Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea Magistrale in Informatica

Image-Specific Protection Against Manipulation

Relatore:
Prof. Giuseppe Lisanti

Presentata da:
Filippo Bartolucci

Sessione II
Anno Accademico 2022/2023

Abstract

The rapid development of Generative Models (GMs) in image synthesis poses challenges for accurately identifying manipulated images. Previous methods exploited a finite set of templates to proactively counter image manipulation, but this raises some concerns about potential vulnerabilities. A finite number of templates can provide a predictable exploit for malicious attackers, allowing them to reverse-engineer a template and evade detection by reapplying it to the modified image.

This work presents a template-based detection system that integrates transformer-based models to generate personalised templates for each image. Our approach enhances template protection while also improving the accuracy of manipulation detection. Furthermore, the generated template is employed to localize manipulated areas within an image, enabling the identification of specific regions that have been altered.

Our solution achieves high detection accuracy and robust localization performance on both trained attributes and previously unseen attributes, while also outperforming existing solutions.

Contents

List of Figures

List of Tables

1

Introduction

State-of-the-art generative models for image synthesis are constantly evolving, reshaping and revolutionizing the field of image creation. These models demonstrate their versatility across various domains, especially excelling in complex tasks like image-to-image translation and facilitating seamless transitions between different visual domains. However, the remarkable capability of these models to generate highly realistic images prompts a fundamental question: can we reliably determine whether an image has undergone manipulation?

Fake image generation can be broadly classified into two categories: complete image generation and partial image manipulation. In the former, entirely new images are created by inputting noise into a generative model. In contrast, the latter involves selectively modifying a real image. This partial manipulation can significantly alter the semantics of genuine images and pose risks to security and trust, particularly in the case of malicious exploitation.

With the increasing accessibility of generative models, the possibility of their abuse increases, resulting in various problems such as dissemination of misinformation and falsification of evidence. The importance of countering these models cannot be overstated.

One example of a counter tool is the use of detection models trained to distinguish manipulated images from real ones. Typically, they are trained by analyzing both real and manipulated images. Yet, their performance and

adaptability to new models or novel manipulations are limited, requiring them to relearn specific artifacts in their detection process each time.

Recent advancements in the literature have brought forth notable improvements in this challenging task through the introduction of a novel approach based on template protection. An undetectable signal, known as the template, is added to an image for encryption purposes. If the encrypted image undergoes manipulation through generative models, the template serves as verification to effectively distinguish between the encrypted image and its altered version. Manipulation of a protected image results in an image with a tampered template, effectively serving as an unambiguous indicator that the image has undergone modification. This enhancement significantly empowers the model, enabling more effective detection of various forms of manipulation with increased accuracy and reliability.

Despite these advancements, there are still some limitations due to the limited number of templates used by current techniques. Indeed, an attacker could reverse-engineer one of the templates used, exploiting the predictability of a finite set. They could then use the reversed template to authenticate manipulated images as real. This thesis aims to address this limitation by improving the template creation process. Our new approach involves creating personalized templates for each image, thereby strengthening template protection. Furthermore, the enhanced protection afforded by the use of templates will be leveraged for a secondary task: manipulation localization. This involves not only detecting manipulated images but also precisely identifying the regions within the image that have undergone manipulation. This dual-purpose approach contributes to a robust and comprehensive solution in the evolving landscape of image synthesis and manipulation detection.

The core technology enabling this work is the use of transformer-based models to generate a customized template for each image to be protected. We

operate within an unsupervised learning scenario, due to the absence of any ground truth template as a reference. Our goal is achieved by employing distinct loss functions that are specifically designed to enforce key characteristics throughout the training process.

In our study, we achieved a detection accuracy of 100% when tested on a trained attribute and an accuracy of 99.90% on previously unseen attributes. Moreover, the detection performance was robust, with a cosine similarity value of 0.95 for the trained attribute and between 0.80 and 0.86 for the unseen ones. Comparing our models with previous work, we outperformed in both tasks using the same training attribute.

2

Related Works

In this chapter, we will explore some of the most recent advancements in contrasting image manipulation orchestrated by GMs. These techniques can be broadly categorized into two groups: passive and proactive methodologies. In passive approaches, the model is restricted to analyzing the input image. Conversely, proactive techniques add imperceptible templates to detect and counteract GM manipulations.

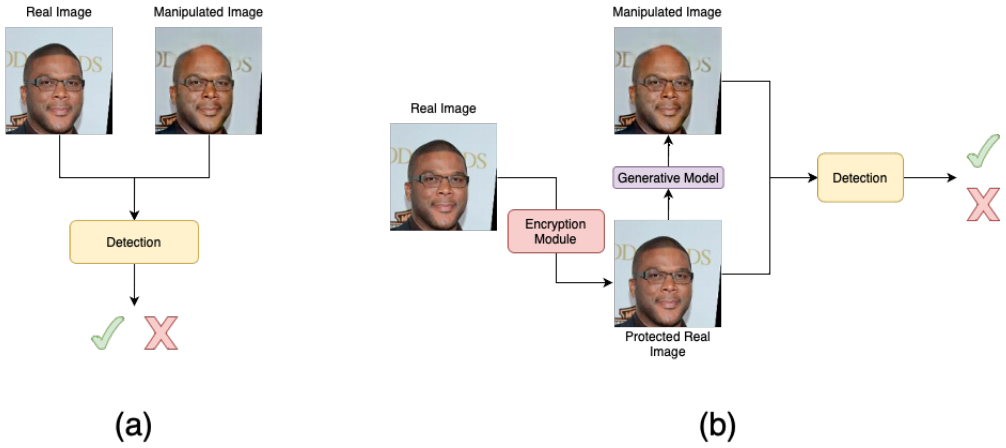

Figure 2.1: Comparison between passive (a) and proactive (b) approach

2.1 Passive Detection

In literature, there exists a range of techniques dedicated to the detection of image manipulation. These methods target various vulnerable aspects, such as mouth movement [1], steganalysis features [2], and attention mech-

anisms [3]. However, these methods are limited in their ability to analyze the entire photo, because they rely solely on artifacts left by GMs. In [4], the authors aim to uncover key characteristics that make manipulated images detectable. They employ shallow networks with limited receptive fields to pinpoint specific manipulated areas within the image, ultimately arriving at an overall prediction. This approach prioritizes local details over global image semantics, highlighting the significance of focusing on specific regions for effective manipulation detection.

These networks with limited receptive fields are used as patch-based classifiers to provide predictions on whether each patch is real or fake. These predictions are then used to generate heatmaps, visually indicating areas more likely to be genuine or manipulated. This approach has demonstrated some adaptability across a spectrum of model parameters, generator architectures, and datasets, but identifying differences between real and fake images is an evolving challenge, sensitive to even minor preprocessing nuances. The key insight gained from this is that, in the pursuit of effective manipulation detection, prioritizing local details holds greater significance than focusing solely on global image semantics.

2.2 Proactive Approach

Expanding the defense strategy beyond passive analysis, and instead adopting a proactive approach, opens up novel possibilities for effectively dealing with image manipulation. In addition to detection methods [5] [6], research has also explored techniques like deepfake disruption [7] and tagging [8].

2.2.1 Disruption

In [7], a proactive defense technique involving deliberate disruption of the GM's output is examined. This technique introduces imperceptible modifications to an image-alterations that escape human visual detection. However,

when employed within a GM, these subtle adjustments render the GM output unusable. The approach is based on the understanding that neural networks are vulnerable to adversarial attacks [9]. The goal is to create a disruption by introducing a human-imperceptible perturbation η to the input image x resulting in the disrupted input image $\tilde{x}$. Given a generative model G, $\tilde{y}$ and y represent the output images obtained from $G(\tilde{x})$ and $G(x)$ respectively. A successful disruption is one that introduces noticeable corruptions to $\tilde{y}$, indicating tampering, while preserving the original appearance of the image $\tilde{x}$.

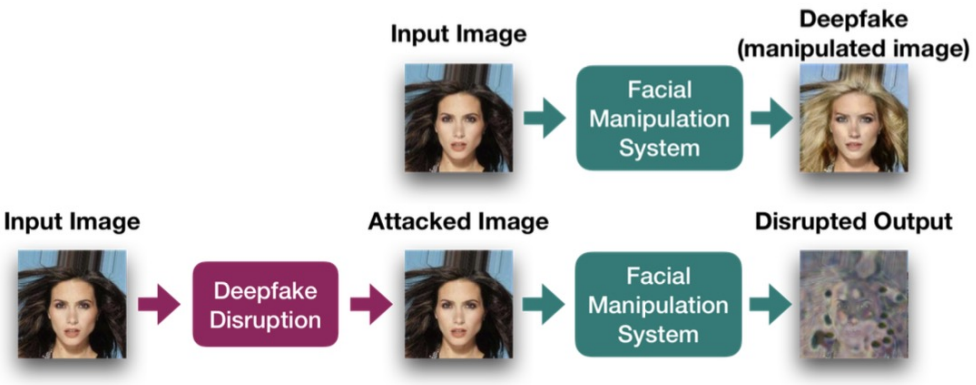

Figure 2.2: Example of disruption

[7] explores various established adversarial attack techniques, including Fast Gradient Sign Method [9], Iterative Fast Gradient Sign Method, and Projected Gradient Descent. It is observed that a disruption for the data/class pair (x, c_i) is not guaranteed to transfer to the data/class pair (x, c_j) when $i \neq j$. That's why [7] introduces two disruption techniques, Iterative Class Transferable Disruption and Joint Class Transferable Disruption, both modified versions of Iterative Fast Gradient Sign Method, capable of transferring disruption across different classes in a class-conditional GM. The results demonstrate its effectiveness in disrupting prominent image translation architectures (StarGAN, pix2pixHD, and CycleGAN) effectively highlighting

alterations in the images and ensuring no unnoticed manipulations.

The limitation of this work is rooted in its operation in a "gray-box scenario" and "white-box scenario" where researchers have full knowledge of the model and its parameters.
The limitation of this study lies in its "gray-box" and "white box" settings. In the gray-box scenario, researchers possess comprehensive knowledge of the model and its parameters. In the white-box scenario, this knowledge extends to include an understanding of the defense mechanisms in place, such as blurring. This, however, does not align with typical real-world scenarios where such comprehensive information is not readily available.

2.2.2 Image Tagging

In [8], a new defense technique against deepfakes based on image tagging is presented. Image tagging refers to a technique that allows the concealment of a hidden string within an image, rendering it imperceptible to the naked eye. This technique can provide crucial insights for tracing the origin of a deepfake. An illustrative example could involve empowering social networks to swiftly block and track images by checking tags. The authors introduce a novel tagging methodology named "FakeTagger" which leverages image tagging for deepfake forensic analysis and tracking. The process involves a five-component pipeline: Message Generator, DNN-Based Encoder, GAN Simulator, DNN-Based Message Decoder, and Channel Decoder.

The Message Generator produces a redundant message X', deliberately duplicated to account for potential loss during manipulation. The DNN-Based Encoder then seamlessly incorporates X' into a facial image. The GAN Simulator manipulates the encoded images, and the Decoder aims to retrieve the embedded message from the manipulated image. The Channel Decoder processes the decoded message, producing the definitive message X. The jointly trained DNN-based encoder and decoder play complementary roles in

the process. The encoder embeds messages into facial images, ensuring perceptual similarity, while the decoder retrieves the embedded message after GAN-based transformations.

To ensure minimal error in the decoded message, two major losses were employed in model training: Message Loss and Image Loss. FakeTagger was tested on a dataset of 30,000 facial images from the CelebA-HQ dataset, demonstrating superior performance compared to a baseline approach. The study highlights the effectiveness of FakeTagger in both white-box and black-box settings for message retrieval across various DeepFake transformations. Additionally, it emphasizes the significance of larger input image sizes and reduced region manipulation. FakeTagger is also resilient against common image perturbations encountered in DeepFake video production, such as compression, resizing, blurring, and Gaussian noise. The results highlight FakeTagger's effectiveness in countering various forms of DeepFake manipulation.

2.2.3 Detection

In [5], a proactive defense approach for addressing image manipulation detection is introduced. Compared to traditional passive algorithms, this method focuses on estimating a set of templates that, when applied to real images, enhance the accuracy of manipulation detection. This template addition effectively serves as an encryption process, providing a means to distinguish between genuine and manipulated versions of an image. When these encrypted images undergo manipulation by a GM, the algorithm can recover the added template, enabling differentiation between the encrypted and manipulated versions. However, learning a set of templates presents challenges due to the absence of an established ground truth template for supervision, as well as the complexities of recovering the template from manipulated images.

The framework comprises two stages: image encryption and template recovery. The encryption stage involves the selection and addition of tem-

plates, while the recovery stage focuses on retrieving the templates from both encrypted real and manipulated images. Extensive evaluations were conducted on images processed by unfamiliar GMs, demonstrating significant improvements in performance compared to a pretrained model from a previous study [10].

2.2.4 Localization

In [6], the authors present Manipulation Localization Using a Proactive Scheme (MaLP), an advancement beyond their earlier work [5]. This time, the focus extends beyond detection to include the introduction of manipulation localization achieved through the use of templates. While recent approaches in manipulation localization primarily operate passively by estimating manipulation masks in face-swapped images, MaLP [6] takes a proactive approach. It employs an optimized template to enhance manipulation localization in real images, addressing potential alterations by unseen GMs. The framework includes encryption, detection, and localization modules for processing encrypted images, enabling effective localization and detection of manipulations.

To optimize the template set in the encryption module, criteria such as low magnitude, orthogonality, and high-frequency content are enforced as constraints. The localization module utilizes a two-branch architecture that combines a shallow CNN network for high inference efficiency with a ViT transformer to capture global information. The fakeness map estimation involves maximizing cosine similarity and structural similarity index measure for fake images while ensuring a zero map for encrypted images.

The proposed framework is equipped with distinct encryption, localization, and detection modules, each with its respective loss function. Overall, an image is encrypted by adding a template from the set, and the localization module generates a fakeness map. This map is then used by the detection

module to determine whether the image has been tampered with. MaLP exhibits superior performance in localization and binary detection compared to passive methods, demonstrating its adaptability to unseen attribute modifications.

Nevertheless, both Malp and [5] face limitations due to their reliance on a fixed-size template set. This characteristic exposes them to potential vulnerabilities from reverse engineering attempts on the templates.

In Table 2.1, is presented a summary of the previously discussed related works.

Paper	**Scenario**	**Category**	**Used Techniques**
[4]	Passive	Detection	Patch-based Classification
[7]	Proactive	Disruption	FGSM, I-FGSM, PGD
[8]	Proactive	Tagging	U-Net Based Message Encoder and Decoder
[5]	Proactive	Detection	Set of learnable templates
[6]	Proactive	Detection and localization	Set of learnable templates

Table 2.1: Summary of related works

3

Method

In this chapter, we will present the architectural design of our framework and provide insights into our implementation decisions.

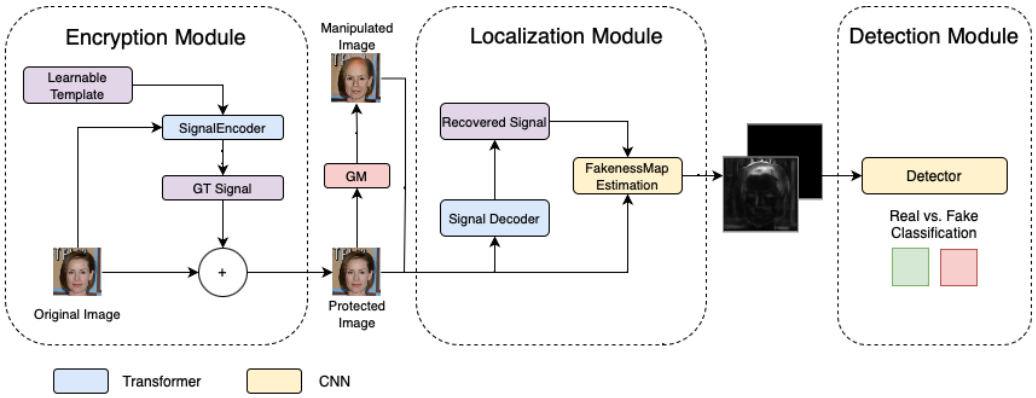

Figure 3.1: Our Architecture

The objective of this framework is to detect and localize manipulation of an image using a proactive approach. While different techniques in the literature address countering GMs manipulation, such as deepfake disruption and tagging, we have chosen to focus on detection and localization. We believe this approach is the most effective way to combat manipulation, as also shown in the literature.

The proposed architecture consists of three primary modules:

1. **Encryption Module**: This module is responsible for encrypting real images.

2. **Localization Module**: Module designed to estimate a map that highlights manipulated areas within images.

3. **Detection Module**: Binary detection for the encrypted and manipulated images.

3.1 A new template protection

MaLP [6] demonstrated the efficacy of template-based image safeguarding for detecting and localizing image manipulation. However, a notable limitation lies in its use of a fixed set of templates, which represents a potential vulnerability. An attacker could reverse engineer one of the predetermined templates and exploit the system's predictability to manipulate images and authenticate them as real using the reversed template.

To address this concern, our work focus on refining the template protection mechanism. We leverage transformer models renowned for their outstanding performance in various computer vision tasks (refer to Appendix A.2). Our aim is to enhance template protection by transitioning from a finite template set to a per-image customized template approach.

In this advancement, we utilize a transformer model with learnable parameters as source template. This template is then seamlessly integrated with the image data. The outcome is a new uniquely tailored template for each specific image, ensuring a diverse and customized level of protection for every individual image. This novel approach strengthens the system's resilience against protection counterattacks by introducing variability into the template creation process, thus mitigating the predictability associated with a fixed set of templates.

Another consideration in our work pertains to the operational aspects of vision transformers. When a generative model modifies an image, only spe-

cific portions undergo alteration. The inherent patch-by-patch functionality of transformer models makes them more intuitive for both generating and analyzing templates. When an image is manipulated using a patch-based template, only the patches within the manipulated area of the image are being compromised, meaning that it should be easier to locate the manipulated area in the image by searching for the manipulated patch. This strategic application should optimize the performance of our process, ensuring an effective safeguarding mechanism that works well, especially in accommodating local changes within images.

3.2 Encryption Module

The initial stage in our architecture involves the encryption module, which is responsible for safeguarding an image by incorporating an imperceptible signal into it. This process is characterized by the application of a transformation denoted as τ to a real image I^R, resulting in the generation of the encrypted image $\tau(I^R)$.

The fundamental concept here is that the signal applied by the transformation τ serves as a distinctive signature, enabling the verification of whether an image has undergone any form of manipulation. In our framework, this transformation is executed by adding to an image a special signal generated by a transformer model.

The challenge of training a model to compute a meaningful signal is nontrivial, given the absence of a definitive ground truth transformation for comparison. This implies that the entire training process occurs in an unsupervised manner, emphasizing the necessity for a set of custom losses, designed for each module, that play a crucial role in effectively steering the process toward our desired outcome. Defining effective losses for unsupervised training can

be challenging. The model tends to minimize the loss in unpredictable ways, leading to unexpected results.

To eliminate ambiguity, we will specifically use the term "template" to denote the learnable parameters inside the Encryption Module model. Additionally, we will use the term "Image-Specific Signal" $(s_{IS}) \in \mathbb{R}^{c \times m \times n}$ to represent the original signal introduced during the encryption process. Signal Decoder $\mathcal{S}_E$ is the transformer model used to generate the s_{IS}. One important factor is that s_{IS} needs to be optimized such that, it does not introduce any noticeable visual artifact in the original image. This optimization is essential to prevent any reduction in image quality, guaranteeing that the final images remain usable.

The encryption process is defined as follows:

$$\tau(I^R) = I^R + m \times \mathcal{S}_E(I^R) \tag{3.1}$$

where m is a fixed parameter used to control the strength of the added signal.

The Signal Decoder $\mathcal{S}_E$ model is based on a visual transformer architecture. As with all Visual Transformers, the model works by patching images, so the first layer of the model reshapes the images into patches and projects them linearly into the inner dimension of the model. Inside the model is the learnable template, which is used as the basis for the generation of s_{IS}. The dimensions of the template match the dimensions of the linearly projected images.

The model is made of a series transformer block which consists of a self-attention layer, a cross-attention layer, and a feed-forward. The input to the self-attention layer is the template, which is used as a query, key and value in the calculation of attention. During self-attention, the images do not contribute to the generation of the signal. The output of this layer is used as input of the cross-attention layer and here the image patches are

used as context for cross-attention. This means that the template processed by self-attention is used as a query, while the image patches are key and value. In this way, it is calculated how much importance each element in the key (image patches) should attach to the corresponding element in the query (template). The result, after a sum weighted with the value (again the image patches), is the output of the cross-attention.

In this process, the cross-attention mechanism assesses the relevance of different elements within the template by considering the content of the image. This weighted attention ensures that the template adapts dynamically to the characteristics of the image, providing a distinctive and tailored form of protection.

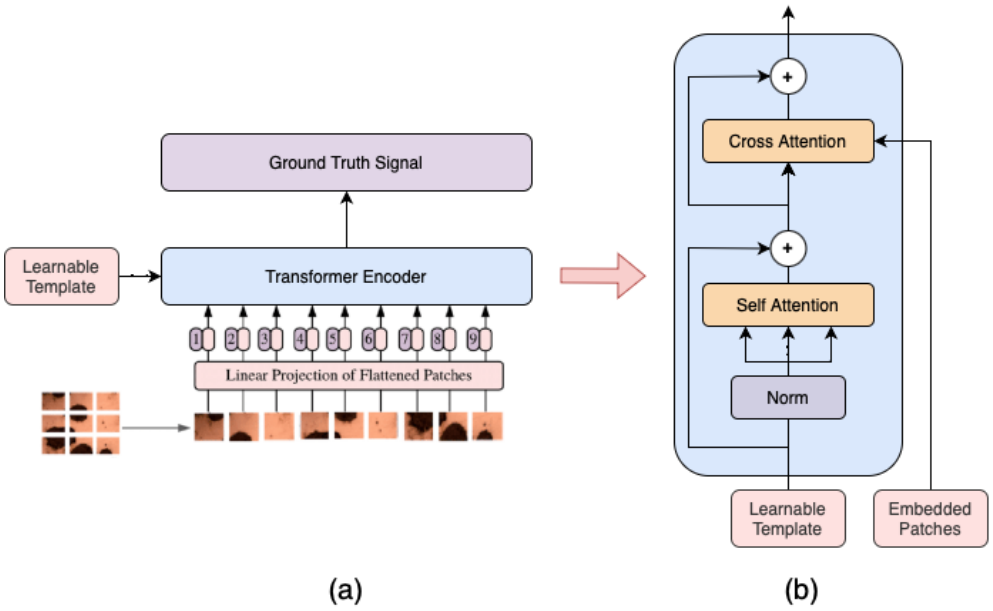

Figure 3.2: Architecture of Signal Encoder(a) and Transformer Block (b)

The output generated by $\mathcal{S}_E$, denoted as s_{IS}, is added to the image. Prior to this addition, the signal is multiplied by a constant strength weighting factor. In order to guide the training of the encryption module, we employ a loss function defined as follows::

$$\mathcal{L}_1 = \lambda_1 \times \|s_{IS}\|_2 \tag{3.2}$$

Here, $\lambda 1$ serves as a hyperparameter that determines the relative importance of this particular loss. This is essential as we will ultimately have multiple losses, each with varying degrees of significance. This loss function is designed to minimize the computed signal s_{IS}, which is crucial for ensuring that the protected image closely resembles the original. If the magnitude of the generated signal is too strong the image quality might be compromised.

While this loss alone may not be sufficient to render the s_{GT} meaningful, as the only constraint imposed is related to the signal's magnitude, it becomes more significant when integrated with the subsequent Localization Module architecture and losses.

3.3 Localization Module

This module is responsible for recovering the signal from protected images and estimating a map indicating potential manipulated areas, referred to as the "Fakeness Map" (see Figure 3.3). In contrast to certain prior approaches that rely on a threshold [11] to assess the difference between real and manipulated images (an undesirable approach since threshold selection is highly subjective and sensitive), we choose to adopt the definition of "Fakeness Map" derived in MaLP [6]. This definition involves utilizing a continuous grayscale map to compute the ground truth fakeness map.

Figure 3.3: Example of a fakeness map

Given the encryption transformation τ, the generative model $\mathcal{G}$, and the real image I^R, we define the ground truth fakeness map for the manipulated image $\mathcal{G}(\tau(I^R))$ as:

$$M_{GT} = Gray(|I^R - G(\tau(I^R))|)/255 \tag{3.3}$$

where $Gray()$ is a function that converts a colored image to grayscale.

The objective of this module is to extract the added signal from an input image, which could be either an original protected image or a modified version of it. When provided with an unaltered image, its objective is to precisely recover a distinctive signal closely resembling the original encryption signal S_{IS} that was encrypted with. Instead, when a manipulated image is given as input, the module aims to retrieve a signal that is as diverse as possible from the authentic signal, facilitating easy differentiation. The ability to differentiate and recognize the two types of signal means that the model is able to recognize when an image has been manipulated or not, which makes the subsequent detection task easier.

This approach is chosen to maximize the distinction between the signal of a genuine image and that of an altered image. If a portion of the image has been altered, the signal in the manipulated image should be tampered with

in such a way that the model should not be capable of recovering it. Meanwhile, an unaltered image should strive to preserve as much of the original added signal as possible. The signal obtained from a protected image will be referred to as real signal s_R whereas the signal extracted from a manipulated protected image will be called fake signal s_F

In the MaLP framework [6], the Localization module for the encoder relied on a basic CNN architecture that bifurcated into two branches: one processes the signal while the other generates the fakeness map. However, in our framework, this approach is no longer adequate. (See Appendix A.3).

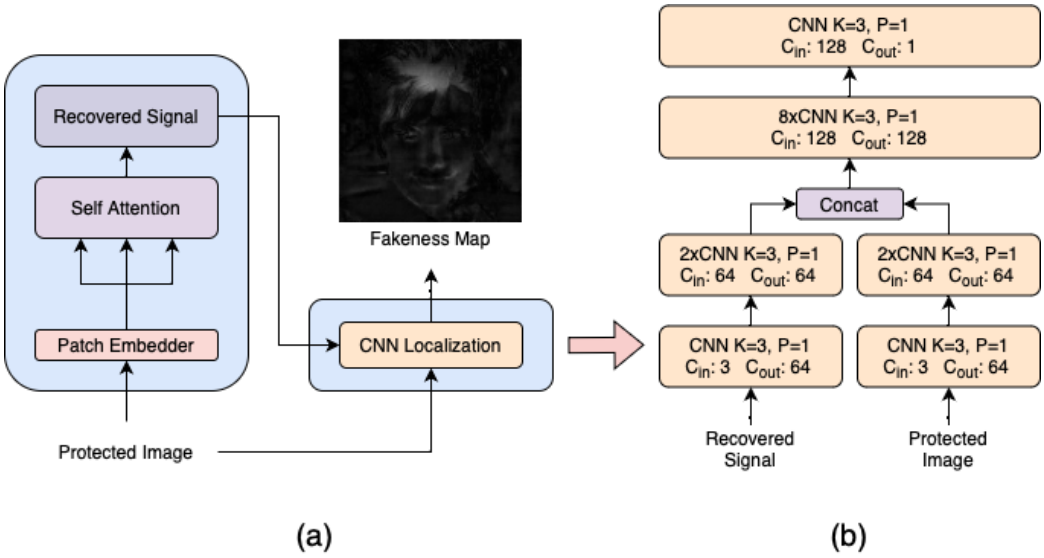

Figure 3.4: Architecture of Localization Module (a) and CNN branch (b)

Since the s_{IS} is now computed and applied per patch, a CNN alone falls short in capturing the nuances of per-patch signal characteristics. To address this, we chose to develop a transformer-based decoder, leveraging self-attention layers, specifically designed for signal recovery.

Despite opting for a transformer-based approach for signal recovery, our module still incorporates a parallel CNN branch. Our experiments demonstrated

that relying solely on the transformer for estimating the fakeness map yielded unconvincing results, as the generated fakeness maps were mere patches of noise. For this reason, we introduced an additional, dedicated CNN branch specifically designed for the fakeness map estimation task. The CNN combines the recovered signal from an image with the protected image. Indeed, during the fakeness map estimation process, the recovered signal provides essential information to help the model differentiate between authentic, unaltered images and manipulated ones. This dual-input approach enhances the module's capacity to make accurate predictions, thereby contributing to the overall effectiveness of the framework in localizing image alterations.

In the training process, we employ the actual signal generated by the encryption module and the fakeness map, as defined in Equation 3.3. This map is computed using both the unaltered image and a modified image generated by STGAN [12], a reference generative model, which serves as the basis for supervision.

As for the encryption module, we have defined specific loss for this model. Our goal is for the real signal to approximate s_{IS}, while conversely, we strive for the fake signal to deviate as much as possible from s_{IS}.

Regarding the fakeness map, our aim is for the estimated fakeness of manipulated images to closely correspond with the ground truth fakeness map. While, for authentic protected images that have not undergone alterations, we aim for the fakeness map to be as close to zero as possible, indicating that the model does not detect any modified regions.

We will now introduce the notation used in the equations for the loss functions. Let I represent the protected input image. When we apply the localization module, referred to as Signal Decoder $\mathcal{S}_D$, to the input image I, it

produces s_{Real} and the corresponding localization map M_{Real}:

$$s_{Real}, M_{Real} = \mathcal{S}_D(I) \tag{3.4}$$

Similarly, for the manipulated image $\mathcal{G}(I)$ generated by the generative model $\mathcal{G}$, $\mathcal{S}_D$ produces s_{Fake} along with its associated localization map M_{Fake}:

$$s_{Fake}, M_{Fake} = \mathcal{S}_D(\mathcal{G}(I)) \tag{3.5}$$

For this module, we have six loss functions that aim to maintain the characteristics of the signal and fakeness map, as previously outlined.

Signal Losses: These losses play a crucial role in ensuring the adherence of the recovered signal to specified properties. Here, CS represents the cosine similarity function (see Appendix A.4). In Equation 3.6, minimizing the CS allows us to reduce the similarity between s_{Real} and s_{Fake}, making them as discernible as possible.

$$\mathcal{L}_2 = \lambda_2 \times CS(s_{Real}, s_{Fake})) \tag{3.6}$$

In contrast, Equation 3.8 serves the purpose of ensuring that s_{IS} for distinct images exhibits variation within the batch. This particular loss is crucial because our experiments revealed a tendency of the $\mathcal{S}_E$ to neglect the contribution of input images over time. This behavior stems from the fact that the $\mathcal{S}_E$ has only one loss, which doesn't impose any constraints on the signal's form apart from its magnitude.

Meanwhile, the $\mathcal{S}_D$ has losses associated with recovering the signal and fakeness map. During training, this module forces the $\mathcal{S}_E$ to discard the contribution of the input images, as, in this case, it still minimizes all the losses while facilitating its own task of retrieving the signal and fakeness map. The consequence of this was that the $\mathcal{S}_E$ started generating a fixed signal for all images, rendering the model essentially useless.

$$\mathcal{L}_3 = \lambda_3 \times \sum_{i=1}^{N} \sum_{j=i+1}^{N} CS(s_{IS}[i], s_{IS}[j]) \tag{3.7}$$

To fix this issue, we introduced the new loss reported in Equation 3.7. This loss is designed to enforce diversity among signals across different images. Specifically, it achieves this by computing the cosine similarity between s_{IS} representations for pairs of images within the batch, thereby ensuring that the signals evolve distinctly over various instances in the dataset.

Minimizing the cosine similarity alone proved insufficient. With cosine similarity values ranging from 1 (indicating the same direction) to -1 (indicating opposite directions), and 0 representing orthogonality, relying solely on cosine similarity, without clamping values below zero, led the model to learn only two distinct signal types. Remarkably, these signals had a cosine similarity of -1, meaning that they were opposite signals. This outcome occurred because learning only two opposite signals was the most straightforward way for the model to minimize the loss.

$$\mathcal{L}_3 = \lambda_3 \times \sum_{i=1}^{N} \sum_{j=i+1}^{N} Max(CS(s_{IS}[i], s_{IS}[j]), 0) \tag{3.8}$$

When calculating the loss in equation 3.8 within a batch containing these two signals, the mean cosine similarity within the batch tended to approach zero. This was due to the compensatory effect of the values of 1 for one signal being balanced by the values of -1 for the other signal in the batch. By applying a clamp (as in Equation3.8) to restrict negative values to 0, we effectively eliminated the contribution to the loss from signals with negative directions. This forced all signals to be orthogonal to each other as the only way to minimize this loss.

Map Losses: These losses are employed to enforce properties for fakeness map estimation. Equations 3.9 and 3.10 ensure that the estimation of

$MapReal$ is as close as possible to a black image (indicative of no changed pixels in the image) and that Map_{Real} and Map_{GT} are as dissimilar as possible.

$$\mathcal{L}_4 = \lambda_4 \times ||M_{Real}||_2 \tag{3.9}$$

$$\mathcal{L}_5 = \lambda_5 \times CS(M_{GT}, M_{Real}) \tag{3.10}$$

Equations 3.11 and 3.12 try to maximize the similarity between Map_{Fake} and Map_{GT}. The structural similarity loss (appendix A.5) is denoted as $SSIM$.

$$\mathcal{L}_6 = \lambda_6 \times (1 - CS(M_{GT}, M_{Fake})) \tag{3.11}$$

$$\mathcal{L}_7 = \lambda_7 \times (1 - SSIM(M_{GT}, M_{Fake}) \tag{3.12}$$

3.4 Detection Module

The final component of the framework is the detection module, responsible for determining whether an image has been manipulated. Given the defined loss functions, this task is now relatively straightforward. Thanks to the map losses, we have clear distinctions between M_{Fake}, which resembles M_{GT}, and M_{Real}, which is designed to be as close to a completely black image as possible.

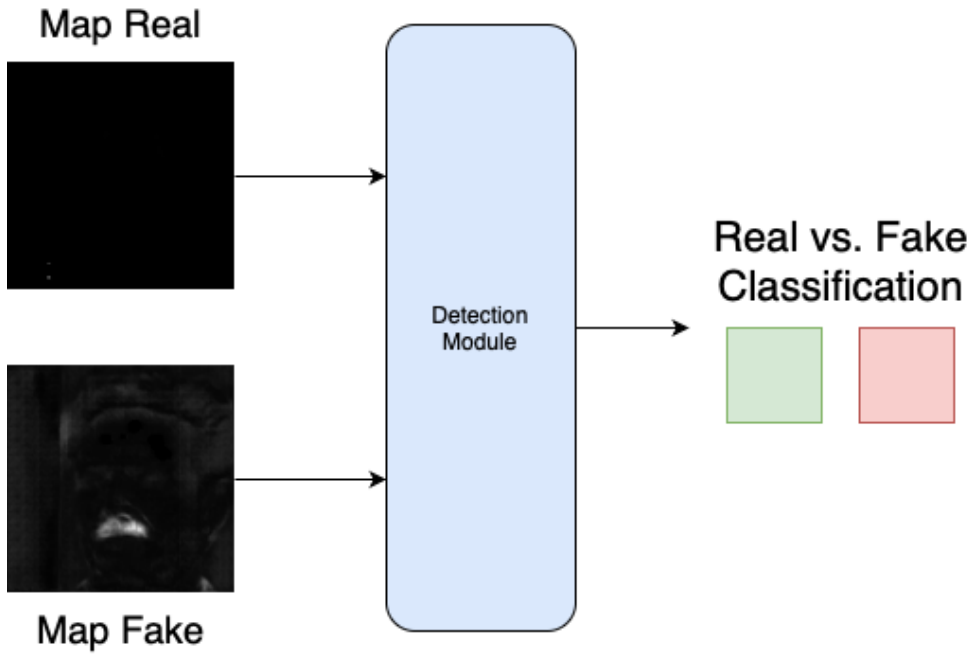

Figure 3.5: Detection Module

This distinction is not only achieved through the fakeness map, but also due to the added signal incorporated in all the previous modules. Ideally, the localization module should leverage the recovered signal to enhance the differentiation between real and manipulated images. When it successfully recovers a signal indicative of a real image, its contribution should transform the fake map into a completely black representation.

With this understanding, it's clear that the final detection task is simpler compared to passive detection methods, where the model must learn to recognize imperfections left by the generative model.

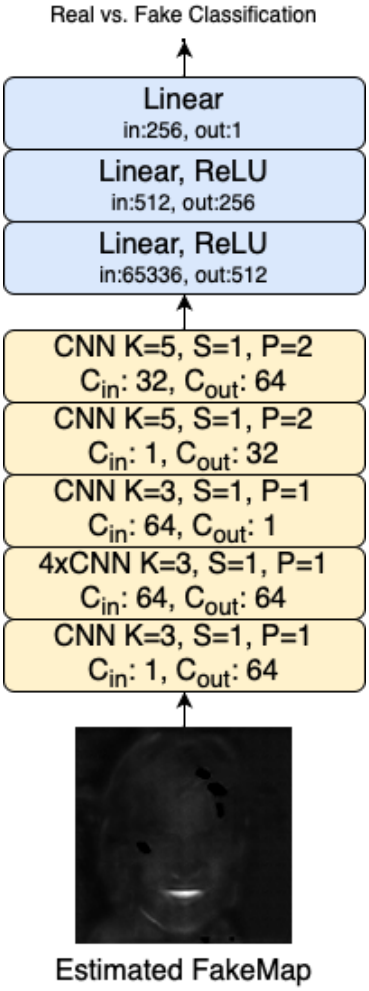

Figure 3.6: Architecture of Detection Module

We chose to retain the same detection architecture as MaLP [6], as we deemed the specific design of this module less critical compared to others. This architecture was still considered sufficient for the task. This detection module receives an estimated fakeness map from the localization module as input and produces a prediction indicating whether the image is real or manipulated.

Therefore, the detection model utilizes a simple classification loss, such as the binary cross entropy, defined as:

$$\mathcal{L}_8 = \lambda_8 \times \sum_{i=1}^{N} [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)] \tag{3.13}$$

where:

- y_i is the true label for sample i (0 or 1)
- $\hat{y}_i$ is the predicted probability that the sample i belongs to class 1.

All modules are trained collectively in an end-to-end manner, with the total loss computed as the sum of individual components:

$$\mathcal{L} = \mathcal{L}_1 + \mathcal{L}_2 + \mathcal{L}_3 + \mathcal{L}_4 + \mathcal{L}_5 + \mathcal{L}_6 + \mathcal{L}_7 + \mathcal{L}_8 \tag{3.14}$$

As a side note, during the initial epochs of training, the estimated fakeness maps are not well-defined. Consequently, the training of the detection module begins after a warm-up period for the other modules. This warm-up spans the first 50 iterations. After that, the entire model is trained end-to-end.

4

Experiments

In the following chapter, we will present all the experiments conducted and their results.

4.1 Settings

In our experiments, we used the CelebA dataset, comprising 202,599 images featuring celebrity faces. For training, we employed the initial 182,000 images, while images numbered from 182,001 to 182,637 were allocated for validation. The remaining 19,962 images were used in the test set. All faces in the dataset were cropped, aligned, and resized to a resolution of 128x128.

All experiments were performed on an RTX 3090 with 24GB of VRAM with a batch size of 32.

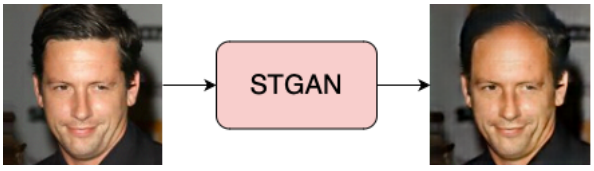

Figure 4.1: Example of STGAN Manipulation of facial attribute "Bald"

We opted to employ STGAN as our generative model, a choice aligned with MaLP [6]. This decision was made in order to establish a meaningful baseline for comparison. STGAN is a pre-trained GM for the task of facial attribute

manipulation, i.e., it allows the manipulation of specific facial features. In our experiments, we will clearly specify both the facial attribute on which the model was trained and the attribute on which it was tested.

These values have been selected following the settings from MaLP [6]. 4.1.

λ_1	λ_2	λ_3	λ_4	λ_5	λ_6	λ_7	λ_8
80	20	2	25	25	25	50	70

Table 4.1: Loss hyperparameters

4.2 Evaluation Metrics

We will use specific metrics tailored to each task for evaluation. To assess binary detection, we will employ the accuracy score. For the Localization task, we will measure the similarity between the generated map and the ground truth map using cosine similarity. To quantify the degree of image alteration following the addition of the signal s_{IS}, we will use two image distance metrics: Learned perceptual image patch similarity (LPIPS) and mean squared error (MSE). We selected these two metrics for evaluating image quality because they measure image degradation from different perspectives. MSE measures pixel distance, while LPIPS measures the difference in perceptual similarity, capturing how we perceive an image.

4.3 Transformer Depth

We will begin by presenting results on our model, specifically examining the impact of varying transformer depths in both Signal Encoder $\mathcal{S}_E$ and Signal Decoder $\mathcal{S}_D$. Our goal is to determine the optimal model in terms of encoder-decoder architectural depth and how varying the complexity of the transformer-based models can affect signal generation.

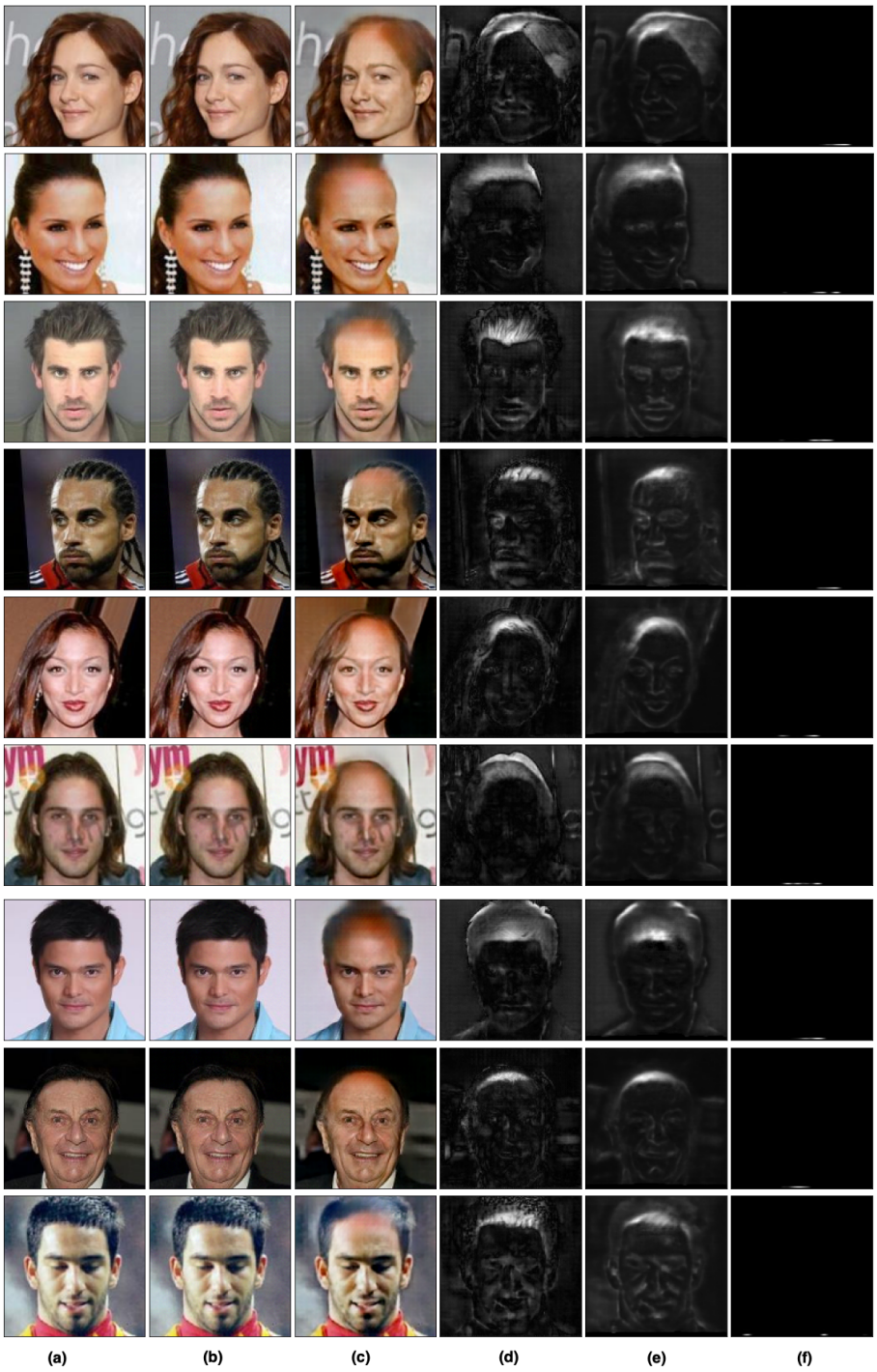

Figure 4.2: Results Visualization: (a) original image, (b) encrypted image, (c) manipulated image, (d) ground truth fake map, (e) estimated fake map for the manipulated image, (f) estimated fake map for the protected image.

Across our experiments, we explored encoders and decoders with depths of transformer layers 1,3 and 6 maintaining the same depth values for both models. During both training and testing, we utilized "Bald" as the attribute for STGAN manipulation.

Depth	**Fakeness Map CS**	**Accuracy**
1	0.951839	0.999975
3	0.950642	0.999975
6	0.951260	1.0

Table 4.2: Performance with different depth for Encoder/Decoder

The results presented in Table 4.2 indicate that all models exhibit nearly identical performance in both localization and detection tasks. This can be attributed to the shared CNN architecture among all models. Similarly, the consistency in detection accuracy arises from the unchanged detection architecture across all experiments. In Figure 4.2 are shown some samples of outputs generated by the depth 6 model.

Even with a depth of 1, the models can easily detect the added signal. However, the effect of varying the depth of the transformer block becomes clear in the signal generation and recovery process. The key difference is in the type of signal generated. The following figures illustrate samples of generated signals for each model depth: Figure 4.3 for depth 1, Figure 4.4 for depth 3 and Figure 4.5 for depth 6.

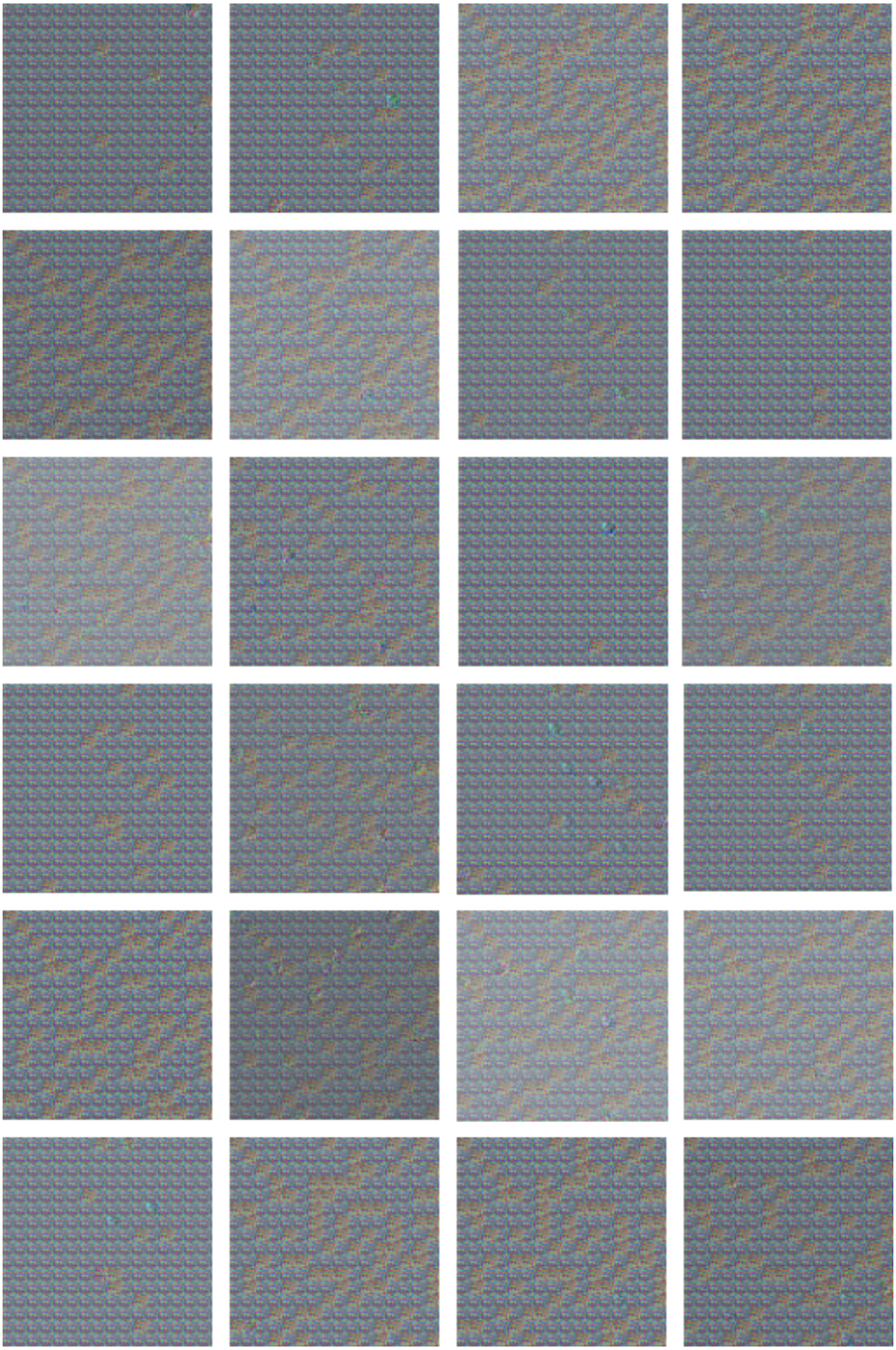

Figure 4.3: Signals generated by an encoder with depth 1

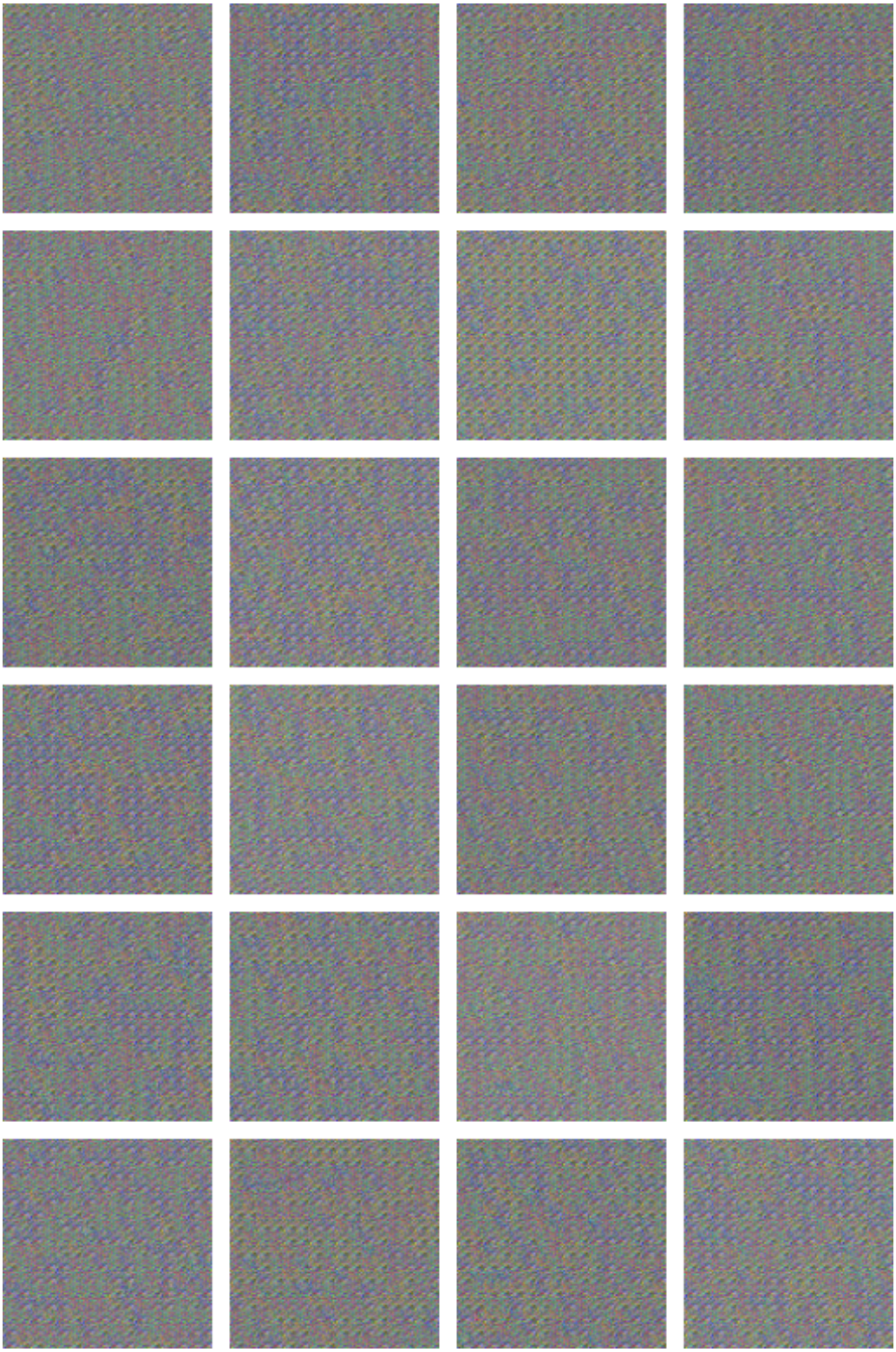

Figure 4.4: Signals generated by an encoder with depth 3

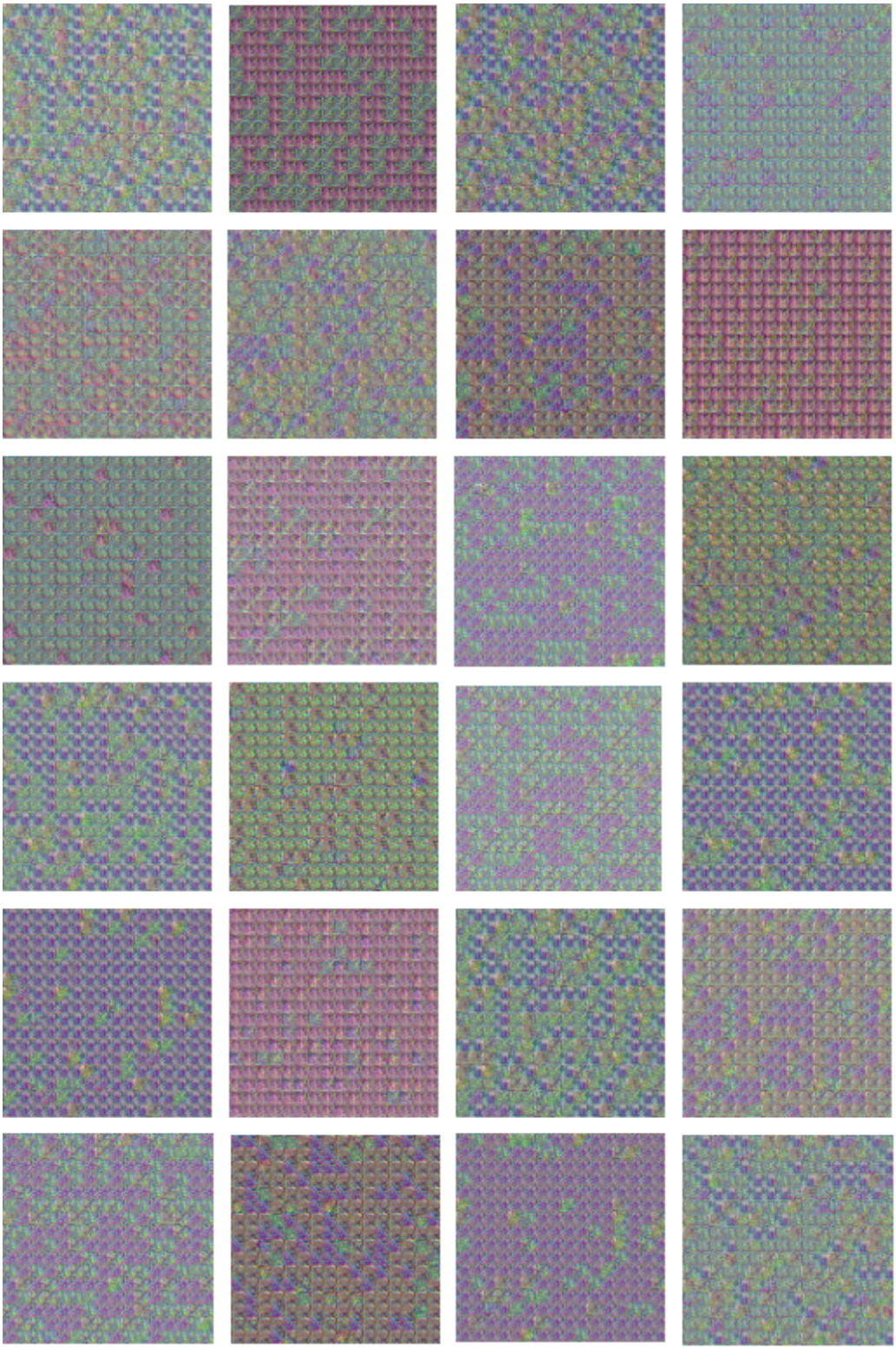

Figure 4.5: Signals generated by an encoder with depth 6

From these figures, it is possible to appreciate that deeper models create more complex and diverse signals across multiple images. Such complexity arises from the greater number of layers in deeper models, which enables more complex signal generation. All signals exhibit a similar patch-based aesthetic, resulting from the use of a vision transformer. The signal generated by the depth 1 model has a simple and less diverse patch pattern compared to bigger models like 3 and 6. Although a small model may achieve high performance in both tasks, we cannot expect it to provide the same degree of protection as with larger models.

4.4 Image Degradation

In this section, we will show how much an image is degraded by the addition of s_{IS}. A too strong and distinguishable signal would render the images unusable, so this aspect is crucial in order to be able to use this framework in real-world contexts. Table 4.3 presents the MSE and LPIPS values for different tested models.

Model	**MSE**	**LPIPS**
Depth 1	0.000105	0.000353
Depth 3	0.000105	0.001034
Depth 6	0.000101	0.001894

Table 4.3: Image degradation results

Regardless of the depth, the table results indicate that all models generate a signal with minimal effect on image quality, both in terms of pixel level (MSE) and perception level (LPIPS). This outcome is achieved by using the parameter m to control the magnitude of the signal in combination with the loss L_1 (Equation 3.2).

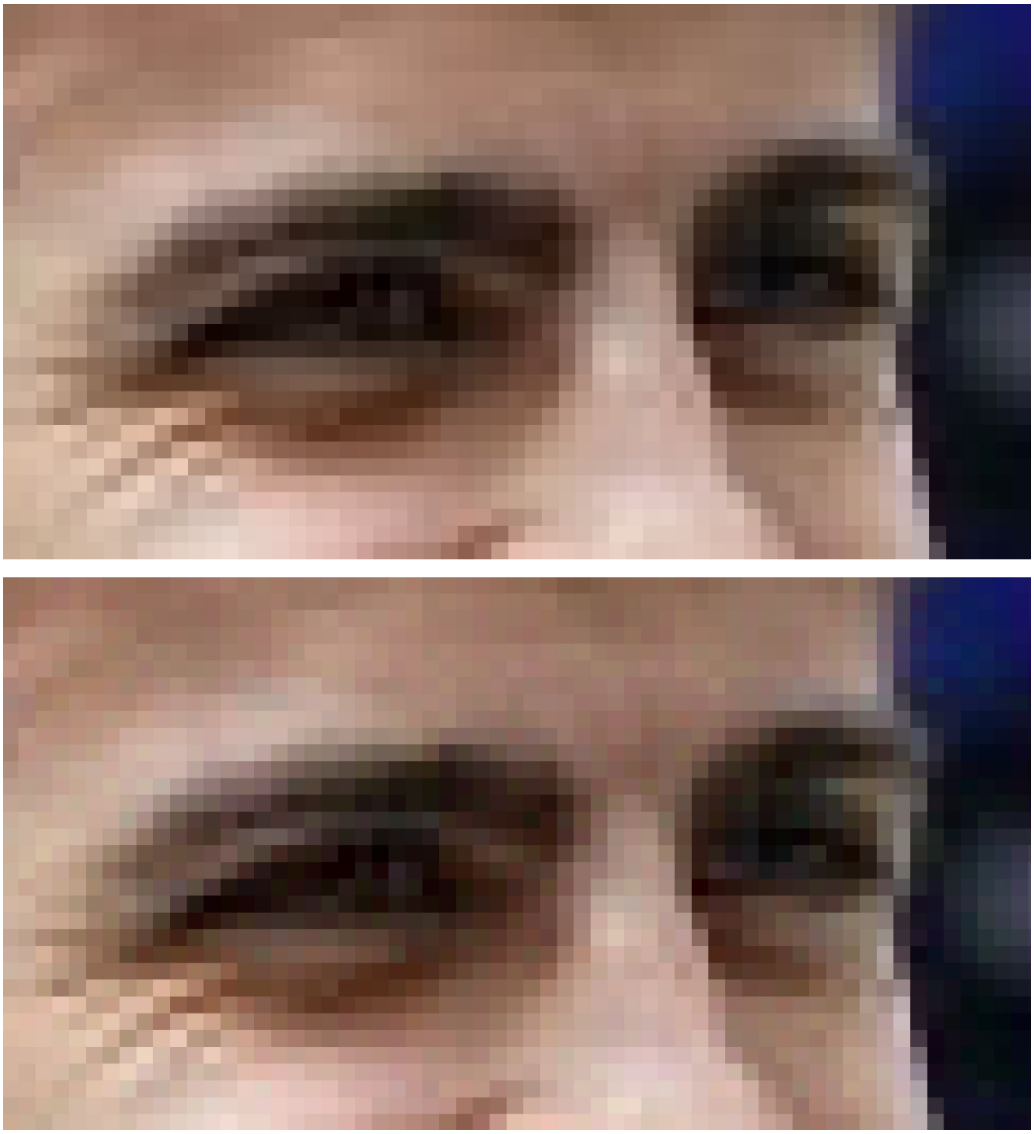

Figure 4.6: Detail of a protected image. Above the real image, below the protected image

To better understand the effect of the added signal, Figure 4.6 shows a detail of a face with a comparison between the real image and the protected image. There are no discernible differences in terms of colour or structure between the two images. However, the visual impact of the signal can be attributed to the greater amount of noise present in the protected image. This noise is

more noticeable in the brighter regions of the image.

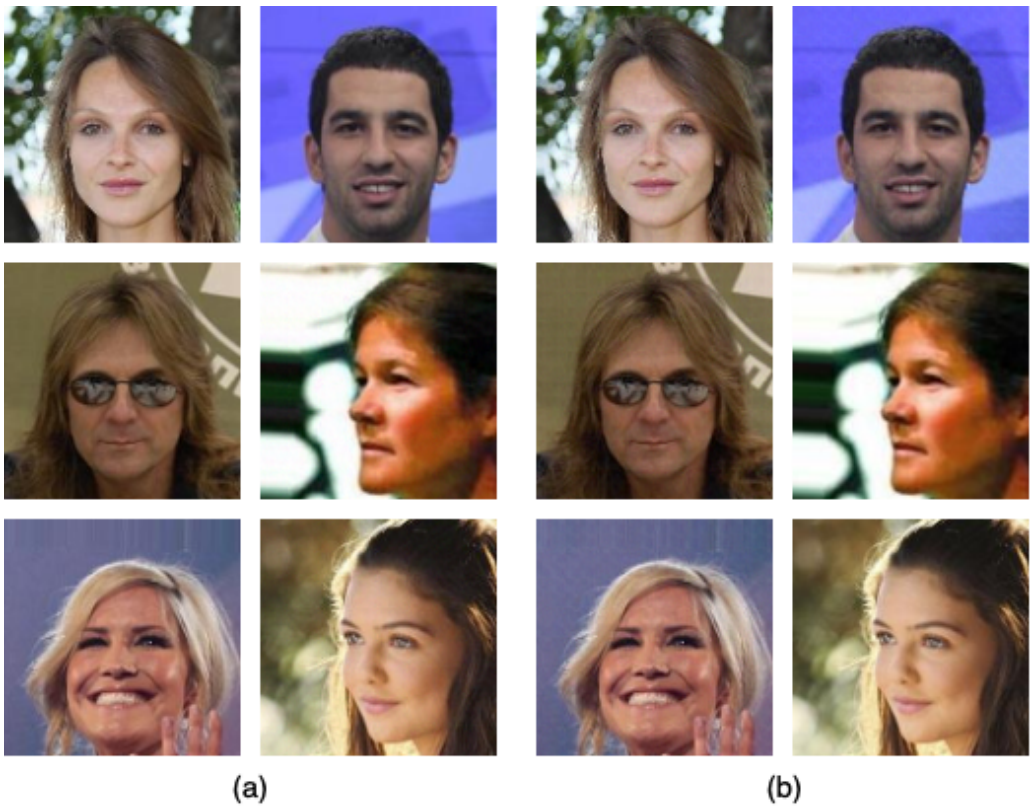

Figure 4.7: Comparison of real (a) and protected images (b)

These results confirm that adding the signal to the original image s_{IS} causes minimal degradation that does not affect the visual appearance.

4.5 Evaluating Signal Diversity

A critical aspect of our framework revolves around the customization of individual signals for each image. To this end, we compute the average distance between signals across various images, utilizing MSE as the metric. This metric is preferred because it directly considers pixels in the distance calculation, effectively highlighting significant pixel-level deviations between signals.

We avoid using the cosine similarity metric due to its potential ambiguity in specific situations. For instance, if the model learned to generate only two types of signals that are opposite to each other, and a batch consists solely of these two signals, the average cosine similarity would be 0. This is analogous to what we observed with loss 3 (Equation 3.8) without the *max* operation. Conversely, if all the signals were orthogonal to each other, the average cosine similarity would also be zero. Using cosine similarity does not allow for a distinction to be made between these two scenarios. This highlights the potential for misinterpretations when using cosine similarity, making MSE a more robust choice in this context.

Previous findings indicated that more complex patterns were associated with more complex generated signals. For each of the aforementioned $\mathcal{S}_E$ sets, 100 batches of 32 signals were generated. The MSE was then calculated between every two signals in each batch, resulting in a total number of unique comparisons of 24800.

Model	**Mean MSE**
Depth 1	57.805305
Depth 3	98.524712
Depth 6	98.204061

Table 4.4: Mean MSE for each model

The actual MSE distributions for the depth 1 (Figure 4.8), depth 3 (Figure 4.9) and depth 6 (Figure 4.10) models are shown below.

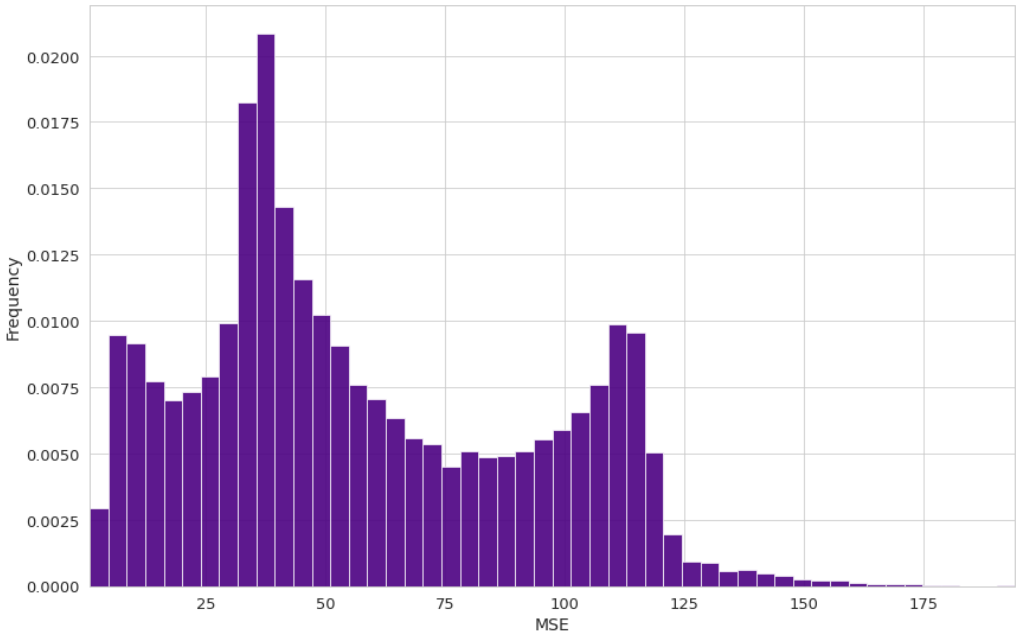

Figure 4.8: Distibution of MSE across signals with depth 1 model

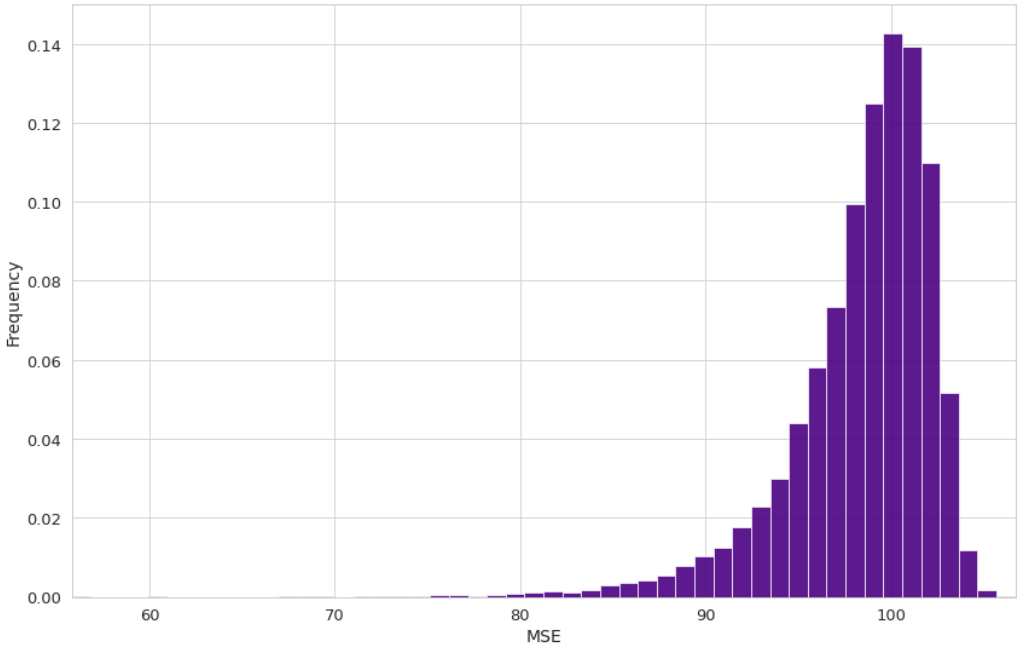

Figure 4.9: Distibution of MSE across signals with depth 3 model

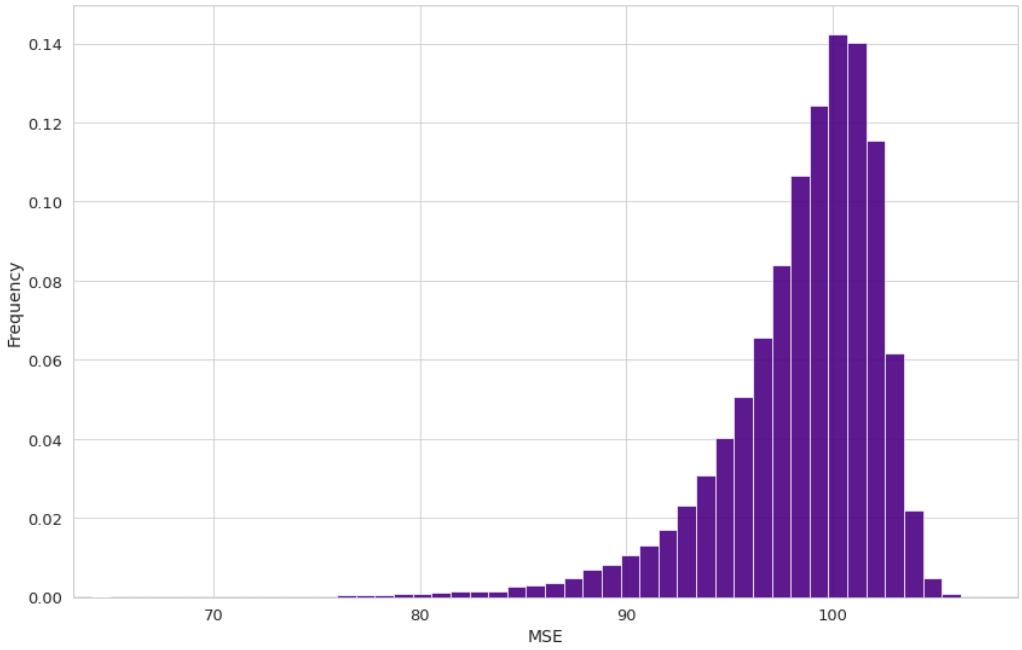

Figure 4.10: Distibution of MSE across signals with depth 6 model

Upon examination of the signal produced, it is clear that the depth 1 model is inadequate in generating a diverse range of signals. This conclusion is supported by the distribution of its MSE (Figure 4.8), which has a mean of 57.805305 and is skewed towards zero, indicating that the generated signals are exceedingly similar to one another. In comparison, the two other models exhibit comparable distributions, with mean values of 98.524712 and 98.204061 for depths 3 and 6, correspondingly. These values deviate significantly from zero, revealing the ability in generating a wider range of signals.

4.6 Assessing the usefulness of signals

Our current goal is to determine the actual usefulness of the signal in our framework. To achieve this, we conducted a test with a trained model that omitted the use of the encryption module. This means that all input images received by the localisation module have no applied signal. The rationale for this decision is to eliminate the signal's impact as much as possible, in

order to determine the level of its contribution. The testing attribute used was 'Bushy Eyebrows' to ensure the model relies solely on the signal in the detection process.

Model	Signal	Fakeness Map CS	Detection Accuracy
1	Without	0.801142	0.550979
1	With	0.804666	0.992761
3	Without	0.817571	0.702652
3	With	0.826790	0.99975
6	Without	0.800226	0.842961
6	With	0.808187	0.990332

Table 4.5: Performance without signal

As shown in Table 4.5, it is evident that all the models experience a decline in performance when presented with images without a signal. This confirms that the signal is indeed employed for detection.

In addition, it appears that the larger models (3 and 6) experience a notable, but less pronounced, drop in performance in the absence of signal compared to the smaller model. This implies that, in terms of detection, these models depend not only on the reconstructed signal but also on the image itself. This behaviour might be caused by the artifacts left by STGAN, the GM we used during training. These artifacts might be big enough to be spotted by a bigger transformer that learns to exploit the artifacts left by the GM, in addition to relying on the signal.

4.7 Generalization across attributes

Until now, the models have been tested on the same 'bald' attribute that was used during training. We are now interested in observing the performance

with novel attributes, previously not seen during training, to establish how well the models are able to generalise to new types of manipulation. This is a critical point as we cannot predict the type of image manipulation in a real world scenario, so it is important to understand how well the model generalises to new, unseen manipulations.

The table displays the results of evaluating the models based on an attribute that was not encountered during training.

Model	**Test Attribute**	**Fakeness Map CS**	**Detection**
1	Bushy Eyebrows	0.804666	0.992761
3	Bushy Eyebrows	0.826790	0.999750
6	Bushy Eyebrows	0.808187	0.990332
1	Mouth Slightly Open	0.863566	0.990983
3	Mouth Slightly Open	0.867282	0.999040
6	Mouth Slightly Open	0.864114	0.999048

Table 4.6: Performance across different attributes

All models were trained with the attribute “Bald” and evaluated with “Bushy Eyebrows” and 'Mouth Slightly Open'. Table 4.6 shows that regardless of complexity, the models achieve outstanding detection performance and a satisfactory localisation outcome. These outcomes indicate that image manipulation is detected regardless of the type of alteration, enabling the models to generalise to unseen alterations. On the other hand, the localisation task is not as easy to generalise, probably due to its greater complexity, but the model still manages to perform well on an attribute that was never seen during training. In Figure 4.11 are shown some samples of outputs generated by the depth 3 model using the attribute “Bushy Eyebrows”.

Figure 4.11: Results Visualization for "Bushy Eyebrows" attribute using depth 3 model: (a) original image, (b) encrypted image, (c) manipulated image, (d) ground truth fake map, (e) fake map for the manipulated image, (f) fake map for the protected image.

The model with depth 6 performs slightly worse than the model with depth 3, despite being more complex. This behaviour can be explained by looking at the results of table 4.5, where it can be seen that model 6 manages to perform detection, albeit with significantly lower performance, using only the image, indicating that for this model the contribution of the signal is used, but in combination with the artifacts left on the manipulated image. This behaviour may be due to two factors. Firstly, the loss function in equation 3.6 may not be strict enough to constrain such complex models. Secondly, the GM models used may generate many subtle artifacts that the transformer can distinguish, causing the model to rely less on the signal.

4.8 Baseline Comparison

In this section, we will compare our models with MaLP across localisation and detection. All the models will be evaluated with the "Bald" attribute, as this is the attribute on which they were all trained using STGAN.

The results are presented in Table 4.7, where we assess the performance of different models based on Fakeness Map CS and accuracy.

Model	Fakeness Map CS	Detection Acccuracy
MaLP	0.924732	0.980206
Depth 1	0.951839	0.999975
Depth 3	0.950642	0.999975
Depth 6	0.953754	1.0

Table 4.7: Baseline performance comparison

As shown in Table 4.7, our models outperform the MaLP consistently in both detection and localisation tasks. The Depth 1, Depth 3, and Depth 6 models demonstrate superior performance compared to MaLP in Fakeness Map CS, and in Detection Accuracy, all three depth models surpass MaLP,

with Depth 6 achieving a flawless score. These outcomes emphasise the improved capabilities of our models in effectively countering manipulation by a GM.

5

Conclusions

This thesis focus on detecting image manipulation performed by generative models. We introduced a novel approach for detecting and localising manipulated images with a particular focus on improving image protection through the implementation of a unique per-image signal protection mechanism. A vision transformer is utilised to adapt a learnable source template to each input image, creating a tailored signal for each image as protection. We deploy an additional vision transformer to extract the augmentations from authentic as well as manipulated images. The vision transformer output is then combined with CNN layers to detect and locate image manipulation. This entirely unsupervised process is guided by personalized losses for each model during training. Our findings outperform previous work, while having stronger protection capabilities, as other work relies on a fixed size template set. We have demonstrated the satisfying performance of our model on previously unseen manipulations. Additionally, we highlighted the significance of the signal for our models and their capacity to generate novel and diverse signals.

One limitation of this work is that it is restricted to the use of only one GM due to resource and time constraints. Conducting further evaluations on various GMs could provide greater insights into the models' generalisation capabilities. Another area for improvement is the use of transformer models. Currently, some models incorporate CNN, but it would be worthwhile to investigate the possibility of creating transformer-exclusive models for localization and detection tasks.

A

Appendix

A.1 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) have significantly impacted computer vision by efficiently capturing spatial hierarchies and learning intricate features within images. Unlike traditional feed-forward neural networks, CNNs use convolutional layers to discern patterns, making them particularly suited at image-related tasks such as pattern recognition, object classification, detection, and image segmentation.

In CNNs, the core operation is the convolutional layer, which involves convolving input data with a set of learnable filters. These filters scan the input image to detect various features, such as edges, textures, or more complex patterns, producing feature maps that highlight the presence and location of these features within the input.

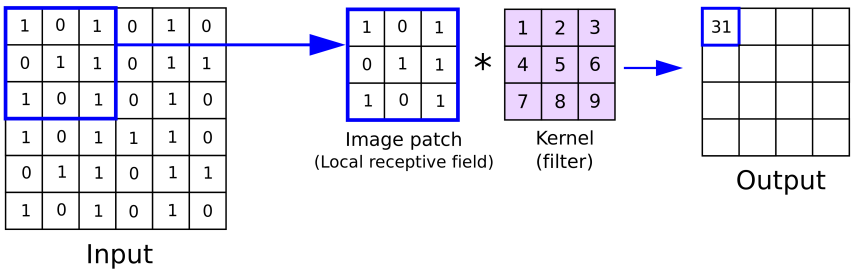

Figure A.1: Convolutional Operation in CNN

Mathematics of Convolution: The convolutional operation of a CNN layer is mathematically defined as the element-wise multiplication and summation of the input data and a set of learnable filters [13]. Given an input matrix I and a filter matrix H, the convolutional operation with output O is computed as follows:

$$O(i,j) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} I(m,n) \cdot H(i+m, j+n) \tag{A.1}$$

Here, M and N represent the dimensions of the filter matrix H, and i, j are the indices of the output feature map O. The filter is applied one step at a time to overlapping regions of the input matrix, and the outcomes are added up to create each element of the output feature map.

The operation described in Equation A.1 is technically a cross-correlation, not a convolution, as per signal theory. This distinction is due to the absence of a necessity to "flip" the kernel in relation to the input during cross-correlation (see Equation A.2) . Despite this, it's worth noting that in the realm of machine learning libraries, the term "convolution" is commonly used to refer to this operation in CNNs. For consistency, we will adhere to this convention in when referring to CNN in our models.

$$O(i,j) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} I(m,n) \cdot H(i-m, j-n) \tag{A.2}$$

Strides and Padding: Convolutional layers use strides and padding to control the spatial dimensions of the output feature maps. Strides determine the step size of the filter movement, influencing the spatial resolution of the output. Padding involves adding extra border pixels to the input to prevent a reduction in spatial dimensions and retain more information.

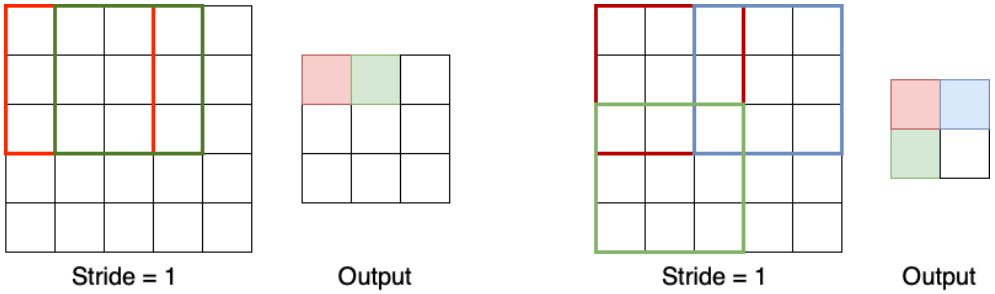

Figure A.2: Stride Visualization

The output size O_{size} for a given input size I_{size}, filter size H_{size}, stride S, and padding P is calculated as:

$$O_{size} = \frac{I_{size} - H_{size} + 2P}{S} + 1 \tag{A.3}$$

Architecture: Following the convolutional operation, an activation function is typically applied element-wise to introduce non-linearity into the model. Commonly used activation functions include Rectified Linear Unit (ReLU), which sets negative values to zero:

$$\text{ReLU}(x) = \max(0, x) \tag{A.4}$$

Non-linearities help the CNN model capture complex relationships within the data.

Additionally, pooling layers are often interspersed with convolutional layers to downsample feature maps. Pooling, commonly in the form of max pooling or average pooling, reduces the spatial dimensions of the feature maps, focusing on the most salient information and enhancing computational efficiency. In max pooling, a kernel traverses the feature map, selecting the maximum value from a group of neighboring pixels. This process emphasizes the most prominent features in a given region. Similarly, in average pooling, the kernel computes the average value within the same context.

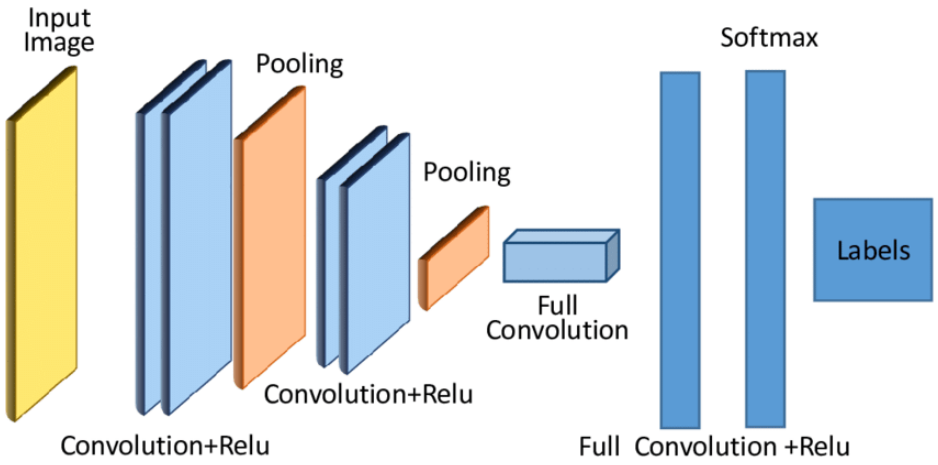

Figure A.3: CNN Architecture

A typical CNN architecture(as seen in Figure A.3) consists of multiple convolutional layers followed by activation functions and pooling layers. These layers are usually followed by one or more fully connected layers for classification or regression tasks. The hierarchical arrangement of these layers allows the network to learn progressively abstract and complex features from the input data.

A.2 Transformers

Transformers, as introduced in [14], have gained significant popularity, especially in the realms of natural language processing and computer vision tasks. They have emerged as a compelling alternative to models like LSTM, primarily due to their ability to address scalability issues associated with sequential processing.

Renowned for their parallelization capabilities, transformers excel in capturing long-range dependencies in data. In contrast, LSTM, which relies on sequential processing, encounters challenges in parallelization. This limitation arises because LSTM processes input data sequentially, making it less efficient in handling tasks that could benefit from simultaneous computation.

Transformers, on the other hand, leverage attention mechanisms to efficiently process input data. This unique reliance on attention mechanisms enables the model to selectively focus on different parts of the input, a crucial feature for tasks that require a nuanced understanding of relationships between various elements in the data.

Attention: In the context of transformers, attention is computed using three linear transformations: Query (Q), Key (K), and Value (V). These transformations are applied to the input data to produce query, key, and value matrices. The attention score is calculated by taking the dot product of Q and K matrices, followed by scaling and applying a softmax function. This product is a measure of the similarity or relevance between different elements in Q and K. It quantifies how much attention should be given to each element in relation to every other element in the sequence. The resulting attention weights are then used to compute a weighted sum of the value matrix, which forms the output of the attention mechanism.

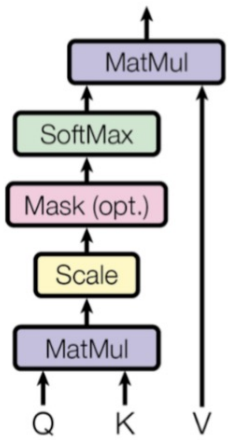

Figure A.4: Scaled Dot Product Attention

The attention formula can be represented as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{A.5}$$

Here, d_k is the dimension of the key vectors. The square root of $d_k$is introduced in the denominator for scaling purposes. This scaling helps stabilize the gradients during the training process.

Transformers employ two types of attention mechanisms: self-attention and cross-attention.

In self-attention, the same input acts as both Q, K, and V. This mechanism assesses relationships within the input sequence, enabling the model to discern the relative significance of different elements within the same sequence in input.

Meanwhile the cross-attention involves using an external context as K and V, while the input acts as Q. This means that the model assesses the input with respect to information from an external source. This enables the model to consider the relevance of elements in the input sequence in the broader context provided by the external information. Cross-attention is vital for tasks involving interactions between input elements and external context, like in machine translation or image captioning. It enables the model to align elements from diverse sources, facilitating more accurate and contextually-aware predictions.

Architecture: A transformer consists of a series of attention blocks, each followed by a feed-forward layer. The initial embedding layer provides the input for the first block, while subsequent blocks receive the output of the preceding block as their input. This iterative process allows the transformer to progressively refine its understanding of the data.

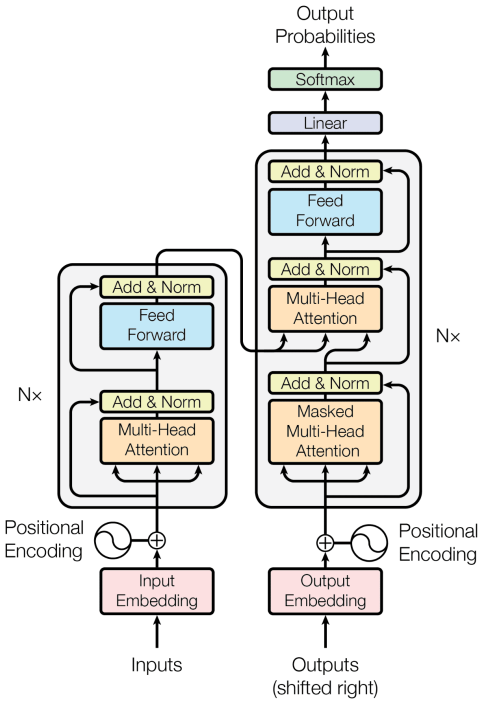

Figure A.5: Transformer Architecture

In the absence of recurrence and convolution, transformers need a method to account for sequence order. To address this, positional encodings were introduced in the input embeddings of both the encoder and decoder. These encodings, sharing dimensions with the embeddings, are summed for seamless integration. Sine and cosine functions with different frequencies were chosen among various options for positional encodings. This augmentation enables the model to discern relative or absolute positions of tokens within a sequence.

Vision Transformers: Transformers have proven effective in computer vision through architectures like Vision Transformers (ViT). ViT breaks down an image into fixed-size patches, linearly embedding them to create a sequence for transformer processing. This enables the model to capture both

global and local features simultaneously, leveraging the attention mechanism to understand spatial relationships between patches.

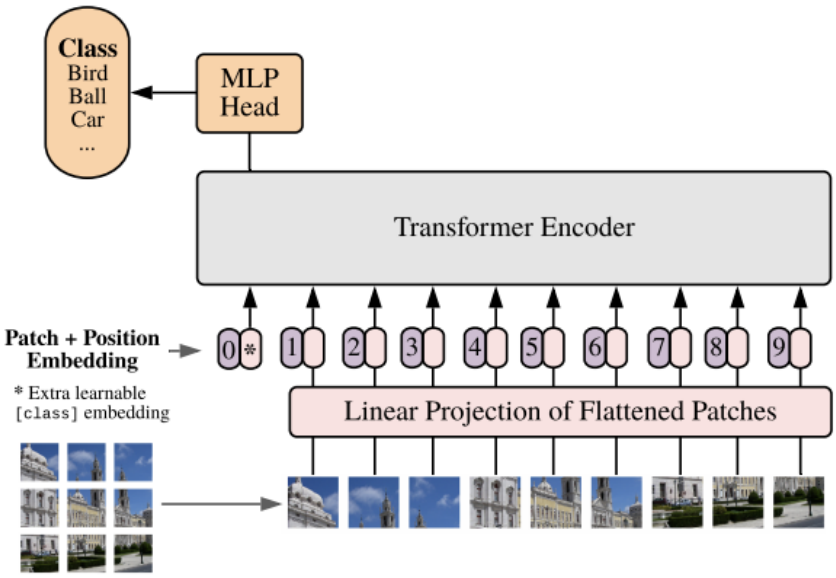

Figure A.6: Vision Transformer (ViT)

In ViT, the nature of the model poses a challenge in conveying the spatial arrangement of patches. This problem is solved by introducing positional embeddings, learnable parameters that are added to the patches, allowing the model to distinguish the relative positions of different patches within the image. Through the injection of positional embeddings, the model acquires an understanding of spatial layout, preserving crucial spatial information, accurately capturing relationships between patches and enhancing the model's interpretation of complex spatial structures in visual data.

Limitations: ViT exhibits certain inherent limitations when compared to traditional CNNs used in computer vision, notably lacking key inductive biases such as translation equivariance and locality.

Translation equivariance, a characteristic of traditional CNNs, ensures that a shift in input data corresponds to a corresponding shift in the model's output, providing robustness to variations in feature positions within an image (see Figure A.7). In contrast, ViT may struggle to capture patterns undergoing spatial shifts due to the absence of inherent translation equivariance.

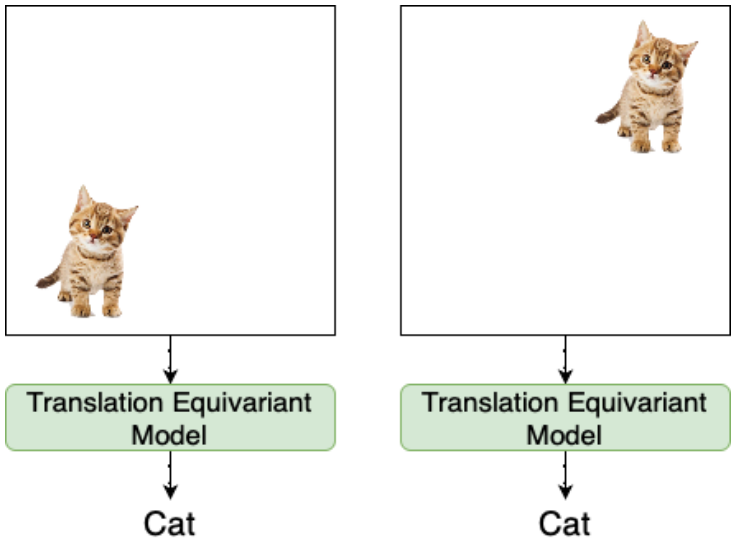

Figure A.7: Translation Equivariance

Locality in computer vision denotes that nearby pixels in an image are likely related and share information. Traditional CNNs use local receptive fields to capture these spatial relationships effectively. ViT, however, processes the entire image as a sequence of tokens, potentially neglecting crucial local interactions among pixels.

The absence of translation equivariance in ViT can present challenges for effective generalization, especially with limited training data. In contrast, CNNs, with their convolution operations and max pooling/striding, exhibit translation invariance, contributing to their robust performance. However, it's noteworthy that Transformers, such as ViT, can overcome these generalization challenges when trained on more extensive datasets, ranging from

14 million to 300 million images. The increased dataset size allows ViT to learn and capture diverse patterns, compensating for its initial limitations in handling spatial shifts.

A.3 CNN Based Decoder

During the initial phase of this study, we sought to adopt the architecture of MaLP [6], with the sole alteration being the substitution of the encryption module with a transformer. While maintaining a CNN-based structure for the localization module, this configuration ultimately proved ineffective, leading to unsatisfactory model performance.

Challenges emerged in both signal recovery and localization tasks. The fakeness map of real images did not yield blank outcomes, indicating the model's difficulty in distinguishing and reconstructing the authentic signal from the manipulated one. Only the detection aspect appeared to be operational, albeit with performance levels comparable to previous passive works. This suggests that the model was neglecting the added signal.

The performance of this initial experiment is summarized in the table below:

Metric	**Value**
Detection Accuracy	0.938734
Map Fake Similarity	0.602581

Table A.1: CNN Based Encoder Performance

A.4 Cosine Similarity

Cosine similarity serves as a metric for measuring the similarity between two non-zero vectors within an inner product space.

Given two vectors, A and B, the cosine similarity (cosine of the angle θ between them) is defined as:

$$\mathrm{CS}(A, B) = \frac{A \cdot B}{\|A\|\|B\|} \tag{A.6}$$

The numerator quantifies the shared directional components of the vectors, while the denominator normalizes the result based on their magnitudes.

Cosine similarity is particularly beneficial when the emphasis lies on the directional alignment of vectors rather than their absolute magnitudes. Geometrically, it assesses the cosine of the angle between vectors in a multidimensional space, reflecting the similarity in their directional trends. A cosine similarity of 1 indicates perfect alignment, where vectors point in the same direction, while a value of 0 signifies orthogonality (perpendicularity), and -1 implies perfect misalignment, where vectors point in exactly opposite directions.

It's essential to note that while cosine similarity excels in scenarios where the magnitude of vectors is less critical, there may be instances where considering both direction and magnitude is necessary.

A.5 Structural Similarity Index

The Structural Similarity Index (SSIM) is a metric used to quantitatively assess the similarity between two images. It takes into account luminance, contrast, and structure, providing a more comprehensive evaluation compared to pixel-wise metrics like Mean Squared Error (MSE).
The SSIM is calculated using the following formula:

$$\mathrm{SSIM}(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{A.7}$$

where:

- x, y represent the compared images
- μ_x, μ_y are the mean luminance values of x and y, respectively
- σ_{xy} denotes the covariance between x and y
- σ_x^2 and σ_y^2 are the variances of x and y
- C_1 and C_2 are constants introduced to stabilize the division with a weak denominator, typically taking small values (e.g., 0.01)

The SSIM ranges between -1 and 1, where 1 indicates perfect similarity between the images. A higher SSIM value implies a closer resemblance between the compared images. This metric not only considers pixel values but also their spatial arrangements and structural information, proving to be a more accurate measure of image similarity, especially in cases where slight structural differences may be imperceptible to the human eye.

A.6 Learned Perceptual Similarity

Learned Perceptual Image Patch Similarity (LPIPS) [15] measures the perceptual similarity between two images, focusing on local patches rather than global structures. Valuable in computer vision and image processing, LPIPS provides insights into how humans perceive image differences.

Unlike metrics relying solely on pixel-wise or global features, LPIPS considers the perceptual significance of image patches, evaluating similarity at a granular level.

LPIPS computes similarity by comparing activations of two image patches for a pre-defined network, matching human perception well. A low LPIPS score indicates perceptual similarity.

The metric assesses local image patches based on factors like color, texture, and structure, using features extracted from deep neural networks trained on large datasets. The LPIPS score ranges between 0 and 1, with 0 implying perfect patch-level similarity. It's advantageous where absolute pixel-wise differences may not align with human perception; for example, two images with different pixel values but similar visual content may receive a low LPIPS score, indicating higher perceptual similarity.

Bibliography

[1] Andreas Rossler et al. *FaceForensics++: Learning to Detect Manipulated Facial Images*. 2019. arXiv: 1901.08971 [cs.CV].

[2] Xi Wu et al. "SSTNet: Detecting Manipulated Faces Through Spatial, Steganalysis and Temporal Features". In: May 2020, pp. 2952–2956. DOI: 10.1109/ICASSP40776.2020.9053969.

[3] Hao Dang et al. *On the Detection of Digital Face Manipulation*. 2020. arXiv: 1910.01717 [cs.CV].

[4] Lucy Chai et al. *What makes fake images detectable? Understanding properties that generalize*. 2020. arXiv: 2008.10588 [cs.CV].

[5] Vishal Asnani et al. *Proactive Image Manipulation Detection*. 2022. arXiv: 2203.15880 [cs.CV].

[6] Vishal Asnani et al. *MaLP: Manipulation Localization Using a Proactive Scheme*. 2023. arXiv: 2303.16976 [cs.CV].

[7] Nataniel Ruiz, Sarah Adel Bargal, and Stan Sclaroff. *Disrupting Deepfakes: Adversarial Attacks Against Conditional Image Translation Networks and Facial Manipulation Systems*. 2020. arXiv: 2003.01279 [cs.CV].

[8] Run Wang et al. *FakeTagger: Robust Safeguards against DeepFake Dissemination via Provenance Tracking*. 2021. arXiv: 2009.09869 [cs.CR].

[9] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. *Explaining and Harnessing Adversarial Examples*. 2015. arXiv: 1412.6572 [stat.ML].

[10] Sheng-Yu Wang et al. *CNN-generated images are surprisingly easy to spot... for now*. 2020. arXiv: 1912.11035 [cs.CV].

[11] Hao Dang et al. *On the Detection of Digital Face Manipulation*. 2020. arXiv: 1910.01717 [cs.CV].

[12] Ming Liu et al. *STGAN: A Unified Selective Transfer Network for Arbitrary Image Attribute Editing*. 2019. arXiv: 1904.09709 [cs.CV].

[13] Piotr Skalski. *Gentle Dive into Math Behind Convolutional Neural Networks*. Accessed on: 11/11/23. 2019. URL: https://towardsdatascience.com/gentle-dive-into-math-behind-convolutional-neural-networks-79a07dd44cf9.

[14] Ashish Vaswani et al. *Attention Is All You Need*. 2017. arXiv: 1706.03762 [cs.CL].

[15] Richard Zhang et al. *The Unreasonable Effectiveness of Deep Features as a Perceptual Metric*. 2018. arXiv: 1801.03924 [cs.CV].