second set of noise combinations and percentage points difference with Base 1.

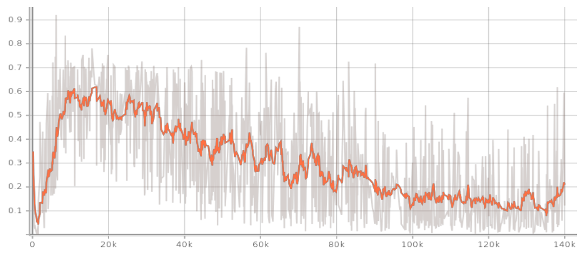
Noise combination					
Test 7	Test 8	Test 9	Test 10	Test 11	Test 12
82,4%	78,0 %	76,2%	78,5%	60,9%	75,9%
↑ 12,1 pp	↑ 7,7 pp	↑ 5,9 pp	↑ 8,2 pp	↓ 9,4 pp	↑ 5,6 pp

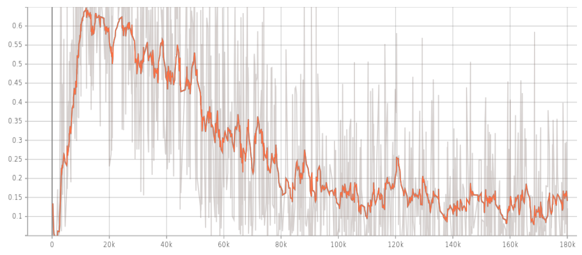
In the third set of results presented in table 5.6, which underwent testing under the same conditions as the second set, three distinct combinations were stipulated. The first combination consists of posterization, planckian jitter, and poisson noise, in that particular sequence. The second group is formed by posterization, planckian jitter, misregistration noise, and poisson noise. The third combination is composed by of posterization, planckian jitter, misregistration noise, motion blur, and poisson noise. For simplicity reasons, it will be label the first group as "Group 1", the second group as "Group 2", and the third as "Group 3".

For the final set of results presented in table 5.7 and figure 5.4, we explored some of the combinations using the algorithm with GANs. This set showcases the outcomes obtained when employing GANs with both the Wasserstein loss and Dual Contrastive Loss. It is used the combination of Test 7 along with combination of Group 1 for the executions.

Table 5.6: Metrics of the second set of noise combinations.

Noise group	Epochs	Decoding rate	ΔBase 1	SSIM	PSNR
Group 1	140.000	81,4%	↑ 11,1 pp	0,71	64,4
	180.000	78,3%	↑ 8,0 pp	0,69	51,7
	140.000	83.6%	↑ 12,3 pp	0,69	57,9
Group 2	180.000	81,3 %	↑ 11,0 pp	0,69	63,9
Group 3	160.000	79,1%	↑ 8,8 pp	0,71	51,8
	180.000	74,6%	↑ 4,3 pp	0,70	51,9


(a) Secret loss of Group 1


(b) Secret loss of Group 2

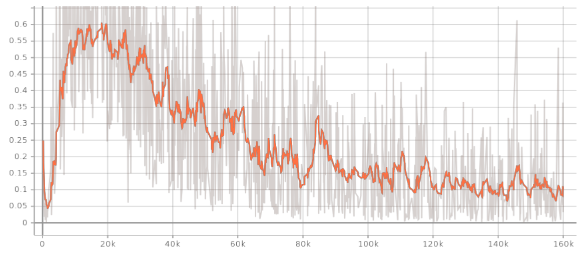

(c) Secret loss of Group 3

Figure 5.3: These loss functions represent the secret loss of each noise combination group.

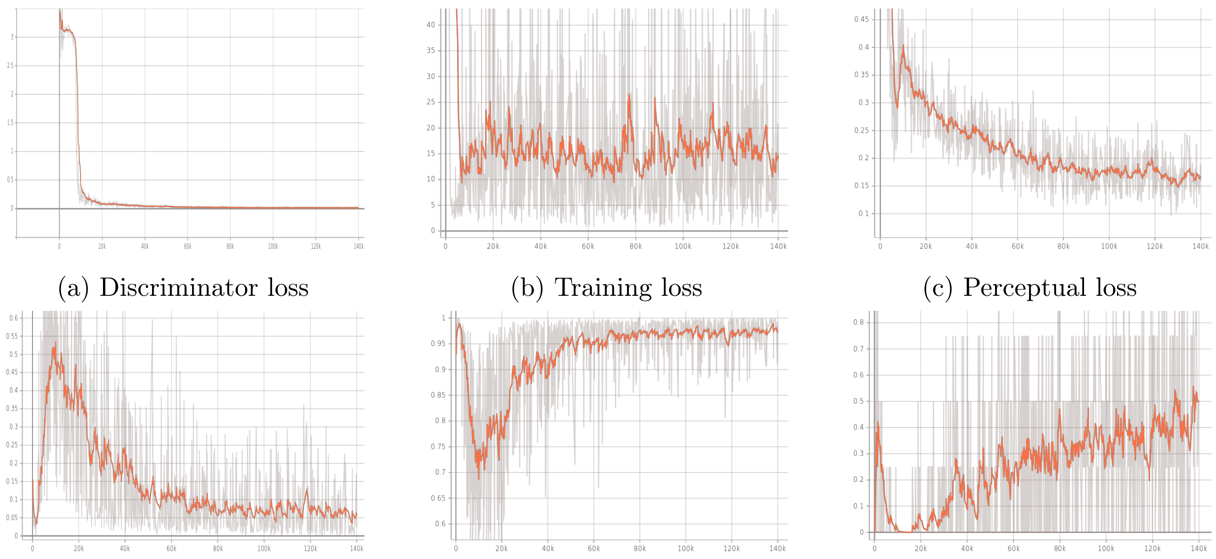
Table 5.7: Metrics of the model training with GANs.

Noise combination	Decoding rate	ΔBase 1	Loss	SSIM	PSNR
Test 7	66,4%	↓ 3,9 pp	Wasserstein		
Group 1	0,2 %	↓ 70,1 pp	Dual Contrastive	0,57	49,3

5.1.4 Data augmentation

To assess the impact of data augmentation on the model, it was predominantly evaluated using three distinct noise combinations: Group 1, Group 2, Group 3. Additionally, it is important to highlight that the parameters used in this development are consistent with those used in the second set of results from the noise research topic.

In the first approach (table 5.8), a simple set of data augmentation transformations was employed. This set is composed of gray scale and horizontal flip transformations, and the evaluation



(a) Discriminator loss (b) Training loss (c) Perceptual loss (d) Secret loss (e) Bit accuracy (f) String accuracy

Figure 5.4: These loss functions and metrics represent the behavior of the model with the noise combination of Group 1 towards the use of Dual Contrastive loss.

was conducted using the Test 7 combination. Subsequently, from this elementary set of transformations, a data augmentation techniques group was formed, resulting in the second set of results present in table 5.9. This group is composed of various combinations, including horizontal and vertical flips, gray scale, color inversion, and auto contrast adjustments.

Table 5.8: Metric results from the development of the first group of data augmentation techniques.

Noise combination	Epochs	Decoding rate	ΔBase 1
Test 7	140.000	75,9 %	↑ 5,6 pp
Group 1	140.000	82,2 %	↑ 11,9 pp
	160.000	81,4 %	↑ 11,1 pp
Group 2	180.000	79,9 %	↑ 9,6 pp

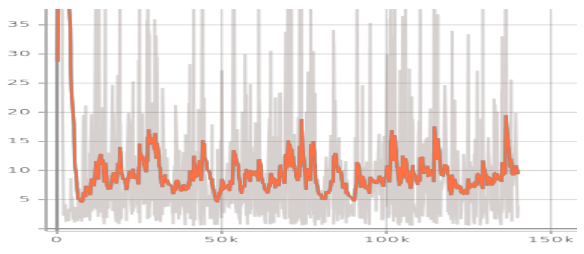
To further increase the size of the dataset to reach 100.00 and 200.000 images (table 5.10 and figure 5.5), another group of data augmentation techniques was developed. This group is composed of the following transformations: horizontal and vertical flip, gray scale, cropping, and auto contrast.

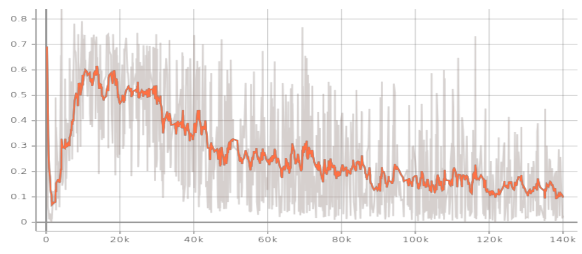
Table 5.9: Metric results from the final version of the first group of data augmentation techniques.

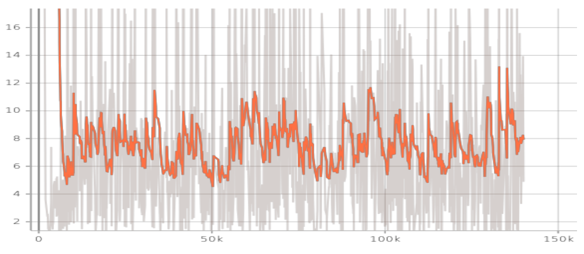
Noise combination	Epochs	Decoding rate	ΔBase 1	SSIM	PSNR
Group 1	140.000	83,0 %	↑ 12,7 pp	0,73	52,4
Group 2	180.000	82,3 %	↑ 12,0 pp	0,69	51,9
Group 3	160.000	81,7 %	↑ 11,4 pp	0,70	52,0

Table 5.10: Result of data augmentation with a total size of 100.00 and 200.000 samples using Group 1.

Dataset size	Decoding rate	ΔBase 1	SSIM	PSNR
100.000	83,7 %	↑ 13,4 pp	0,71	52,1
200.000	80,0 %	↑ 9,7 pp	0,73	52,4


(a) Training loss


(b) Secret loss


(c) Training loss

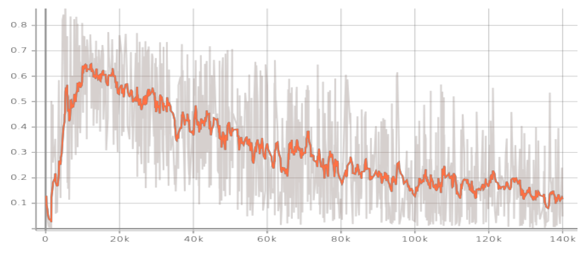
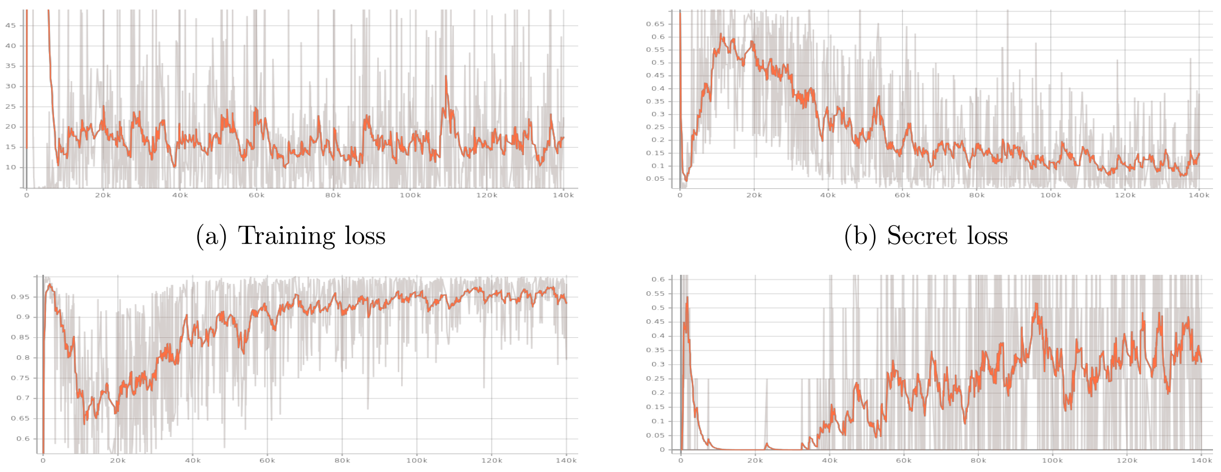

(d) Secret loss

Figure 5.5: The graphics (a) and (b) represent the loss functions of the test with 100.000 samples, and the graphics (c) and (d) represent the loss functions of the test with 200.000 samples. Both executions used the noise combination of group 1.

5.1.5 Self-supervising Learning

To investigate the impact of SSL on the model, we used the noise combination algorithm of Group 1. The results of using SSL for the encoder can be seen in figure 5.7 and table 5.11, while figure 5.7 shows the results for the decoder.



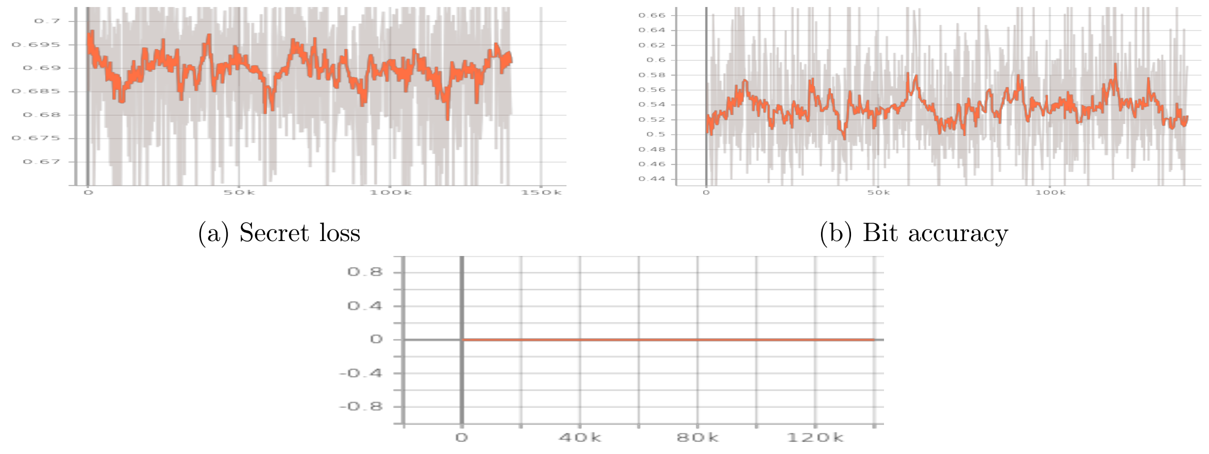
(a) Training loss (b) Secret loss

(c) Bit accuracy (d) String accuracy

Figure 5.6: These loss functions represent the execution of Group 1 with the use of SSL regarding the encoder part.

Table 5.11: Result of SSL with pre-training the encoder, using the noise combination of Group 1.

Decoding rate	ΔBase 1	SSIM	PSNR
82,7 %	↑ 12,4 pp	0,71	58,3



(a) Secret loss (b) Bit accuracy

(c) String accuracy

Figure 5.7: These loss function and metrics represent the execution of the isolated decoder for SSL process.

5.2 Discussion

5.2.1 Baseline

The results presented in this small experiment serve as a foundational reference for assessing the effectiveness and resilience of the techniques incorporated into this project. One important aspect to notice is the distinction in the utilization of the Spatial Transformer Network within the model. Examining the graphical representations in figure 5.1 and figure 5.2, it is possible to verify that the loss functions depicted in figure 5.1 exhibit better convergence and performance compared to those observed in the model with STN.

Subsequently, a comparative analysis of the metrics behavior in Base 2 reveals higher values in contrast to those found in Base 1. The incorporation of the STN into the model significantly increases the overall complexity of the model by introducing an additional network responsible for image rectification before transmission to the decoder network. However, this additional network not only amplifies the model intricacy but also intensifies computational demands, resulting in extended training times. Consequently, the noticeable difference in the decoding rates between the two executions present in table 5.1 can be attributed to the inherent simplicity of the Base 2 model compared to the model of Base 1.

5.2.2 Noise research

The results of the noise research reveal two noticeable aspects. Firstly, there exists a distinct disparity in the degree of increase in the decoding rate metric between the individual noise and their corresponding base model, highlighting a notable contrast between the two scenarios. Secondly, the influence of each noise on the model performance.

In the first aspect, the disparity in values between the two scenarios can be the result of two reasons. The first reason is regarding the evolution of the neural network. The initial set of results presented showcase a network that remains largely unaltered from its base version, since these tests were conducted in the early stages of development. On the other hand, the second set of results pertains to a network that can be considered to be at its best performance, having undergone several modifications, including adjustments to loss weights and network modifications. The second reason is that the varying impact of each individual noise on the model contributes to the observed differences. In the first set, the individual noises implemented have a higher influence on the image, contrasting with the second set, where their impact is comparatively reduced. Consequently, this divergence in influence contributes to increase complexity within the decoding process during training for the first set of results.

The second aspect involves assessing the influence of each noise on the model. Regarding the results of table 5.2, it becomes evident that, on the whole, each individual noise leads to an increase in the decoding rate of approximately $1pp$ to $4pp$ compared to its corresponding baseline, Base 2. However, some of the results did not contribute to the model improvement and, in two instances, even worsened its performance. The principal reason for this behavior is the influence of each noise on the image, which reduces the quality of the image containing the hidden message and increases the complexity of the decoder process.

The results of table 5.3, where the influence of each individual noise is less pronounced than in the previous set, each noise was able to improve the decoding rate by approximately 4*pp* to 14*pp* compared to its respective baseline, Base 1. By looking at the values of SSIM and PSNR metrics, it becomes feasible to evaluate the overall quality of the images generated by these models relative to the base model. The overall values of SSIM are below $0,70$, which indicates noticeable differences between the images. However, certain values surpass this threshold, meaning images with good quality and fewer deviations from the images produced by the base model. Regarding the PSNR values, two observations may arise. Values in the range of $[50, 60]$ indicate images with acceptable quality, suggesting a presence of degradation in image aspects. Conversely, values above 60 are indicative of images with good quality. If the values obtained were below 50, it would suggest a significant presence of degradation in the images.

5.2.3 Noise ensemble

The development of the noise ensemble stage was mostly influenced by a trial-and-error process. From the previous results, it is know what noises increase the robustness of the model, however it is not knowledgeable how the model will behave in the presence of several noises, demanding for a long process to investigate the right influence of each noise into an image, and its position on the noise module.

The results of table 5.4 and 5.5, the primary approach, was a trial-and-error process. In the first set of results, it was observed that combining Planckian Jitter and Poisson noise led to a modest 5*pp* improvement in the model performance. However, in the case where Posterization and Misregistration noise were added, the model performance deteriorated. At this point, it would be possible to continue on two paths. The first path involves continuing to experiment with different pairs of noise combinations. However, this approach is impractical due to the time consumption for each experiment. The second path consists of using the first combination to verify the behavior of the model with a larger combination. From the obtained results, it is possible to conclude two aspects. Firstly, it is necessary to find the optimal balance between individual noises. Secondly, an alternative approach is needed to mitigate the reliance on the trial-and-error process and minimize time consumption.

In the second set of results, represented in table 5.5, the behavior of the same noise combinations with the model functioning with STN was verified. Furthermore, it was observed the behavior of other noise combinations, along with how the adjustment of some parameters would affect the performance of the model. Despite these parameter modifications and the introduction of various noises, the improvements of the model performance remained low, not surpassing the value achieved in Test 7. Thus, in order to find better combinations, a complementary algorithm was used that allowed for the two aspects that were mentioned to be overcome. As a result, in the initial stages of this new approach, a combination of five distinct noises was identified, resulting in an approximate 5*pp* enhancement in the decoding rate. This supplementary algorithm serves the purpose of assessing the impact of a noise combination on an image, allowing the creation of noise combinations with proper influence from each individual noise. However, this algorithm does not provide with any insight of how the model will behave when exposed to a noise combination.

Through the implementation of this new approach, it was possible to formulate three different groups of noise combinations with different number of noises. The corresponding metric values are detailed in table 5.6. A comparative analysis of the SSIM and PSNR metrics reveals the general quality of the images to be satisfactory, as can be seen in figure 5.8. Nevertheless, the values of PSNR suggest that the images contain noticeable distortions. In a comparison between the graphics of figure 5.3, it is possible to notice that the difference between the performance of each model is very small. One thing that can be noticed is that as the number of incorporated noises increases within the model, the convergence trajectory of the secret loss becomes more gradual and stable. A similar behavior is also noticed in the metrics, since the secret loss influences their values. Thus, the increase of noise quantities is accompanied by a reduction in the model overall performance and decoding rate. However, with larger noise combinations, the model attains heightened robustness, stemming from its exposure to diverse noise types during training.

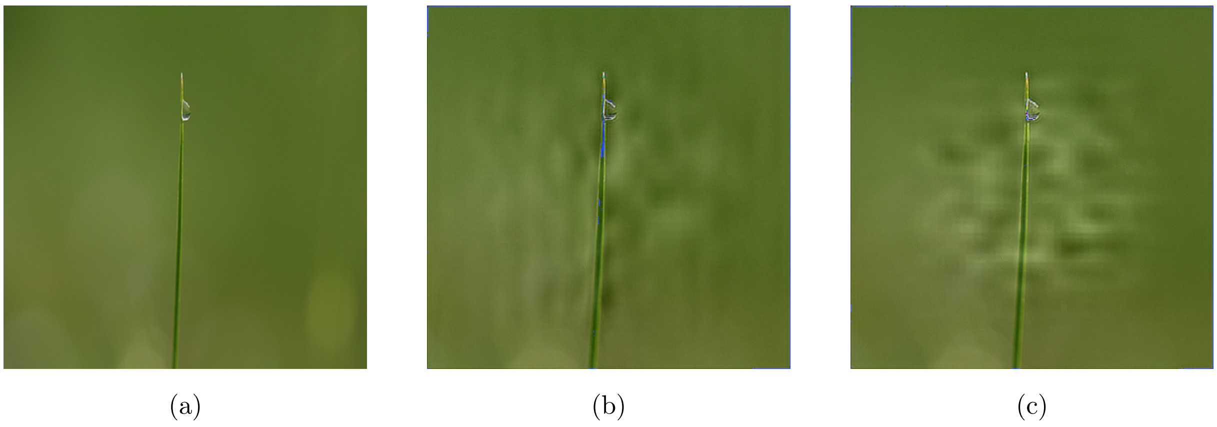


Figure 5.8: Examples of the quality of the encoded image obtained. (a) represents the original image. (b) is the encoded image from Base 1. (c) represents the encoded image from Group 1

The overall development of this dissertation project was performed without employing GANs, since it was achieved satisfactory results through other methods. Meanwhile, some training with this method was carried out. Both of these approaches yielded suboptimal results, primarily due to inadequate parameter fine-tuning. In this process, the loss weights of the model need to follow a particular relation, where the weight of a loss should not be much different from another loss. For example, the difference between the Perceptual loss and the secret loss should not exceed 0.25. Another aspect of this process involves trial-and-error methodology and requires a few executions to obtain a favorable outcome. By looking at the figure 5.4, we can observe that the overall performance of the model is the expected behavior, however, the Perceptual loss graph reveals a slower convergence. This behavior arises from the training process, where the generated images containing the hidden messages exhibit a consistent presence of horizontal lines. Evaluating the values obtained from image metrics in table 5.7, it is possible to state that the created images have noticeable differences and distortions, when compared to the image created by the base version. Hence, it is not possible to definitively determine whether the implementation of the Dual Contrastive loss in the model constitutes a successful approach or not.

5.2.4 Data augmentation

The addition of data augmentation comes from the need to enhance the model performance while maintaining its overall robustness. In the previous section, a pattern emerged: as the volume of noise increased, the model performance demonstrated a corresponding decrease. However, concurrently, the model robustness improved due to the introduction of diverse noise types.

The initial set of results, represented in table 5.8, consists of experiments aimed at identifying effective combinations of data augmentation techniques. This process is important because it is fundamental to finding a good combination of transformations where the balance between introducing useful variations and the integrity of the original data is maintained. This balance prevents the model from generalizing overly similar images, thus averting the problem of overfitting.

Looking at table 5.9, it becomes evident that doubling the dataset size yields a relatively modest growth in the decoding rate, increasing by approximately $2pp$. In this scenario, a subsequent expansion of the dataset ensued, intended to increase the number of samples to be processed. Scaling the dataset to 100.000 and subsequently to 200.000, leveraging distinct combinations to ensure sample diversity, yielded an insignificant increase in the decoding rate. For the dataset containing 100.000 samples, the increase amounted to less than $1pp$, while for the other dataset, a decline in the decoding rate was observed. This occurrence points to an instance of overfitting brought on by an inadequately diversified dataset. Comparing the loss graphics present in figure **??**, reveals that the performance of the dataset containing 200.000 samples is slightly better than the dataset with 100.000 samples, since the loss graphics reveals a convergence to smaller values. Hence, it is possible to conclude that using conventional methods, the overall performance of the model and decoding rate yield a small growth. To solve this impasse, two paths were defined. The first involved testing the model with a more extensive dataset, while the second consisted of the exploration of alternative data augmentation techniques.

Following the second step, we introduce the technique of neural style transfer. This technique creates an image using the style and content of two different images. Using this method, it is possible to mitigate the problem of a dataset with insufficient diversity. Nonetheless, when testing this technique, it was found unsuitable for the task at hand. The process to originate new images involved using the original dataset as the content image and a dataset similar to the original as the style image. However, even by adjusting the parameters of the method, the resultant images displayed inferior quality or exhibited structures that were unsuitable for the problem of this model. This outcome is illustrated in the subsequent figure 5.9.

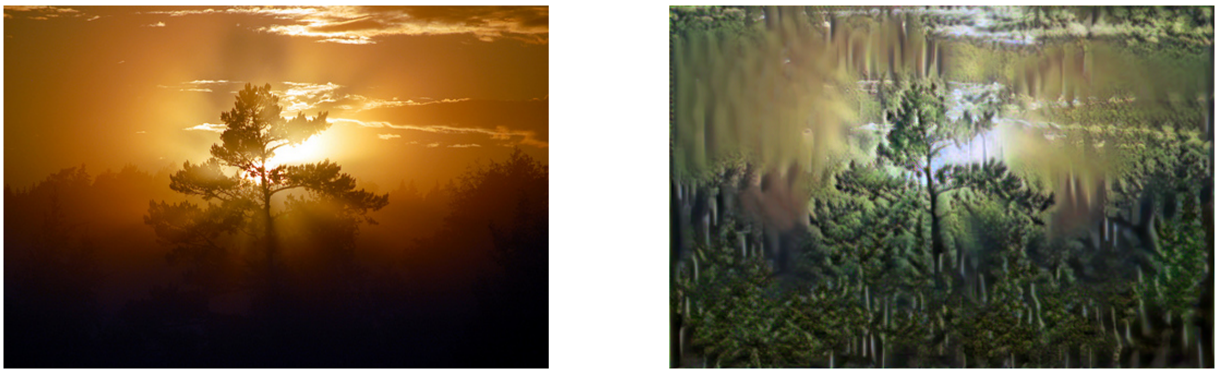

(a) Original image (b) Generated image

Figure 5.9: Image created with the use of the method of neural style transfer.

5.2.5 Self-Supervised learning

With the addition of data augmentation to the model, the overall improvement of the model performance, as well as the metric decoding rate was modest. To improve model performance, Self-Supervising Learning was introduced, focusing on enhancing specific parts of the architecture, such as the encoder and decoder process. Furthermore, is important to note that the implementation of SSL is in its initial stages of development.

Looking at the results obtained in table 5.11 and figure 5.6, it is evident that the results are satisfactory. In contrast to the values outlined in table 5.6, a moderate enhancement is observable when compared with the initial two values, while a decline is visible when compared to the third outcome of Group 1. The execution used for this SSL test is under the same conditions as the third result of Group 1 within table 5.6. A comparison between these two outcomes reveals their similarity; however, it is noticed that the performance of the SSL test worsened. One potential explanation for the modest performance of the SSL result is the suboptimal selection of parameters for the SSL process.

The results obtained for the decoder are not satisfactory. As illustrated in figure 5.7, the isolated decoder model exhibited poor performance. This behavior can be attributed to two reasons. The first reason stems from the dataset used. While the dataset consisted of images with hidden messages, the size of each hidden message varied. The second reason may be caused by the inadequate performance of the message retrieval process. In assessing the model performance and accuracy, it becomes essential to have prior knowledge of the hidden message. Consequently, adopting a method for retrieval message could lead to incorrect interpretation of the retrieved message, thereby jeopardizing the training process.

5.2.6 Global discussion

Based on the results obtained, it can be mentioned that the hypotheses formulated in this research have provided satisfactory outcomes, resulting in an increase in the decoding rate ranging from 10*pp* to 14*pp* compared to its baseline. This notable improvement underscores the effectiveness of the procedure plan defined in Chapter 4. Nevertheless, it is essential to note that the approach chosen is just one of several possibilities that can be developed. Upon a closer examination of the general perspective, there are aspects for further refinement and growth.

Several aspects can be identified for potential improvement in this research. Firstly, the use of a different dataset could improve the research outcomes. The current dataset, which contains 25.000 samples, limits the use of data augmentation and model performance enhancement. It is worth mentioning that a larger and more diverse dataset tends to yield improvements in the model performance. Therefore, choosing an alternative dataset with more samples may lead to better results. Secondly, the ensemble of noise groups is an approach focused on trial and error. Thus, improving or developing a new approach for identifying new combinations more effectively could lead to finding new groups of noise combinations that could have similar or better outcomes. Lastly, the use of deep learning techniques, such as Self-Supervising Learning, in this research is in its initial stages. Compared to the existent state-of-the-art, it is a new approach to improve the performance of the model while maintaining the robustness of printer-proof steganography solutions. Thereby, it is a method that needs further investigation and development to find the ideal technique for the question at hand.

The overall development and testing of this dissertation project took place within a digital environment, providing valuable insights into the model behavior in relation to a print-scan environment. However, it is feasible to make assumptions about the process towards application to printed images. The results expected to be obtained would be similar to those in the digital environment. It is expected to improve the effectiveness of the decoding process since the model would have better robustness against the distortions that occurred during the print-scan process. Another important aspect of printed images is the possibility of a better assessment of the effectiveness of the developed hypothesis.

In the meantime, the overall process developed is not yet suitable for real world applications. This research has increased the robustness of printer-proof end-to-end solutions; however, it does not achieve the necessary effectiveness to be employed in the real world. This research can indeed be employed for real world applications, such as improving the efficiency of the security measures of IDs, with the condition that it be further developed.

6 Conclusions

6.1 Conclusion

Nowadays, images are widely used across diverse applications, transforming the way we interact with the digital world. One important use of images is their innate ability to carry information, in which we can underline its capacity to have concealed information. This concept has led to the development of several applications, not only in security measures but also in various aspects of the daily live. Two prominent techniques that leverages this concept are watermarking and steganography, which greatly contribute to the dynamic of concealing information within an image.

In recent years, steganography, computer vision, and deep learning have represented the vanguard in this branch. Steganography models attempt to conceal a secret message into the encoded image with the aim of making it imperceptible to the human naked eye. The process of extracting the hidden message from the cover image becomes feasible only with the appropriate decoder tools. End-to-end steganography solutions have introduced widely used techniques and new methods to produce encoded images. These steganography solutions are trained under adjusted conditions to produce new images with better quality and increase the difficulty of perceiving the hidden message.

In this dissertation, we have improved an end-to-end steganography solution named StegaStamp. This end-to-end solution was the first successful example of steganography with printed images, demonstrating a robust decoding message under physical transmission. Within this approach, we have not only enhanced the noise simulation module but also improved the model performance in the face of increased robustness against diverse real-world noise sources. The proposed method adopts a step-by-step development, imparting the model with a diverse array of noise types and combinations, ultimately augmenting its resilience. Furthermore, through the implementation of data augmentation techniques and deep learning methods, such as Self-Supervising learning, the model performance has been improved while maintaining its robustness. These measures also lay the groundwork for future advancements in enhancing the model performance.

In addition, the results of this project demonstrate that the use of several sources of noise can provide a robust and efficient method to improve printer-proof steganography. Furthermore, it illustrates the challenge of increasing the robustness of the system without compromising the performance of the model and the outcomes of the metrics.

In the end, this dissertation project has contributed to the improvement of printer-proof steganography through the development of noise simulation. The proposed work is one of the many approaches to improve printer-proof steganography, and there are many open challenges and opportunities for future research in this field.

6.2 Future work

During the development of this dissertation project, one of the aspects that could be further developed and researched is the incorporation of deep learning techniques such as transfer learning to enhance the model performance. Some preliminary results of the use of Self-Supervising learning were presented in this work. From these results, it was concluded that it is still necessary to undergo further investigation and, as well, a new approach to implementing these techniques into the model. The use of SSL can bring many advantages to the model. The SSL method empowers models to autonomously extract meaningful features and representations from data, enabling better initialization for downstream tasks. Another aspect is the versatility of the models. The learned representations can be transferred and fine tuned on various specific tasks, leading to improved generalization. Furthermore, this method of deep learning, has new possibilities for strategies to be implemented and tested, making it an area of research and growth.

7 Bibliography

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 214–223. PMLR, 06–11 Aug 2017.

[2] P. Arulpandy and M. Pricilla. Speckle noise reduction and image segmentation based on a modified mean filter. *Computer Assisted Methods in Engineering and Science*, 27(4):221–239, 2020.

[3] M. Barni, F. Bartolini, and A. Piva. Improved wavelet-based watermarking through pixel-wise masking. *IEEE Transactions on Image Processing*, 10(5):783–791, 2001.

[4] Sebastiano Battiato and Rastislav Lukac. *Color-Mapped Imaging*, pages 83–88. Springer US, Boston, MA, 2008.

[5] Rémi Cogranne, Quentin Giboulot, and Patrick Bas. Alaska#2: Challenging academic research on steganalysis with realistic images. In *2020 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–5, 2020.

[6] Omar Elharrouss, Noor Almaadeed, and Somaya Al-Maadeed. An image steganography approach based on k-least significant bits (k-lsb). In *2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT)*, pages 131–135, 2020.

[7] Linus Ericsson, Henry Gouk, Chen Change Loy, and Timothy M. Hospedales. Self-supervised representation learning: Introduction, advances, and challenges. *IEEE Signal Processing Magazine*, 39(3):42–62, may 2022.

[8] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*, 2015.

[9] Ian J. Goodfellow. NIPS 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2017.

[10] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Advances in neural information processing systems*, 27, 2014.

[11] Samuel W Hasinoff. Photon, poisson noise. In Katsushi Ikeuchi, editor, *Computer Vision: A Reference Guide*, pages 608–610. Springer US, Boston, MA, 2014.

[12] Alain Horé and Djemel Ziou. Image quality metrics: Psnr vs. ssim. In *2010 20th International Conference on Pattern Recognition*, pages 2366–2369, 2010.

[13] Chiou-Ting Hsu and Ja-Ling Wu. Hidden digital watermarks in images. *IEEE Transactions on Image Processing*, 8(1):58–68, 1999.

[14] Rongqing Hui. Chapter 4 - photodetectors. In Rongqing Hui, editor, *Introduction to Fiber-Optic Communications*, pages 125–154. Academic Press, 2020.

[15] Mark J. Huiskes and Michael S. Lew. The mir flickr retrieval evaluation. In *MIR '08: Proceedings of the 2008 ACM International Conference on Multimedia Information Retrieval*, New York, NY, USA, 2008. ACM.

[16] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. *Advances in neural information processing systems*, 28, 2015.

[17] Sajjad Shaukat Jamal, Tariq Shah, Shabieh Farwa, and Muhammad Usman Khan. A new technique of frequency domain watermarking based on a local ring. *Wireless Networks*, 25(4):1491–1503, May 2019.

[18] Jaya Jeswani, Savita Rajput, Karan Umredkar, and Apurva Ware. Digital watermarking using machine learning. *IOSR Journal of Engineering*, 12, 2022.

[19] Varghese Paul Jobin Abraham. An imperceptible spatial domain color image watermarking scheme. *Journal of King Saud University - Computer and Information Sciences*, 2019.

[20] W.Clem Karl. 3.6 - regularization in image restoration and reconstruction. In AL BOVIK, editor, *Handbook of Image and Video Processing (Second Edition)*, Communications, Networking and Multimedia, pages 183–V. Academic Press, Burlington, second edition edition, 2005.

[21] Eero Kurimo, Leena Lepistö, Jarno Nikkanen, Juuso Grén, Iivari Kunttu, and Jorma Laaksonen. The effect of motion blur and signal noise on image quality in low light imaging. In Arnt-Børre Salberg, Jon Yngve Hardeberg, and Robert Jenssen, editors, *Image Analysis*, pages 81–90, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[22] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. arXiv, July 2017.

[23] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[24] Siddharth Misra and Yaokun Wu. Chapter 10 - machine learning assisted segmentation of scanning electron microscopy images of organic-rich shales with feature extraction and feature ranking. In Siddharth Misra, Hao Li, and Jiabo He, editors, *Machine Learning for Subsurface Characterization*, pages 289–314. Gulf Professional Publishing, 2020.

[25] Anguelos Nicolaou, Vincent Christlein, Edgar Riba, Jian Shi, Georg Vogeler, and Mathias Seuret. Tormentor: Deterministic dynamic-path, data augmentations with fractals. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2707–2711, 2022.

[26] Jim Nilsson and Tomas Akenine-Möller. Understanding ssim. *arXiv preprint arXiv:2006.13846*, 2020.

[27] A. M. Raid, W. M. Khedr, M. A. El-dosuky, and Wesam Ahmed. Jpeg image compression using discrete cosine transform - a survey. *arXiv preprint arXiv:1405.6147*, 2014.

[28] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241, 2015.

[29] Farhad Shadmand, Iurii Medvedev, and Nuno Gonçalves. Codeface: A deep learning printer-proof steganography for face portraits. *IEEE Access*, 9:167282–167291, 2021.

[30] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):60, July 2019.

[31] Matthew Tancik, Ben Mildenhall, and Ren Ng. Stegastamp: Invisible hyperlinks in physical photographs. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2117–2126, 2020.

[32] Hai Tao, Li Chongmin, Jasni Mohamad Zain, and Ahmed N. Abdalla. Robust image watermarking theories and techniques: A review. *Journal of Applied Research and Technology*, 12(1):122–138, 2014.

[33] J.R.G. Townshend, C.O. Justice, C. Gurney, and J. McManus. The impact of misregistration on change detection. *IEEE Transactions on Geoscience and Remote Sensing*, 30(5):1054–1060, 1992.

[34] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[35] Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, and Nong Sang. Bisenet: Bilateral segmentation network for real-time semantic segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 325–341, 2018.

[36] Ning Yu, Guilin Liu, Aysegul Dundar, Andrew Tao, Bryan Catanzaro, Larry Davis, and Mario Fritz. Dual contrastive loss and attention for gans. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6731–6742, 2022.

[37] Kevin Alex Zhang, Alfredo Cuesta-Infante, Lei Xu, and Kalyan Veeramachaneni. Steganogan: High capacity image steganography with gans. *arXiv preprint arXiv:1901.03892*, 2019.

[38] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018.

[39] Xin Zhong, Pei-Chi Huang, Spyridon Mastorakis, and Frank Y. Shih. An automated and robust image watermarking scheme based on deep neural networks. *IEEE Transactions on Multimedia*, 23:1951–1961, 2021.

[40] Jiren Zhu, Russell Kaplan, Justin Johnson, and Li Fei-Fei. Hidden: Hiding data with deep networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 657–672, 2018.

[41] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2020.

[42] Simone Zini, Alex Gomez-Villa, Marco Buzzelli, Bartłomiej Twardowski, Andrew D. Bagdanov, and Joost van de Weijer. Planckian jitter: countering the color-crippling effects of color jitter on self-supervised training, 2023.