StylePuncher: encoding a hidden QR code into images

Farhad Shadmand[1][a], Luiz Schirmer[2][b] and Nuno Gonçalves[1][c]
[1] *Institute of Systems and Robotics, University of Coimbra, Portugal*
[2] *University of the Sinos River Valley Rio de Janeiro, Brazil*
farhad.shadmand@isr.uc.pt, luizschirmer@unisinos.br, nunogon@deec.uc.pt

Keywords: StegaStamp, watermark, deep learning, Generative adversarial Networks, Style Transfer

Abstract: Recent advancements in steganography and deep learning have enabled the creation of security methods for imperceptible embedding of data within images. However, many of these methods require substantial time and memory during the training and testing phases. This paper introduces a lighter steganography (also applicable to watermarking purposes) approach, StylePuncher, designed for encoding and decoding 2D binary secret messages within images. The proposed network combines an encoder utilizing neural style transfer techniques with a decoder based on an image-to-image transfer network, offering an efficient and robust solution. The encoder takes a $(512 \times 512 \times 3)$ image along with a high-capacity 2D binary message containing 4096 bits (e.g., a QR code or a simple grayscale logo) and ”punches” the message into the cover image. The decoder, trained using multiple weighted loss functions and noise perturbations, then recovers the embedded message. In addition to demonstrating the success of StylePuncher, this paper provides a detailed analysis of the model's robustness when exposed to various noise perturbations. Despite its lightweight and fast architecture, StylePuncher achieved a notably high decoding accuracy under noisy conditions, outperforming several state-of-the-art steganography models.

1 INTRODUCTION

The primary goal of image steganography is to encode secret messages into a cover image so that the encoded and original images appear visually identical. While focused on steganography, StylePuncher can also be applied to watermarking for copyright protection. Inspired by prior works (Shadmand et al., 2024), (Shadmand et al., 2021), and (Tancik et al., 2020), our model advances the field as an effective steganography technique.

The overall performance of a steganography method is typically evaluated based on four key characteristics: invisibility, information capacity, security, and robustness to transmission through printing media (printer-proof ability). These criteria collectively determine the method's effectiveness in concealing data while ensuring resilience and maintaining the integrity of the encoded message under various conditions. Invisibility is measured by the similarity between original and encoded images, ideally imperceptible to the human eye. We argue that human visual judgment is the most effective evaluation method. Additionally, the capacity is quantified in bits-per-pixel (bpp), which refers to the average number of bits embedded into each pixel of the cover image (Zhang et al., 2019). The security of encoded images involves (1) resilience to noise, preserving encoded information, and (2) robustness against adversaries attempting to decode or manipulate the data using similar models. The printer-proof characteristic is assessed by the success rate of decoding physically printed encoded images, posing a challenge for steganography models to preserve embedded information through the print-scan process.

Current state-of-the-art steganography techniques face several key challenges: (1) limited capacity for secret messages, restricting practical security applications; (2) low robustness against various noise types; (3) the need for higher image quality and improved invisibility; and (4) reliance on large architectures requiring extensive datasets, making them inefficient and impractical for real-time or resource-constrained applications.

This paper introduces StylePuncher, a novel deep learning steganography method designed to address current limitations by embedding high-capacity mes-

[a] https://orcid.org/0000-0003-4399-4845
[b] https://orcid.org/0000-0003-4102-1986
[c] https://orcid.org/0000-0002-1854-049X

sages into RGB images. Inspired by image-to-image transfer models (Goodfellow et al., 2014), (Isola et al., 2017), (Wang et al., 2018), (Abdal et al., 2019), StylePuncher is robust against digital and physical noise. Its separate training of the encoder and decoder reduces GPU requirements, while the lightweight encoder network enhances efficiency compared to models like CodeFace (Shadmand et al., 2021) and StampOne (Shadmand et al., 2024), improving speed and resource utilization.

The encoder design is two folded: neural style transfer and linear interpolation (improving the appearance of the encoded images), as visualised in Figure 1.

The decoder network in StylePuncher is based on the U-Net architecture, as employed in the pix2pix network, and is designed to recover the secret message from encoded ("punched") images. Trained independently from the encoder, the decoder uses encoded images as input and the secret message as the ground truth label, as shown in Figure 2. We evaluated four decoder configurations: standard U-Net, U-Net with a discriminator, U-Net with an attention mechanism (Oktay et al., 2018), and U-Net with a Spatial Transformer Network (STN) (Jaderberg et al., 2015). The decoder training incorporates four loss functions: perceptual loss (Zhang et al., 2018), object loss (Isola et al., 2017), total variation (TV) loss (Arjovsky et al., 2017a), and a StegaStamp discriminator (Tancik et al., 2020) to enhance performance.

As for invisibility and quality of StylePuncher encoded images, they are measured by three perceptual loss functions (Chen and Bovik, 2011) (Zhang et al., 2018)(Kettunen et al., 2019), a face features distance (Deng et al., 2019) and a color histogram loss function (Afifi et al., 2021). The results from the tests with the three loss functions showed than our model is a robust steganography model with capacity as high as 0.52×10^{-2} bpp.

In Section 6, we also present the sensitivity performance of our decoder networks in the presence of various simulated noise sources. The average decoder performance reaches approximately 85% successful rate in the presence of several distortion perturbations while having the best performance between the robust steganography models (Shadmand et al., 2021), (Tancik et al., 2020).

2 Related Work

Image Steganography was initially performed with the use of traditional computer vision tools, such as Discrete Wavelet Transform (DWT) (Barni et al., 2001), Discrete Fourier Transform (DFT) (O'Ruanaidh et al., 1996), Discrete Cosine Transform (DCT) (Hsu and Wu, 1999) or LSB (Tamimi et al., 2013). LSB uses the least significant bits (LSB) to hide a secret message into a cover image (Tamimi et al., 2013). The hiding capacity of LSB is approximately 0.20.2 bits per pixel (bpp) (Zhu et al., 2018). While these methods allow high-capacity message encoding, they often struggle with maintaining the perceptual quality of encoded images and suffer from data loss under noise-induced distortions.

Recent advances in deep learning, particularly Generative Adversarial Networks (GANs) (Goodfellow et al., 2014), (Mirza and Osindero, 2014), have significantly improved image steganography performance.

The first deep learning-based steganography method, DeepStega (Baluja, 2017), utilized autoencoding networks to encode a $64 \times 64 \times 3$ secret image into a cover image of identical resolution. During training, small noise was added to the encoder's output to prevent encoding the secret image directly into the binary space of the cover image, such as LSB.

The StegaStamp method (Tancik et al., 2020) introduced a robust steganography technique capable of validating encoded messages from physically printed

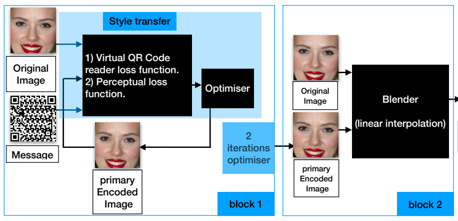

Figure 1: StylePuncher encoder architecture. In block 1, the StylePuncher encoder is designed with loss functions and an optimiser that "punches" a 2D binary message into images while optimizing variables of the primary encoded images in two epochs. In block 2, the primary encoded image is blended with the original image to produce encoded images.

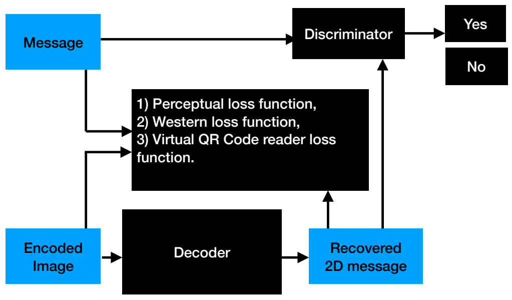
Message
Discriminator
Yes
No
1) Perceptual loss function, 2) Western loss function, 3) Virtual QR Code reader loss function.
Encoded Image
Decoder
Recovered 2D message

Figure 2: The decoder follows a structure similar to the pix2pix network and is trained using perceptual loss, Western loss, and a modified virtual QR Code mechanism to enhance performance and robustness.

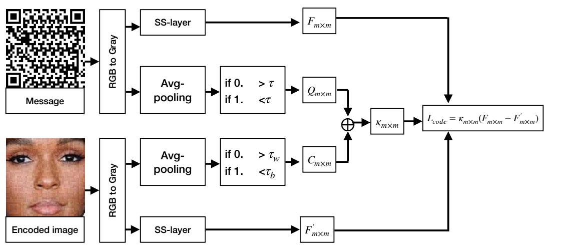

Figure 3: Virtual QR code reader is inspired in ArtCode (Su et al., 2021), where we replace average-pooling to find the center pixel faster than the original model. The kernel of the SS-layer for the encoder is a Gaussian function like ArtCode, but we use a matrix with ones for the decoder.

images. This was achieved through noise simulation techniques to mimic printer and digitization perturbations. By employing LPIPS perceptual loss (Zhang et al., 2018) during training, the method minimized perceptual quality differences between encoded and cover images, significantly enhancing the appearance of encoded samples.

The CodeFace model (Shadmand et al., 2021) is the first practical full model for encoding and decoding secret messages from small face images in ID documents. It enhances image quality by minimizing differences in facial features between encoded and original images and improves decoder performance by training with low-resolution encoded images. Both CodeFace and StegaStamp encoders can hide up to 100 bits in $400\times400\times3$ pixel images but decode reliably only from large printed images with high texture levels, such as 6×6 cm images.

The HiNet model (Jing et al., 2021), built on deep learning normalizing flows, employs an invertible neural network (INN) to embed one image within another of the same size. Utilizing wavelet domain transformation and inverse learning, HiNet enables high-capacity, secure, and imperceptible message embedding with strong recovery accuracy. It supports a payload capacity of $24-120$ bpp but is highly sensitive to noise and perturbations.

Recently, the StampOne (Shadmand et al., 2024) model proposed a generalized approach to enhancing steganography using deep learning, specifically based on Generative Adversarial Networks (GANs). This method aims to increase both the message capacity and robustness of the model when subject to various printer and camera augmentations.

3 Methodology

StylePuncher consists of a style transfer encoder network, inspired by ArtCode (Su et al., 2021), and an image-to-image transfer decoder. The encoder incorporates a virtual QR Code simulator loss function introduced in ArtCode, optimized using the Adam optimizer. The style transfer optimization process is executed twice to embed the message points into the original image, resulting in encoded images. Following this embedding phase, the quality of the encoded images is further refined through linear interpolation with the original images, enhancing their visual fidelity. Further details are given in section 3.1.

The decoder network then takes the encoded images as input and reconstructs the QR Code message as output. It is trained over 28,000 steps for noise-free conditions, or 100,000 steps when simulating noise, to minimize the corresponding loss functions.

We implemented two separate applications for the encoder and decoder. Additionally, for detecting the region of interest in the encoded images, we utilized two models: YOLOv4 (Wang et al., 2021) and PRnet face detection (Wang and Solomon, 2019).

3.1 Encoder

StylePuncher's encoder network consists of two main components: a modified ArtCode Neural Style Transfer (NST) network (Su et al., 2021) and a linear interpolation operator that blends the primary encoded and original images (see Figure 1).

In modifying ArtCode, we replaced its original loss functions with a perceptual loss function (Zhang et al., 2018), instead of the VGG16 NST, to better preserve the appearance of the encoded images during training. To redesign the virtual QR code simulator mechanism, we retained the Sampling-Simulation (SS) layer (I_{ss}), which employs a convolution layer with a non-trainable Gaussian kernel and an $s\times s$ matrix to detect the center of black and white blocks, as follows:

$$G_{(i,j)} = \frac{1}{2\pi\sigma^2} e^{-\frac{i^2+j^2}{2\sigma^2}} \tag{1}$$

where (i, j) is a pixel of a kernel matrix and the origin at the module center. The factor σ adjusts the size in bits of the message punched in the cover image.

The loss function is illustrated in Figure 3. Both the encoded image and the secret message are converted from RGB to grayscale. The secret message is passed through the frozen SS-layer, where a binary feature vector ($F_{m\times m}$) is extracted. It is also processed through a 2D average pooling layer and mapped using ε_τ to compute the binary matrix ($Q_{m\times m}$) of the message, where τ is the binary threshold applied to the secret message pixels (QR Code).

The encoded image follows a similar process. Initially, the primary encoded image's pixel values

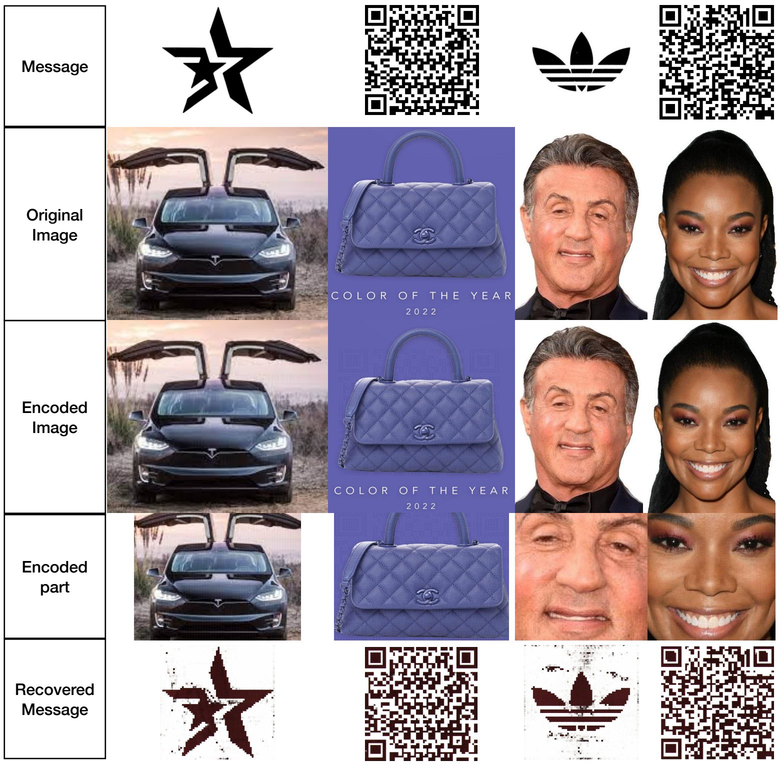

Figure 4: Samples of encoded images - StylePuncher can hide and read a message in an image's region of interest.

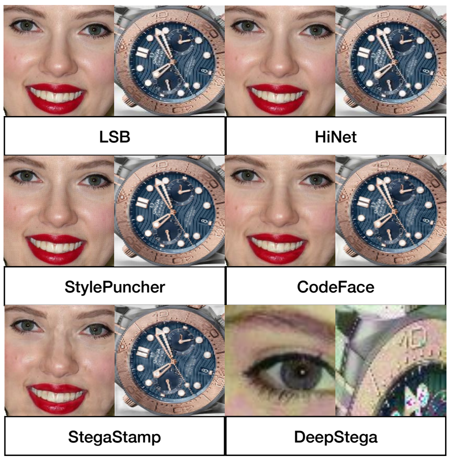

Figure 5: Examples of steganography models that encode a hidden message into image's ROI. The size of DeepStega's input and output is $64 \times 64 \times 3$, smaller than other models. Therefore, for a fair comparison, we use a smaller encoding ROI in DeepStega.

match those of the original image. During training, the network converts the encoded image from RGB to grayscale, and the SS-layer extracts its binary feature vector ($F'_{m\times m}$). It then passes through a 2D average pooling layer and is mapped using $\varepsilon'_{\tau_w,\tau_b}$, which computes the matrix of maximum pixels ($C_{m\times m}$) with thresholds τ_w and τ_b for white and black pixels, respectively.

The error weight ($\kappa_{m\times m}$) is defined as zero when $C_{m\times m}$ and $Q_{m\times m}$ are identical (both 0 or 1), and as one otherwise. The virtual QR Code loss function is then calculated using the following expression:

$$L_{code} = \kappa_{m\times m}(F_{m\times m} - F'_{m\times m}) \qquad (2)$$

where m is the size of the message and s is the size of the kernel. The details of L_{code} are shown in Figure 3.

By using the average pooling layer and the Tensorflow toolkit, to extract $C_{m\times m}$ and $Q_{m\times m}$, increases the speed by 100 times over the traditional approach of ArtCode (Su et al., 2021). The encoder produces images such that the perception of the encoded and original images is similar, with indistinguishable differences, as can be seen in Figure 4.

In the second block, to improve the encoder's performance, the pixel values between a primary encoded image and an original image are linearly interpolated to generate an invisible encoded image, as follows:

$$P_{enc}(ijc) = \alpha \times p_{pri}(ijc) + (1.0 - \alpha) \times p_o(ijc) \quad (3)$$

where p_{enc}, p_{pri} and $p_o(ijc)$ are normalized pixels (between 0 and 1) of the encoded, primary encoded and original images, respectively, for the channel c of

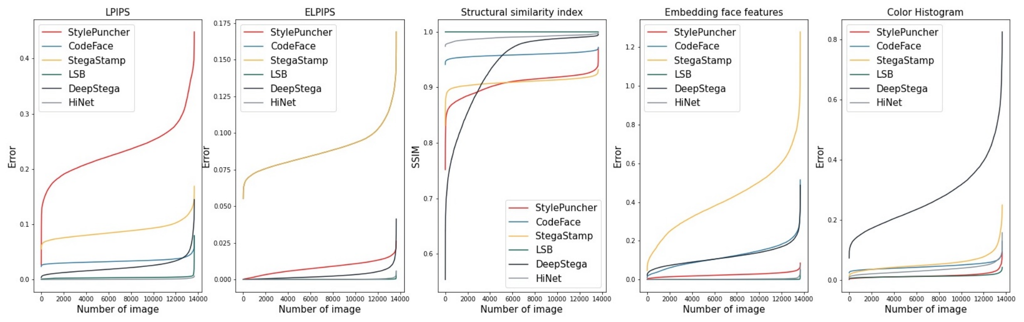

Figure 6: The quality of encoded images as a parameter of the encoder network is measured. The smaller the value, the better LPIPS and ELPIPS which measure the perceptual distance between encoded and original images. We also compute Euclidean distances between original and encoded face images as in ArcFace (Deng et al., 2019). The colour errors are computed between encoded and original photos by comparing colour histograms.

the pixel (ij). The value α is an interpolation factor that lies in the range $[0, 0.4]$.

3.2 Decoder

The decoder is an image-to-image transfer network that takes a $512 \times 512 \times 3$ encoded image as input and outputs a $512 \times 512 \times 1$ recovered message, which can be read by standard QR Code scanning applications, as illustrated in Figure 2

We have implemented several variations of the decoder network, all based on the pix2pix framework (Isola et al., 2017). The first version is a U-Net architecture, as used in the original pix2pix model, without incorporating a discriminator. The second version includes a Conditional Generative Adversarial Network (CGAN) discriminator, which classifies whether each image patch is real or fake. Upon comparison, we found that the inclusion of the discriminator negatively impacts the decoder's performance, as detailed in section 6. In the third model, we integrated an attention mechanism (Oktay et al., 2018) into the pix2pix framework. We chose the attention U-Net to enable the model to focus on relevant features during training, inspired by its application in medical image analysis (Oktay et al., 2018). Finally, we explored the use of a Spatial Transformer Network (STN) combined with pix2pix (Jaderberg et al., 2015). The STN helps to crop and normalize the appropriate region within the image, simplifying subsequent classification tasks and improving the decoder's overall performance.

For training the decoder, our approach utilizes a combination of loss functions, including the virtual QR Code reader loss (L_{DC}), the perceptual loss (L_{Per}) (Zhang et al., 2018), the object loss (L_{obj}) (Isola et al., 2017), and the total variation (TV) loss (L_{TV}) (Arjovsky et al., 2017a). The virtual QR Code reader used in the decoder is similar to the one employed in the encoder, but it uses a matrix with values equal to 1 instead of a Gaussian kernel in the SS-layer.

We incorporate a discriminator within the decoder to minimize the discrepancy between the extracted message and the original message. A Wasserstein adversarial loss (l_{disc}) (Arjovsky et al., 2017b) is computed by evaluating the difference between the discriminator's feature outputs for the extracted and original messages, ensuring a closer alignment between them.

The complete loss function is defined as:

$$Loss = W_{DC} \times l_{DC} + W_{Per} \times l_{Per} + W_{obj} \times l_{obj} + W_{TV} \times l_{TV} + W_{disc} \times l_{disc} \quad (4)$$

where W_{DC}, W_{Per}, W_{obj} and W_{TV} are the respective weights assigned to each loss component. W_{disc} and l_{disc} are the weight and loss function of the discriminator.

4 Datasets

The dataset used for training the StylePuncher model is the CelebFaces Attributes dataset (CelebA) (Liu et al., 2018), which contains 202,599 large-scale images of celebrities. For object images, we utilized the ImageNet dataset (Deng et al., 2009), which includes thousands of diverse images. From these two datasets, we randomly selected 50,000 images for training. For testing, we employed the PICS face dataset (Hancock, 2008) and the Color FERET face dataset (Phillips et al., 2000), comprising a total of 13,659 images. All facial images presented in this study belong to celebrities.

5 Perturbation simulation

Applying perturbation or noise self-attack simulations enhances decoder robustness in real-world scenarios. The three primary noise types impacting images are digital, printer, and sensor perturbations (Cunha et al., 2024).

Digital perturbations. Digital noise refers to distortions during transfer, storage, or application-level manipulation of images. Here, we focus solely on noise from JPEG compression.

Printer and designed perturbations. Printers introduce various noises, such as Gaussian noise, affine transforms, sharpening, and random gray transforms, which can compromise hidden information.

Sensor perturbations. Cameras capture images differently based on lighting and environmental conditions. Sensor perturbations, simulated in our solution, include random brightness, contrast, hue shifts, medium blur, perspective warps, and added padding.

6 Experiments

6.1 Training configuration and hardware setup

The StylePuncher networks were trained using the Adam optimizer (Kingma and Ba, 2014) with a learning rate of $10e^{-5}$ and $\beta = 0.95$. The training process involved 11×10^4 epochs with a batch size of 5, over a total duration of 48 hours. The input and output image sizes were $512 \times 512 \times 3$. We used two Nvidia Geforce GTX 1060 GPUs for training the networks.

6.2 Capacity evaluation of steganography models

The capacity, or the amount of information a model can hide in an image, is a critical metric for evaluating the network performance. This capacity is influenced by the network architecture, vulnerability to noise distortion, and the design of loss functions. Normalizing models like HiNet achieve the highest capacity at 120 bpp, setting the baseline for a benchmark comparison. LSB (0.2 bpp) and DeepStega (0.1 bpp) follow, with lower capacities than the normalizing flow models.

CodeFace (Shadmand et al., 2021) and the StegaStamp (Tancik et al., 2020) model demonstrate a much smaller capacity, retrieving 0.21×10^{-3} bpp and 0.13×10^{-3} bpp, respectively, when recovering hyperlinks after applying error-correcting algorithms. Although their capacities (regarding the maximum amount of hidden information) are lower, these models are specifically designed to withstand distortions caused by image transfer through physical media, such as printing and re-digitization, making their printer-proof characteristic particularly valuable.

Table 1: Qualitative Results: Model Ranking Based on Scores (0-10)

Model	Score (0-10)
LSB	8.89
HiNet	8.69
StylePuncher (Ours)	8.68
CodeFace	6.89
DeepStega	6.05
StegaStamp	4.08

Finally, the StylePuncher model, while maintaining robustness against various noise distortions, significantly improves upon these robust models with a capacity of 0.52×10^{-2} bpp, making a substantial enhancement in the field of resilient steganography methods.

6.3 Encoder performance

Fidelity, the quality of the encoded image and its similarity to the original, is a key measure of encoder performance. Human judgment remains the most reliable method for evaluating encoder effectiveness, particularly across different steganography models. As shown in Figure 5, we conducted a survey based on the ITU-R BT.500.11 recommendation (BT, 2002) and (Zhai and Min, 2020), commonly used for image quality assessments, such as by the JPEG working group (ISO/IEC JTC 1/SC 29/WG 1).

In a survey of 76 participants, eight pairs of original and encoded images from six steganography models were evaluated for similarity. Based on qualitative scores (0–10) and the Mean Opinion Score (MOS), as presented in Table 1. StylePuncher ranked slightly below HiNet and LSB, which are highly noise-sensitive. Among robust models, however, StylePuncher was the top performer in encoded image quality.

Additionally, Certain loss functions, such as SSIM (Chen and Bovik, 2011), LPIPS (Zhang et al., 2018), and eLPIPS (Kettunen et al., 2019), quantify perceptual image similarity, aligning with human judgment. These metrics are particularly useful for evaluating steganography models' fidelity.

However, the perceptual loss functions used in our evaluation are not fully rigorous in accurately measuring human visual perception (Avanaki et al., 2024). In fact, these functions suggest that DeepStega-encoded images offer better perceptual quality than StylePuncher, CodeFace, and StegaStamp. However, in reality, DeepStega-encoded images undergo significant color transformations, which are not effec-

Table 2: Global Encoder Performance Based on Average Error Scores

Model	Average Error Score
LSB	0.004
HiNet	0.009
StylePuncher (Ours)	0.065
CodeFace	0.071
DeepStega	0.103
StegaStamp	0.161

tively captured by these loss functions. To address this limitation, we introduced a more comprehensive approach by measuring the color histogram distance (Afifi et al., 2021) between the encoded and original images, as shown in the final (right) plot of Figure 6. This method provides a more accurate reflection of color fidelity in the encoded images.

To further improve the quality of the encoded images in the context of face images, we also measure the distance of face features between encoded and original images, by using the ArcFace model (Deng et al., 2019), which is based on the cosine distance.

The results of the comparison of all loss functions for all models are presented in Figure 6. We computed the average from the five mentioned error functions for every model as the global encoder performance, which is shown in Table 2. LSB (0.004) and HiNet (0.009) have the best results. After them, for the robust models, come StylePuncher (0.065) and CodeFace (0.071) that then have the best encoded images. The worst results were achieved for DeepStega (0.103) and StegaStamp (0.161). As can be seen, the scores obtained by the human judgement survey are in line with these metrics.

6.4 Decoder performance

The decoder's performance is evaluated based on its ability to accurately recover hidden messages under various distortions and noise simulations. Its robustness for real-world printed images is enhanced by applying perturbation, noise, or self-attack simulations, as outlined in Section 5. This subsection examines the model's sensitivity to several distortions, including hue adjustment, JPEG compression, random contrast, random brightness, resolution changes, linear interpolation, Gaussian noise, and arbitrary rotations. This analysis assesses the decoder's effectiveness in preserving message integrity under challenging conditions.

The noise resistance ratio (d_{noise}) is used to evaluate the decoder's performance, measuring its ability to recover encoded messages under noise perturbations. The metric is defined as:

$$d_{noise} = \frac{\triangle w}{\triangle W} \tag{5}$$

where $\triangle w$ represents the range of noise intensity levels within which the decoder successfully recovers the message, and $\triangle W$ is the total possible range of noise intensities. This ratio directly quantifies the decoder's robustness against various noise distortions, facilitating a clear comparison of its resistance capabilities across different noise levels.

For example, in the case of JPEG compression, the quality factor varies from 0 to 100 ($\triangle W = 100 - 0$). The contrast factor varies from 0 to 1.0. The brightness value varies between -1 for complete darkness and 1 for maximum brightness. The Linear interpolation of pixels between encoded and background images varies from 0 to 1.0 (see Eq. 3). As the maximum resolution that the decoder network can get is $512 \times 512 \times 3$, the resolution value varies between $1 \times 1 \times 3$ and $512 \times 512 \times 3$ (the resolution value is variable for each model according to the input size). The standard deviation of Gaussian noise varies between 0 and 0.90. We also consider random rotation which we consider that varies between 0.0001 and 0.5 radians. Finally, the full decoder performance ($D_{decoder}$) of every model is computed by averaging the results of every model in the presence of every type of noise (d_{noise}). The performances of the four selected network structures are summarized in Table 4. A standard QR code reader, due to built-in redundancy, can correctly interpret QR code messages with up to 40% distortion. Accordingly, we accept recovered messages with a maximum of 30% binary cross-entropy error (BCEE), remaining safely below the maximum acceptable distortion threshold. When all four models were trained with various noise types, the U-Net network with Spatial Transformer Network (STN) demonstrated the best performance, while the U-Net with a discriminator (pix2pix) showed the weakest results among the four. However, none of the models were capable of decoding the message from encoded images with slight rotations exceeding 0.001 radians.

Following the identification of U-Net+STN as the best-performing decoder network through ablation studies, the network was further trained with random noise simulations. Initially, training was conducted over 10^5 epochs, introducing JPEG compression, contrast, and brightness noise incrementally. The noise levels were applied to the encoded images within the following intervals: JPEG compression ranged from 25 to 60, brightness noise varied between -0.95 and -0.95, and contrast noise was adjusted within the range of 0.050.05 to (0.05, 0.20).

Additional noise types, including random resolution, linear interpolation, Gaussian noise, and random rotation, were sequentially added to the pre-

Table 3: StylePuncher decoder performance (four different decoder models are considered)

Noise	U-Net	U-Net +discriminator	U-Net +attention	U-Net+STN
JPEG Compression	0.42	0.22	0.14	**0.65**
Contrast	0.6	0.5	0.65	**0.85**
Brightness	0.55	0.45	0.47	**0.87**
Linear interpolation	0.3	0.25	0.29	**0.4**
Different resolution	0.19	0.01	0.01	**0.52**
Gaussian noise	0.13	0.13	0.15	**0.13**
Decoder performance	0.36	0.26	0.28	**0.57**

Table 4: The best-elected decoder of StylePuncher is compared with two robust steganography models: CodeFace and StegaStamp.

Noise	StegaStamp	CodeFace	StylePuncher (Ours)	StylePuncher (Ours) with noise simulation
JPEG Compression	**0.93**	0.96	0.65	0.87
Contrast	0.90	0.90	0.85	**0.95**
Brightness	0.87	0.70	0.87	**0.99**
Linear interpolation	0.01	0.01	0.4	**0.96**
Different resolution	**0.86**	**0.86**	0.52	0.70
Gaussian noise	0.62	0.28	0.13	**0.80**
Rotation	0.5	0.5	0.01	**0.67**
Decoder performance	0.67	0.60	0.49	**0.85**

trained network, with each type trained for 70×10^4 epochs. This incremental training with increasingly complex noise simulations improved the decoder's performance from 0.49 to 0.83, as shown in Table 3.

Table 3 compares our best model (U-Net+STN, with and without noise simulation) to state-of-the-art methods. While HiNet and LSB excel in capacity and encoder performance, their decoders are highly noise-sensitive, with a performance of zero under noisy conditions. DeepStega achieves a robustness rate of 0.05 under rotation but shows zero performance against other noise types.

CodeFace and StegaStamp, designed with noise simulation for printer-proof robustness, perform well under JPEG compression and varying resolution. However, our StylePuncher model, also trained with noise simulation, outperforms all others when exposed to multiple noise types. StylePuncher achieves the highest decoder performance, with an 85% success rate under noise conditions, setting a new benchmark among steganography models.

6.5 Discussion

The StylePuncher model surpasses existing steganography methods with key advantages. It demonstrates exceptional robustness against noise and distortions, including JPEG compression, brightness and contrast variations, Gaussian noise, and arbitrary rotations. This resilience is achieved through noise simulations during training, enabling the model to handle real-world perturbations, such as printer-proof scenarios. Furthermore, StylePuncher is computationally efficient, allowing separate training of the encoder and decoder, which reduces GPU usage and accelerates training. Its lightweight encoder architecture, with fewer layers than models like CodeFace and StampOne, further enhances efficiency.

StylePuncher excels in fidelity by maintaining high visual quality in encoded images, preserving their similarity to the originals even with embedded data—an essential feature for steganography applications. The model employs perceptual loss functions and an accurate color histogram distance metric, ensuring that encoded images remain visually indistinguishable from the originals to the human eye.

7 Conclusion

In this work, we present StylePuncher, a robust steganography model uniquely resistant to various physical noises. It outperforms other printer-proof methods, such as CodeFace and StegaStamp, in hidden message capacity. The encoder produces significantly higher-quality encoded images compared to other robust models. Among the four decoder structures explored, experiments identified the U-Net combined with a Spatial Transformer Network (STN) as the most effective configuration.

Future work on StylePuncher will focus on improving its robustness in decoding messages from printed images affected by stronger perturbations, such as scratches or significant color changes. This can be achieved by integrating a frequency balance method into the style transfer process, enhancing the model's resilience to distortions introduced during printing.

REFERENCES

Abdal, R., Qin, Y., and Wonka, P. (2019). Image2stylegan: How to embed images into the stylegan latent space? In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4432–4441.

Afifi, M., Brubaker, M. A., and Brown, M. S. (2021). Histogan: Controlling colors of gan-generated and real images via color histograms. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7941–7950.

Arjovsky, M., Chintala, S., and Bottou, L. (2017a). Wasserstein generative adversarial networks. In *International Conference on Machine Learning*, pages 214–223. PMLR.

Arjovsky, M., Chintala, S., and Bottou, L. (2017b). Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR.

Avanaki, N. J., Ghildiyal, A., Barman, N., and Zadtootaghaj, S. (2024). Lar-iqa: A lightweight, accurate, and robust no-reference image quality assessment model. *arXiv preprint arXiv:2408.17057*.

Baluja, S. (2017). Hiding images in plain sight: Deep steganography. *Advances in Neural Information Processing Systems*, 30:2069–2079.

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

BT, R. I.-R. (2002). Methodology for the subjective assessment of the quality of television pictures. *International Telecommunication Union*.

Chen, M.-J. and Bovik, A. C. (2011). Fast structural similarity index algorithm. *Journal of Real-Time Image Processing*, 6(4):281–287.

Cunha, T., Schirmer, L., Marcos, J., and Gonçalves, N. (2024). Noise simulation for the improvement of training deep neural network for printer-proof steganography. In *Proceedings of the 13th International Conference on Pattern Recognition Applications and Methods*, pages 179–186.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. IEEE.

Deng, J., Guo, J., Xue, N., and Zafeiriou, S. (2019). Arcface: Additive angular margin loss for deep face recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4690–4699.

Deng, J., Guo, J., Xue, N., and Zafeiriou, S. (2019). ArcFace: Additive Angular Margin Loss for Deep Face Recognition. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4685–4694.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. C., and Bengio, Y. (2014). Generative adversarial nets. In *NIPS*.

Hancock, P. (2008). Psychological image collection at stirling (pics). *Web address: http://pics. psych. stir. ac. uk.*

Hsu, C.-T. and Wu, J.-L. (1999). Hidden digital watermarks in images. *IEEE Transactions on Image Processing*, 8(1):58–68.

Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1125–1134.

Jaderberg, M., Simonyan, K., Zisserman, A., et al. (2015). Spatial transformer networks. In *Advances in Nneural Information Processing Systems*, pages 2017–2025.

Jing, J., Deng, X., Xu, M., Wang, J., and Guan, Z. (2021). Hinet: Deep image hiding by invertible network. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4733–4742.

Kettunen, M., Härkönen, E., and Lehtinen, J. (2019). E-lpips: robust perceptual image similarity via random transformation ensembles. *arXiv preprint arXiv:1906.03973*.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Liu, Z., Luo, P., Wang, X., and Tang, X. (2018). Large-scale celebfaces attributes (celeba) dataset. *Retrieved August*, 15(2018):11.

Mirza, M. and Osindero, S. (2014). Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*.

Oktay, O., Schlemper, J., Folgoc, L. L., Lee, M., Heinrich, M., Misawa, K., Mori, K., McDonagh, S., Hammerla, N. Y., Kainz, B., et al. (2018). Attention u-net: Learning where to look for the pancreas. *arXiv preprint arXiv:1804.03999*.

O'Ruanaidh, J., Dowling, W., and Boland, F. (1996). Watermarking digital images for copyright protection. *IEE Proceedings-Vision, Image and Signal Processing*, 143(4):250–256.

Phillips, P. J., Moon, H., Rizvi, S. A., and Rauss, P. J. (2000). The feret evaluation methodology for face-recognition algorithms. *IEEE Transactions on pattern analysis and machine intelligence*, 22(10):1090–1104.

Shadmand, F., Medvedev, I., and Goncalves, N. (2021). Codeface: A deep learning printer-proof steganography for face portraits. *IEEE Access*, 9:167282–167291.

Shadmand, F., Medvedev, I., Schirmer, L., Marcos, J., and Gonçalves, N. (2024). Stampone: Addressing frequency balance in printer-proof steganography. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4367–4376.

Su, H., Niu, J., Liu, X., Li, Q., Wan, J., Xu, M., and Ren, T. (2021). Artcoder: An end-to-end method for generating scanning-robust stylized qr codes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2277–2286.

Tamimi, A. A., Abdalla, A. M., and Al-Allaf, O. (2013). Hiding an image inside another image using variable-rate steganography. *International Journal of Advanced Computer Science and Applications (IJACSA)*, 4(10).

Tancik, M., Mildenhall, B., and Ng, R. (2020). Stegastamp: Invisible hyperlinks in physical photographs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2117–2126.

Wang, C.-Y., Bochkovskiy, A., and Liao, H.-Y. M. (2021). Scaled-yolov4: Scaling cross stage partial network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13029–13038.

Wang, T.-C., Liu, M.-Y., Zhu, J.-Y., Tao, A., Kautz, J., and Catanzaro, B. (2018). High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8798–8807.

Wang, Y. and Solomon, J. M. (2019). Prnet: Self-supervised learning for partial-to-partial registration. In *33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada*, pages 8814–8826.

Zhai, G. and Min, X. (2020). Perceptual image quality assessment: a survey. *Science China Information Sciences*, 63(11):211301.

Zhang, R., Dong, S., and Liu, J. (2019). Invisible steganography via generative adversarial networks. *Multimedia Tools and Applications*, 78(7):8559–8575.

Zhang, R., Isola, P., Efros, A. A., Shechtman, E., and Wang, O. (2018). 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.

Zhu, J., Kaplan, R., Johnson, J., and Fei-Fei, L. (2018). Hidden: Hiding data with deep networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 657–672.