electronics

MDPI

Article

A Steganographic Message Transmission Method Based on Style Transfer and Denoising Diffusion Probabilistic Model

Yen-Hui Lin [1], Chin-Pan Huang [1] and Ping-Sheng Huang [2,*]

[1] Department of Applied Artificial Intelligence, Ming Chuan University, Taoyuan 333, Taiwan; 11166016@me.mcu.edu.tw (Y.-H.L.); hcptw@mail.mcu.edu.tw (C.-P.H.)
[2] Department of Electrical Engineering, Ming Chuan University, Taoyuan 333, Taiwan
* Correspondence: pshuang@mail.mcu.edu.tw

Abstract

This study presents a new steganography method for message transmission based on style transfer and denoising diffusion probabilistic model (DDPM) techniques. Different types of object images are used to represent the messages and are arranged in order from left to right and top to bottom to generate a secret image. Then, the style transfer technique is employed to embed the secret image (content image) into the cover image (style image) to create a stego image. To reveal the messages, the DDPM technique is first used to inpaint the secret image from the stego image. Then, the YOLO (You Only Look Once) technique is utilized to detect objects in the secret image for the message decoding. Two security mechanisms are included: one uses object images for the message encoding, and the other hides them in a customizable public image. To obtain the messages, both mechanisms need to be cracked at the same time. Therefore, this method provides highly secure information protection. Experimental results show that our method has good confidential information transmission performance.

Keywords: denoising diffusion probabilistic model; message transmission; steganography; style transfer; YOLO

check for updates

Academic Editors: Fabio Mendonca, Morgado Dias and Sheikh Shanawaz Mostafa

Received: 25 June 2025
Revised: 13 August 2025
Accepted: 14 August 2025
Published: 16 August 2025

Citation: Lin, Y.-H.; Huang, C.-P.; Huang, P.-S. A Steganographic Message Transmission Method Based on Style Transfer and Denoising Diffusion Probabilistic Model. *Electronics* **2025**, *14*, 3258. https://doi.org/10.3390/electronics14163258

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

With how convenient the internet has become, we almost forget how much the internet has changed our daily life. Through the shortening of distances the internet has turned online information sharing into second nature. But this convenience comes with risk, and data security is a real concern. Even if a connection seems secure, sensitive information like ID numbers or ATM passwords can still be intercepted by someone with the wrong intentions. This is why steganography is receiving more attention. Steganography is a technique in which a message is embedded inside another type of media such that it is difficult to identify or detect the existence of the message.

Lately, due to advances in artificial intelligence, image steganography [1] has made major strides. End-to-end image-to-image translation allows content from one image to be embedded into another, which can be regarded as a type of image hiding problem [2]. Style transfer can take the content features from a secret image and embed them into the style features of a cover image. The final image looks like the cover, but the message is secretly carried. On the decoding side, extracting this hidden information can be seen as a type of image restoration task [3]. The objective is to accurately reconstruct the original secret image from the stego image through inpainting.

In this paper, we propose a new steganography method for message transmission. We combine cryptography with image steganography and use an image as a kind of secret key. At the encoding stage, we apply style transfer, then adjust the balance between style and content features to produce the stego image. At the decoding stage, we regard the image restoration problem as global image inpainting and use a denoising diffusion probabilistic model (DDPM) [4] to gradually recover the secret image from the stego image. The advantage of the proposed method is the mechanism of built-in double protection. First, the message is turned into an image made up of objects. Second, the image is hidden inside a public image of our choosing. Unless someone cracks both layers, the message stays safe. Even if the stego image is detected or partially compromised, the secret information cannot be decoded without the correct image key. This makes our method much more secure and robust than traditional approaches. The main contributions of this paper are as follows:

1. We propose a dual-layer protection method. The message is encoded using object-based images and then hidden inside a customizable cover image.
2. Inspired by conditional models, we are the first to use DDPM by regarding the image restoration problem as global image inpainting, to recover the original secret image from the stego image.

The rest of this paper is organized as follows: Section 2 introduces related work. Section 3 provides background information relevant to the proposed approach. In Section 4, the proposed steganographic system is described, including the algorithm and overall workflow. Section 5 presents the experimental results, and finally, Section 6 concludes the study with a discussion on future research directions.

2. Related Work

2.1. Steganography Techniques

Transmitting information over the internet is never entirely secure. There is always a risk that messages could be intercepted or stolen. Many researchers have started treating confidential communication as a data hiding problem. Bender et al. [5] explored a variety of traditional data hiding techniques. Their focus was on how much data could be embedded, how much distortion the image would suffer, and how well the hidden data could survive if a third party tried to intercept, alter, or delete it. Lin and Delp [6] introduced a fragile image watermark system that can detect whether the image has been tampered with. Any modification to the image, even if minor, would destroy the watermark. Compared to other methods, this approach is simpler and makes it harder to remove the watermark. However, it is extremely sensitive. Later, Kumar and Pooja [7] proposed a combination of steganography and cryptography. A message encrypted by cryptography is unreadable without the correct password or key. Steganography hides the message inside the image. Putting these two together makes for much stronger defense. Huang and Hsieh [8] reported on an image embedding technique based on Wavelet Transform Modulation Representation (WTMR). They made use of the spatial similarity of wavelet transforms and modulated the hidden information in the smooth sub-band, which refers to the low-frequency component containing the approximation coefficients that represent the general structure and smooth variations of the image. By adjusting characteristics according to the mean and variance, they were able to effectively embed and recover the hidden message in the transform domain.

With the rapid rise of deep learning, several recent studies have started using neural networks for steganographic image generation. One of the earliest was proposed by Rahim et al. [9], who introduced a simple encoder–decoder structure based on Convolutional Neural Networks (CNNs). Their setup only embedded a one-dimensional black-and-white image to ensure reliable recovery. As network architecture improved, Baluja [10] used

a deeper network that could fully hide a three-channel RGB image. His approach was split into three parts. The Prep-Network converts the secret image into features better suited for encoding. The Hiding Network takes both the processed secret and the original cover image to generate the stego image. The Reveal Network extracts the hidden content from the stego image. Generative models have gained popularity in image synthesis and detection. Building on the CNN-based approaches, researchers have explored the integration of steganography with style transfer techniques. Zheng et al. [11] proposed an image steganography method based on style transfer, which leverages the artistic transformation process to conceal secret information. Their approach takes advantage of the natural distortions introduced during style transfer to mask the presence of hidden data. Zhang et al. [12] further conducted research on robust image steganography for arbitrary style transfer. Their work addresses the challenge of maintaining hidden information integrity when images undergo various style transformations, ensuring that secret data remains recoverable even after different style transfer processing. Generative Adversarial Networks (GANs) [13] have been used to generate stego images that are harder to detect, while still allowing the secret image to be recovered. Volkhonskiy et al. [14] proposed two new approaches: Steganographic GAN (SGAN) and Steganographic Encryption GAN (SEGAN). SGAN combines a generator, discriminator, and steganalyzer, using multi-adversarial training to produce stego images that are extremely hard to detect. SEGAN further integrates both message embedding and extraction, enabling full image encryption and recovery within the network. Fu et al. [15] introduced the HIGAN architecture, which can hide a full-color image inside a cover image. Using adversarial training, this approach significantly improves both stealth and decoding accuracy. Their method highlights the growing potential of GAN-based systems for high-capacity steganography.

2.2. Denoising Diffusion Probabilistic Models

The basic principle of diffusion models operates through sequential forward diffusion steps that introduce noise to data, followed by reverse denoising steps, which learn to eliminate noise progressively. Ho et al. [4] presented the basic DDPM framework, which describes a Markov chain of diffusion steps that progressively destroy structure in data through Gaussian noise addition. The reverse process can denoise images through neural networks trained to predict the noise added at each step.

Song et al. [16] expanded this research through Score-based Generative Models, which establish a broader theoretical framework by linking diffusion models to score matching principles. The image quality produced by diffusion models has improved substantially through multiple architectural and training advancements. Dhariwal and Nichol [17] showed that diffusion models generate better samples than GANs in image synthesis benchmarks using classifier guidance. The method trains a classifier on noisy images, and then the gradients are used to direct the sampling process toward specific classes. The latest research has concentrated on enhancing both sampling efficiency and controllability. Song et al. [18] developed Denoising Diffusion Implicit Models (DDIMs) to achieve deterministic sampling with reduced step requirements while maintaining high-quality results. Rombach et al. [19] developed Latent Diffusion Models, which operate in pre-trained autoencoder latent spaces to achieve lower computational costs without compromising generation quality.

2.3. Image-to-Image Translation

2.3.1. Style Transfer

Style transfer is an important area of research in both artificial intelligence and image processing. The technique initially uses Convolutional Neural Networks (CNNs) to transfer

the visual style of one image onto arbitrary photographs, producing striking and artistic results. A key aspect of style transfer is how to define and capture the style of an image. Gatys et al. [2] resolved this question by introducing the use of the Gram matrix to represent artistic style. They adopted the VGG19 network [20], which includes 16 convolutional layers and three fully connected layers. Only the convolutional layers were kept for feature extraction. The Gram matrix is computed by taking the inner product between feature maps generated at each channel of the VGG19 network. Lower layers capture texture details, while deeper layers extract abstract structure and shape. This allows the Gram matrix to reflect the correlation between different feature maps and to describe the style of an image. Building on this research, Johnson et al. [21] proposed a real-time style transfer method. Their approach uses a pre-trained CNN to extract high-level features and optimizes the transformation by comparing these features when passed through a VGG network. When the training is finished, the model can immediately produce stylized images, eliminating the need to repeatedly optimize the image during each transformation. Later, Mallika et al. [22] applied style transfer to the field of steganography. They made use of the characteristics of the VGG19 network architecture and adjusted the balance between content loss and style loss. By experimenting with different layer combinations, they were able to generate stego images that visually match the cover image's style while embedding secret content.

More robust GAN architectures have been proposed for style transfer. Zhu et al. [23] introduced CycleGAN, which allows for unpaired image-to-image translation. A cycle-consistency loss is adopted to ensure that the original content feature is preserved during style transformation. This enables the model to transfer images across domains without needing matched pairs. Chen et al. [24] proposed a dual style-learning network. By incorporating an attention mechanism, the model learns both content and style features. The model can emphasize salient regions, thereby enhancing the transferred image's ability to capture the colors and textures of the target image style.

2.3.2. Image Inpainting

The process of image inpainting involves restoring missing and damaged sections that appear within an image. This task needs knowledge about image semantic content and surrounding texture structure to produce suitable pixels for image completion. The first image restoration approaches depended on mathematical models together with statistical methods. The PDE-based approach proposed by Bertalmio et al. [25] conducts image restoration by letting pixel values move from the mask borders toward the center. Pathak et al. [26] introduced Context Encoder as a groundbreaking CNN-based encoder-decoder system, which learns context-aware features from images to predict missing regions. Liu et al. [27] introduced the Partial Convolution technique, which focuses on filling irregularly shaped holes, improving the flexibility and effectiveness of image inpainting. The conditional image generation model pix2pix, which Isola et al. [28] developed, uses conditional GANs (cGANs). The approach uses PatchGAN discriminator evaluations of local regions with input images and noise to achieve better fine detail generation in output images. Iizuka et al. [29] introduced a system that uses dual-discriminator architecture to perform inpainting of extensive missing areas. The system employs two discriminators, which evaluate the image: one local discriminator checks the consistency near missing areas while the global discriminator examines semantic coherence throughout the entire image.

Saharia et al. [30] introduced diffusion-based methods to solve super-resolution problems. The model used conditional diffusion to accept low-resolution images, which are then denoised progressively to create high-resolution output. The research by Lugmayr et al. [31] demonstrates an unconditional pre-trained diffusion model that functions without

task-specific retraining. The method executes complete denoising operations on the entire image, followed by restoring known areas back to their original state. This process leads to both efficient hole fixing and immediate output generation. The conditional diffusion model proposed by Saharia et al. [3] uses a straightforward method by combining the input image directly with the noisy image. Direct injection serves as an easy-to-use approach that enables the model to discover the relationship between conditions and target images throughout training.

2.4. Object Detectors

Object detection algorithms based on CNNs fall into two primary categories. The R-CNN family represents the two-stage approach, which includes R-CNN [32], Fast R-CNN [33], and Faster R-CNN [34]. The models generate regional proposals before executing classification and bounding box regression on each region. The YOLO (You Only Look Once) [35] and SSMD (Single Shot MultiBox Detector) [36] operate as one-stage detectors because they produce object class predictions and location outputs during a single processing step. Real-time applications benefit from their speed because these detectors operate directly without needing multiple passes.

The first YOLO model of object detection was presented by Redmon et al. in 2016 [35], before becoming one of the leading real-time object detection research directions. The unique detection framework is produced from YOLO's single regression approach. The input image is divided into a grid structure by YOLO, which enables each grid cell to predict both bounding boxes and class probabilities simultaneously. The YOLO family has developed through multiple versions, which have enhanced both speed and accuracy performance. YOLOv2 [37] introduced batch normalization and multi-scale training. YOLOv3 [38] implemented a multi-scale prediction system that used three different-sized feature maps to enhance small object detection capabilities. YOLOv4 [39] implemented CSP-Darknet53 alongside SPP and PANet to reach higher accuracy levels without performance degradation. YOLOv7 [40] implemented E-ELAN and trainable bag-of-freebies strategies to enhance network architecture optimization. The current version offers excellent precision while maintaining real-time performance capabilities.

3. Background

3.1. Denoising Diffusion Probabilistic Model

A Denoising Diffusion Probabilistic Model (DDPM) is a state-of-the-art image generation model that uses a Markov chain to model both the forward and reverse process of noise transformation. A Markov chain is a statistical model in which the system changes between states over time, and the next state depends only on the current state. The architecture of a diffusion model is shown in Figure 1, which is divided into two main stages, the forward propagation, in which noise is gradually added to an image, and the backward propagation, in which the image is inpainted by reversing the process.

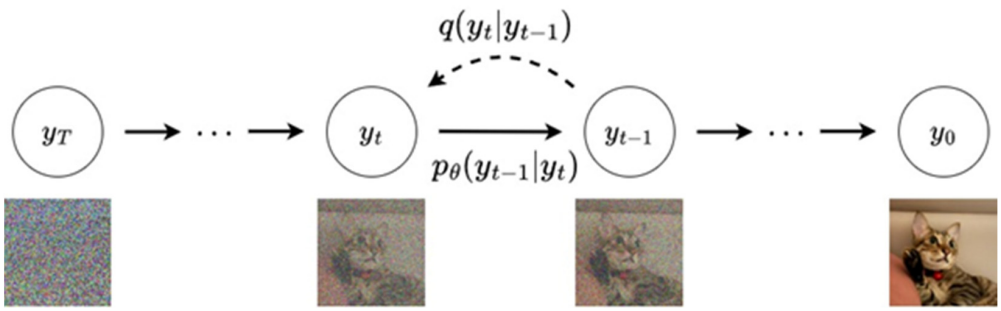


Figure 1. Diffusion model architecture.

In the forward process, we treat the diffusion model as a Markov chain. Starting with an input image $\boldsymbol{y}_0$, the process adds noise step by step over T steps, eventually producing a highly degraded image $\boldsymbol{y}_T$. At each time step t, Gaussian noise is sampled and added to the previous output $\boldsymbol{y}_{t-1}$ to generate $\boldsymbol{y}_t$. This can be formally defined as

$$q(\boldsymbol{y}_{0,1,\cdots,T}|\boldsymbol{y}_0) = \prod_{t=1}^{T} q(\boldsymbol{y}_t|\boldsymbol{y}_{t-1}) \tag{1}$$

where $q(\boldsymbol{y}_t|\boldsymbol{y}_{t-1})$ is the conditional probability of getting $\boldsymbol{y}_t$ given $\boldsymbol{y}_{t-1}$. The joint probability of the entire forward process is computed by multiplying the conditional probabilities across all steps. Each step's conditional probability is modeled as a normal distribution:

$$q(\boldsymbol{y}_t|\boldsymbol{y}_{t-1}) = \boldsymbol{\mathcal{N}}\left(\boldsymbol{y}_t; \sqrt{1-\beta_t}\boldsymbol{y}_{t-1}, \beta_t \boldsymbol{I}\right) \tag{2}$$

In this equation, $\boldsymbol{\mathcal{N}}$ is a Gaussian distribution, and β_t controls how much noise is added at each step. The term $\sqrt{1-\beta_t}$ functions as a decay factor, determining how much of the previous state $\boldsymbol{y}_{t-1}$ is retained. The identity matrix $\boldsymbol{I}$ ensures that noise is added independently across all dimensions, meaning each dimension receives uncorrelated Gaussian noise with a variance of β_t. $\beta_t \boldsymbol{I}$ can be viewed as a diagonal covariance matrix, in which all off-diagonal elements are zero and the diagonal elements are equal to β_t. This formulation allows precise control of noise injection at each time step, as follows:

$$\boldsymbol{\beta_t I} = \begin{bmatrix} \boldsymbol{\beta_t} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\beta_t} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \boldsymbol{\beta_t} \end{bmatrix} \tag{3}$$

By adjusting the parameter β_t, we can control how quickly the original signal decays and how much noise is introduced. When β_t is close to 1, $\sqrt{1-\beta_t}$ approaches 0, meaning the original information is almost completely lost and the resulting output is mostly noise.

The amount of noise added during each timestep in diffusion models follows the description of Equation (2). The distribution of noisy data probability $\boldsymbol{y}_t$ at any timestep t when given the original input $\boldsymbol{y}_0$ can be defined as

$$q(\boldsymbol{y}_t|\boldsymbol{y}_0) = \boldsymbol{\mathcal{N}}\left(\boldsymbol{y}_t; \sqrt{\overline{\alpha}_t}\boldsymbol{y}_0, (1-\overline{\alpha}_t)\boldsymbol{I}\right) \tag{4}$$

The mean and variance of the Gaussian distribution are represented by $\sqrt{\overline{\alpha}_t}\boldsymbol{y}_0$ and $(1-\overline{\alpha}_t)\boldsymbol{I}$, respectively, in this expression. The formulation requires defining $\gamma_t = (1-\beta_t)$ and $\overline{\alpha}_t = \prod_{s=1}^{t} \gamma_s$, which representthe cumulative product of noise retention factors up to timestep t. The value of $\sqrt{\overline{\alpha}_t}$ indicates how much of the original signal $\boldsymbol{y}_0$ remains intact. The system state approaches zero as t grows because the value of $\sqrt{\overline{\alpha}_t}$ decreases, which reduces the influence of the original signal $\boldsymbol{y}_0$. The variance term $(1-\overline{\alpha}_t)\boldsymbol{I}$ expands as the sample becomes increasingly influenced by random noise. The resulting posterior distribution takes the following form:

$$q(\boldsymbol{y}_{t-1}|\boldsymbol{y}_0, \boldsymbol{y}_t) = \boldsymbol{\mathcal{N}}\left(\boldsymbol{y}_{t-1}; \mu, \sigma^2 \boldsymbol{I}\right) \tag{5}$$

The posterior distribution is given by $\mu = \frac{\sqrt{\overline{\alpha}_{t-1}}(1-\gamma_t)}{1-\overline{\alpha}_t}\boldsymbol{y}_0 + \frac{\sqrt{\gamma_t}(1-\overline{\alpha}_{t-1})}{1-\overline{\alpha}_t}\boldsymbol{y}_t$ and $\sigma^2 = \frac{(1-\overline{\alpha}_{t-1})(1-\gamma_t)}{1-\overline{\alpha}_t}\boldsymbol{y}_t$. The mean and variance of the posterior distribution are denoted by μ and σ^2, respectively. The posterior distribution unites information from both the forward

process and prior knowledge of the original image $\boldsymbol{y}_0$ to provide necessary information for reverse denoising. The forward process distribution q is used instead of the true generative process p because the exact posterior distribution proves difficult to calculate in real-world scenarios. The optimization of an approximate posterior $q(\boldsymbol{y}_{t-1}|\boldsymbol{y}_0, \boldsymbol{y}_t)$ aims to achieve close similarity to the actual posterior distribution.

The reverse diffusion process also follows a Gaussian distribution. A U-Net network [41] is used to predict the amount of noise added during the forward diffusion. The U-Net architecture was first proposed in 2015 for medical image segmentation, and the architecture is symmetric, with a contracting path on the left for feature extraction and an expansive path on the right that adapts to different output tasks. To compensate for information loss during down-sampling, residual connections are added between corresponding layers of the encoder and decoder to help preserve fine details. The reverse process is defined as follows:

$$p(\boldsymbol{y}_{t-1}|\boldsymbol{y}_t) = \boldsymbol{\mathcal{N}}\left(\boldsymbol{y}_{t-1}; \mu_\theta(\boldsymbol{y}_t, t), \beta_t \boldsymbol{I}\right) \tag{6}$$

This equation describes the conditional probability of recovering the previous state $\boldsymbol{y}_{t-1}$ given the current noisy state $\boldsymbol{y}_t$. The mean μ_θ is predicted by a U-Net neural network, which learns to estimate the noise added in the forward process. The parameter θ represents the trainable weights of the network, which are optimized during training. To train the model, we simulate noisy inputs using Equation (4), as follows:

$$\boldsymbol{z} = \sqrt{\overline{\alpha}_t}\boldsymbol{y}_t + \epsilon\sqrt{1-\overline{\alpha}_t}, \epsilon \sim \boldsymbol{\mathcal{N}}\left(0, \boldsymbol{I}\right) \tag{7}$$

In this equation, the image $\boldsymbol{y}_t$ is scaled by $\sqrt{\overline{\alpha}_t}$, and the noise ϵ is sampled independently from a standard Gaussian distribution $\boldsymbol{\mathcal{N}}(0, \boldsymbol{I})$. To ensure the noise has a variance of $1 - \overline{\alpha}_t$, as in Equation (4), the noise is scaled by $\sqrt{1-\overline{\alpha}_t}$. Since the variance is the square of the standard deviation, the square root factor ensures that the correct amount of noise is injected. The loss function used to train the network is defined as

$$E_{(\boldsymbol{y}_t, \epsilon)} = \|\mu_\theta(\boldsymbol{z}) - \epsilon\|_2 \tag{8}$$

Here, the network μ_θ receives the noisy image $\boldsymbol{z}$ and aims to predict the exact noise ϵ that was originally added. Minimizing this error helps the model learn to denoise and recover images from noisy inputs.

3.2. Conditional Diffusion Probabilistic Models

The Conditional Diffusion Probabilistic Model extends the standard denoising diffusion model by incorporating an additional conditioning image. The forward diffusion process generates a noisy image, which is then used together with the conditioning input $\boldsymbol{x}$ as inputs. The goal remains identical, to find an optimal denoising network that predicts the added noise at each time step. The forward diffusion process remains the same, but the reverse process receives modified input from the conditional information, which can be expressed as

$$p(\boldsymbol{y}_{t-1}|\boldsymbol{y}_t, \boldsymbol{x}) = \boldsymbol{\mathcal{N}}\left(\boldsymbol{y}_{t-1}; \mu_\theta(\boldsymbol{y}_t, \boldsymbol{x}, t), \beta_t \boldsymbol{I}\right) \tag{9}$$

The main distinction occurs through the addition of the conditioning input $\boldsymbol{x}$, which acts as an auxiliary training input for the network. The conditioning input serves as semantic guidance to help the model generate more precise outputs.

4. Methodology

In the proposed method, we present the detailed workflow and training methodology. The proposed model framework is displayed in Figure 2. Message encoding represents the first step, which transforms messages into sequences of corresponding object images. The images are arranged in a square secret image format with equal width and height by following a left-to-right and top-to-bottom sequence. The embedding stage follows the first stage by uniting the secret image with the cover image through style transfer to produce the stego image. The third stage is secret image inpainting, in which a conditional diffusion model learns to extract the inpainted secret image from the stego image. The message decoding process starts with YOLO object detection for sequential object identification, followed by message decoding to obtain the original message. We obtained object images for message encoding from the Roboflow Universe [42] dataset to establish an appropriate object image library.

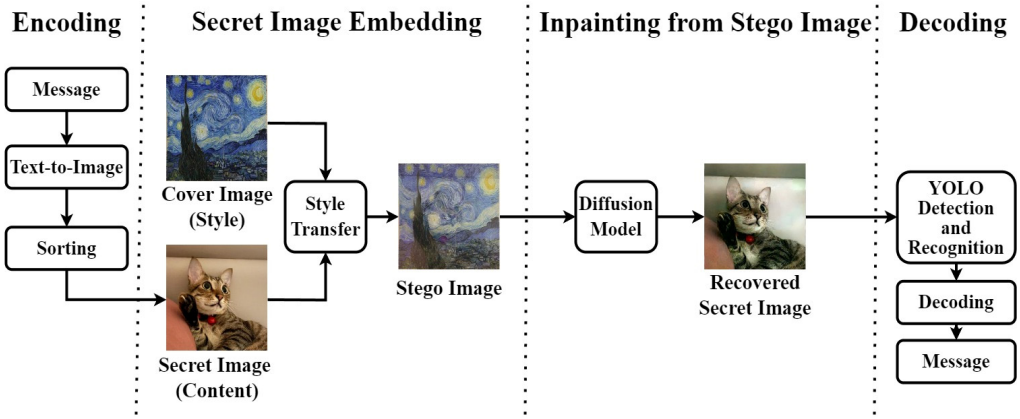


Figure 2. The framework of the proposed model.

4.1. Encoding

The proposed method uses images to encrypt and transmit messages through ASCII hexadecimal encoding. The standardized character encoding system ASCII serves to represent English letters and symbols. We defined 16 distinct object image categories that match the hexadecimal characters from 0 through F. The object categories and hexadecimal codes are linked in Figure 3.

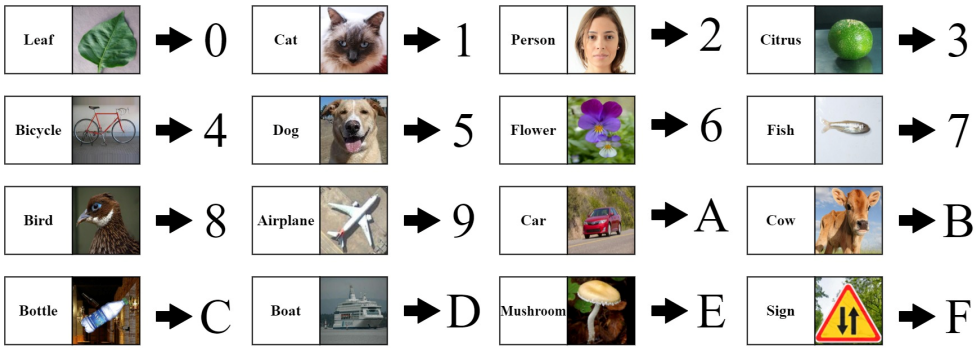


Figure 3. Object category encoding table.

Each English character is converted into its corresponding two-digit hexadecimal ASCII code, which is then represented using two object images. The predefined encoding table enables us to transform messages into sequences of object images after message reception. The object images are arranged into pairs of two images per character before being placed in a left-to-right and top-to-bottom order to form the secret image. The

conversion process is illustrated in Figure 4. The ASCII hexadecimal code 4D representing the character 'M' maps to the object pair bicycle and ship. The two object images combine into a single secret image.

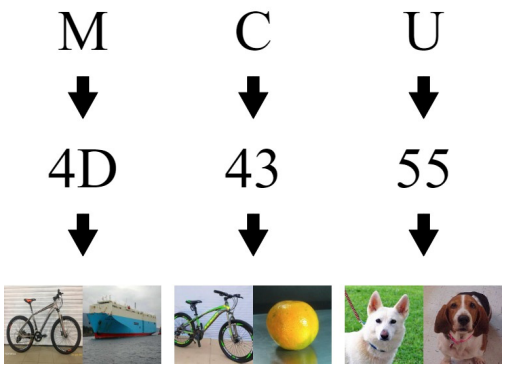


Figure 4. Message conversion process.

4.2. Stego Image Generation

The stego image generation process uses feature fusion capabilities from style transfer techniques. The encoded secret image merges with the cover image through feature-level fusion. In this process, a stego image is generated because the cover image functions as the style image while the secret image serves as the content to be embedded. This process is also adopted by original style transfer approaches [2]. The content loss function maintains the shape and structure of content images, while the style loss function based on Gram matrices preserves the texture and color patterns of style images by analyzing convolutional feature map correlations.

However, since our objective is to embed and conceal information, preserving the shape and outline of the secret image is critical. We implement content loss for both secret and cover images. As shown in Figure 5, the VGG19 network architecture performs feature fusion through the first convolutional kernel of each convolutional block. Following the neural style transfer approach by Gatys et al. [2], the content loss is defined as

$$L_{contnt} = \sum_{l} W_c^l \times \frac{1}{HWC} \sum_{i,j} \left(\hat{Y}_{i,j}^l - Y_{i,j}^l \right)^2 \tag{10}$$

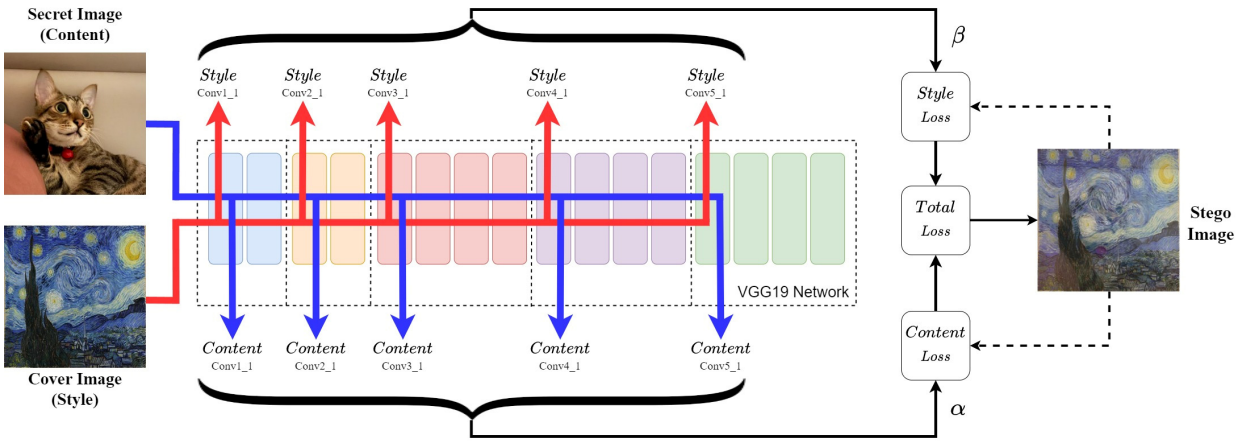


Figure 5. Style transfer network architecture.

The weight W_c^l represents the l-th layer weight and H, W, C define the height, width, and channel number of the feature map. The indices i, j point to the spatial coordinates of

the l-th layer. The fused (stego) image feature map appears as $\hat{Y}$, while the secret (content) image feature map appears as Y. The loss metric employed in this work relies on Mean Squared Error (MSE) between these two feature maps, consistent with the content loss approach in [2]. The style loss is defined as

$$L_{style} = \sum_{l} W_s^l \times \frac{1}{HWC} \sum_{i,j} \left(\hat{Y}_{i,j}^l - Z_{i,j}^l \right)^2 \tag{11}$$

The style weight W_s^l controls the l-th layer and Z shows the feature map of the cover (style) image. The MSE metric measures the difference between the output fusion and style image feature representations. The total loss is defined as

$$L_{total} = \alpha \times L_{contnt} + \beta \times L_{style} \tag{12}$$

The fusion process uses α and β as hyperparameters to determine the importance of secret image and cover image, respectively. The two weights operate as complementary values that must add up to 1 ($\alpha + \beta = 1$).

4.3. Secret Image Inpainting

For the recovery of the secret image, we adopt a conditional diffusion model. The first step requires the generation of the noise schedule. The sequence follows a total length of T steps, which represents the maximum number of diffusion steps. The β_t values are scaled between -1 and 1 to achieve training stability and faster convergence while preventing gradient explosion or vanishing. The initial value of β_1 is set to 10^{-4} and the last value of β_T is set to 0.02, according to the configuration in [3]. The network receives a 6-channel concatenated image in which the first 3 channels contain the stego image and the last 3 channels contain the noisy version of the secret image at timestep t, which combines the stego image with the T step forward diffusion version of the secret image, denoted as y_t. The target output is the predicted noise added to generate y_t, effectively inpainting the original noise from the previous step y_{t-1}, as shown in Figure 6.

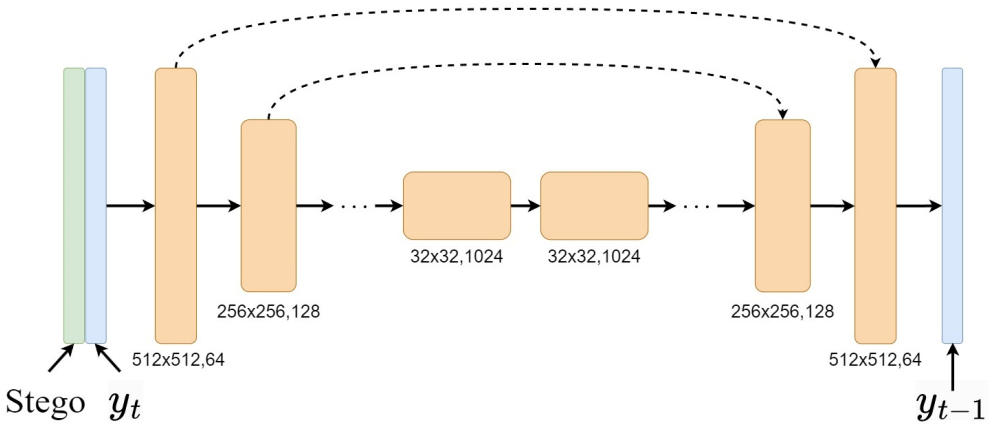


Figure 6. Conditional diffusion model architecture.

In this model, each step associated with a specific noise level is encoded as a 1D vector and passed into multiple layers of the U-Net as a conditional input. The timestep t is embedded and injected into multiple layers of the U-Net, providing temporal context for the denoising process. This allows the network to adjust predictions based on the current noise level. The U-Net processes the input through multiple resolution levels, with feature maps progressively reducing from $512 \times 512 \times 64$ to $32 \times 32 \times 1024$ in the encoder path, then expanding back to the original resolution in the decoder path. The

U-Net architecture is symmetric, with the encoder on the left extracting features through two convolutional layers with ReLU activation per block. Except for the first convolution layer (which processes the input), the second convolution layer maintains the same spatial dimensions and channel depth. A 2×2 max pooling layer follows each block, progressively down-sampling the feature map to one-quarter of the original size. The decoder, on the right side, mirrors this structure and includes 2×2 up-sampling layers in the second convolution of each block to restore the original spatial resolution. The skip connections concatenate features from corresponding encoder and decoder layers, enabling the network to recover details during the inpainting process. The features require cropping to match dimensions when necessary. Finally, a 1×1 convolution layer maps the features back to a 3-channel RGB image. Our network is optimized using the following loss function, based on L2 regularization:

$$E_{\boldsymbol{x},\boldsymbol{y}_t,\epsilon} = \left\| \mu_\theta \left(\boldsymbol{x}, \sqrt{\overline{\alpha}_t}\boldsymbol{y}_t + \epsilon\sqrt{1-\overline{\alpha}_t} \right) - \epsilon \right\|_2 \tag{13}$$

The model μ_θ, parameterized by the U-Net, takes as input a 6-channel concatenated image and predicts the noise ϵ at each timestep. The noise level is defined as $\gamma_t = (1 - \beta_t)$, and the cumulative product $\overline{\alpha}_t = \prod_{s=1}^{t} \gamma_s$ governs the scaling across time. The loss function computes the Euclidean distance between the predicted noise and the true noise ϵ, guiding the model to accurately reverse the diffusion process step by step.

4.4. Decoding

YOLOv7 is selected as the object detection framework for its outstanding accuracy and high processing speed. To prepare the dataset, we use the LabelImg annotation tool to manually label the collected object images across 16 predefined categories, corresponding to the encoding scheme used in our system. The YOLO labeling convention guides the annotation process, in which each image receives position (bounding box) and class label information, which is saved in TXT format. The annotated dataset receives division into training, validation, and test sets to improve model generalization capability.

5. Experimental Results and Analysis

This section discusses and evaluates the performance and experimental implementation of the proposed system. We begin by describing the system environment, development tools, and the data collection and training setup. Then, we present the experimental results for each stage of the pipeline, followed by a quantitative analysis.

5.1. Experimental Set-Up

This section presents the experimental setup. The goal is to hide important secret images within cover images and later recover them on the receiving end. Since the embedding process is based on a pre-trained VGG network, no additional training is required for the embedding stage. However, training is necessary for both secret image inpainting and YOLO-based object detection. All training was conducted on a system equipped with an NVIDIA RTX 3090 GPU (24 GB VRAM, 10,496 CUDA cores) and an Intel (R) Core (TM) i7-10700KF.

5.1.1. Image Quality Evaluation Metrics

In this paper, *PSNR* (Peak Signal-to-Noise Ratio), *SSIM* (Structural Similarity Index), *LPIPS* [43] (Learned Perceptual Image Patch Similarity), and *FID* [44] (Frechet Inception Distance) are used to evaluate the quality of stego images and inpainted secret images. *PSNR* functions as a standard quality assessment tool for images, measuring the extent of the distortion that occurs during image restoration and compression operations. The

calculation of the *PSNR* starts with determining the Mean Square Error (*MSE*) between original and processed images before converting the result to a logarithmic scale for evaluation purposes. The formula for *PSNR* is as follows:

$$PSNR = 10 \cdot log_{10}\left(\frac{MAX_I^2}{MSE}\right) \tag{14}$$

MAX_I denotes the maximum possible pixel value of the image, typically 255. *MSE* represents the Mean Square Error between the original image I and the comparison image K, and is calculated as follows:

$$MSE = \frac{1}{mn}\Sigma_{i=1}^{m}\Sigma_{j=1}^{n}\left(I_{(i,j)} - K_{(i,j)}\right)^2 \tag{15}$$

where m and n represent the image width and height, respectively, and $I_{(i,j)}$ and $K_{(i,j)}$ are the pixel values at position (i, j) in the original and processed images.

SSIM is also a widely used metric that evaluates perceived image quality by considering luminance, contrast, and structural information. *SSIM* better reflects human visual perception by quantifying the similarity in structure between the two images. The formula for *SSIM* is as follows:

$$SSIM(x, y) = \frac{\left(2\mu_x\mu_y + c_1\right)\left(2\sigma_{xy} + c_2\right)}{\left(\mu_x^2 + \mu_y^2 + c_1\right)\left(\sigma_x^2 + \sigma_y^2 + c_2\right)} \tag{16}$$

where μ_x and μ_y are the mean values of images x and y; σ_x^2 and σ_y^2 are the variances; σ_{xy} is the covariance between x and y; and c_1 and c_2 are constants to stabilize the division.

LPIPS is a learned perceptual metric that evaluates image similarity using features extracted from deep neural networks. The distance between these features is computed to assess the perceptual similarity between images, enabling better capture of human perceptual judgment. The formula for *LPIPS* is as follows:

$$d(x, x_0) = \sum \frac{1}{H_l W_l}\sum_{h,w}\|w_l \odot (\hat{y}_{hw}^l - \hat{y}_{0hw}^l)\|_2^2 \tag{17}$$

where $d(x, x_0)$ represents the perceptual distance between the input image x and reference image x_0; $\hat{y}$ denotes the feature maps from layer l of the pre-trained network; H_l and W_l represent the spatial dimensions of the feature maps at layer l; w_l represents the learned linear weights in the channel dimension; $\odot$ denotes element-wise multiplication in the channel direction, and the L2 distance between feature maps is computed. The final distance is calculated as the weighted average sum across all layers. Lower *LPIPS* values indicate higher perceptual similarity between the two images.

FID is a commonly used metric for evaluating the quality of generated images. A pre-trained Inception model [45] is used to extract deep features from images. Then the feature distributions of generated images are treated as two multivariate Gaussians by estimating their respective means and covariances. The formula for *FID* is as follows:

$$FID(x, g) = \left|\mu_\chi - \mu_g\right|^2 + Tr\left(\Sigma x + \Sigma g - 2\sqrt{\Sigma_x\Sigma_g}\right) \tag{18}$$

where x represents the mean of features computed from real images through the Inception network; g represents the mean of features computed from generated images through the Inception network; $\left|\mu_\chi - \mu_g\right|^2$ represents the mean values; $Tr\left(\Sigma x + \Sigma g - 2\sqrt{\Sigma_x\Sigma_g}\right)$ represents the covariance matrices; and Tr denotes the trace operation that calculates the

sum of elements along the main diagonal of a square matrix. Lower *FID* scores indicate that the feature distribution of generated images is closer to that of real images.

5.1.2. Style Transfer Hyperparameters

Different settings of the style transfer hyperparameters α and β result in stego images with varying visual characteristics. When the value of α gradually decreases and, correspondingly, β increases, the influence of the secret image diminishes, while the visual features of the cover image become more dominant, as shown in Figure 7.

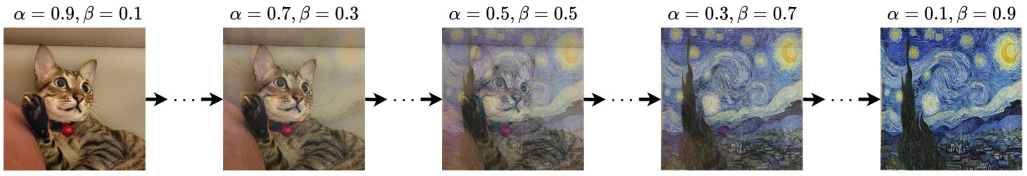


Figure 7. Variations in style transfer hyperparameters.

The continuous nature of parameters α and β enables theoretically unlimited variations in their configurations. To balance analytical comprehensiveness with experimental feasibility, we adopt a fixed step size of 0.1, selecting nine representative configurations ranging from $\alpha = 0.1$ to $\alpha = 0.9$, ensuring $\alpha + \beta = 1$ in all cases. Each configuration is evaluated through quantitative analysis, and the corresponding *PSNR*, *SSIM*, and *LPIPS* scores are summarized in Table 1. While the *PSNR* values tend to be relatively low, the *SSIM* scores and *LPIPS* remain consistently high, resulting in good-visual-quality stego images from a human perspective. After both stego image visual quality and secret image inpainting quality are evaluated, $\alpha = 0.2$ and $\beta = 0.8$ are selected as the optimal parameter combination. This configuration achieves a good balance, preserving essential information from both the secret and cover images. The adjustable parameter mechanism provides flexibility to manage security versus visual quality trade-offs.

Table 1. Results of different hyperparameter combinations.

Hyperparameters	*PSNR* (dB)	*SSIM*	*LPIPS*
$\alpha = 0.9, \alpha = 0.1$	13.8846	0.1764	0.7364
$\alpha = 0.8, \alpha = 0.2$	14.6891	0.2866	0.6994
$\alpha = 0.7, \alpha = 0.3$	15.7198	0.4219	0.6365
$\alpha = 0.6, \alpha = 0.4$	17.0660	0.5717	0.5192
$\alpha = 0.5, \alpha = 0.5$	18.7488	0.7115	0.3385
$\alpha = 0.4, \alpha = 0.6$	20.8038	0.8249	0.2171
$\alpha = 0.3, \alpha = 0.7$	25.2631	0.9035	0.1247
$\alpha = 0.2, \alpha = 0.8$	30.1567	0.9352	0.0596
$\alpha = 0.1, \alpha = 0.9$	33.0311	0.9893	0.0203

5.1.3. DDPM Hyperparameters

The DDPM implementation follows specific hyperparameter configurations. The diffusion model runs with a linear noise schedule that uses 1000 timesteps as the maximum number of diffusion steps. The β_t values receive scaling between -1 and 1 to achieve training stability and faster convergence while preventing gradient explosion or vanishing. The initial β_1 value starts at 10^{-4} and the final β_T value reaches 0.02, according to the configuration in [3].

The U-Net architecture operates with six input channels and three output channels to process images with 512 × 512 pixels. The network operates on multiple resolution levels, which decrease feature maps from 512 × 512 × 64 to 32 × 32 × 1024 in the encoder path before expanding them back to the original resolution in the decoder path. The network contains two residual blocks with ReLU activation at each resolution level and uses skip connections to maintain detail preservation between encoder and decoder layers. The training process uses a learning rate of 5×10^{-5} with zero weight decay. The model adopts Mean Squared Error (MSE) as the main loss function while using Mean Absolute Error (MAE) for evaluation metrics.

5.1.4. YOLO Hyperparameters

The YOLOv7 model of object detection is configured for 16 object categories corresponding to our semantic encoding scheme. Data augmentation techniques include HSV color space adjustments with a hue variation of 0.015, saturation of 0.7, and brightness value of 0.4. Geometric transformations include rotation up to ±45 degrees, translation up to ±0.5, and scaling up to ±0.5.

The training parameters consist of a batch size of 16, and an initial learning rate of 0.01, with a cosine annealing scheduler. The setup of the SGD optimizer contains a momentum of 0.937 and weight decay of 0.0005, trained for 300 epochs, with early stopping. Additional augmentation strategies employ a mosaic probability of 1.0, mix-up probability of 0.15, and horizontal flip probability of 0.5.

5.1.5. Training Dataset

To support the encoding process, we collected images for 16 required object categories using the Roboflow Universe dataset [42], which is specifically curated for object detection tasks using YOLO. The collected object images were used to train the detection model. For consistent training and efficient message transmission, all images were resized to a fixed resolution of 512 × 512 pixels. This standardization ensures compatibility across the entire pipeline and facilitates subsequent image processing steps.

5.2. Experimental Results

5.2.1. Encoding Results

To evaluate the system's ability to process different amounts of embedded information, this study designed four hierarchical levels of training data. The single-image level focuses on learning the features of a single object, serving as the most basic unit of information. The four-image and sixteen-image levels introduce greater scene diversity, allowing the model to extract information from increasingly complex environments. The sixty-four-image level presents a highly intricate visual context, enabling the system to handle large volumes of data. As shown in Figure 8, these four levels correspond to different types of secret images generated using the object-to-category encoding table from Figure 3. Specifically, the single-image level represents the simplest form of information; the "HI" image encodes a short textual message; the "MCU AAI" image represents a more complex combination of characters; and the image encoding "https://web2.mcu.edu.tw/" demonstrates the system's capacity for high-volume data embedding. These varying message formats are transformed into corresponding secret images and used to validate the model's ability to inpaint and recognize hidden information across different complexity levels. In cases in which object images do not completely fill the assigned grid layout, the system supplements the remaining space with randomly selected images to ensure uniform visual size and consistency.

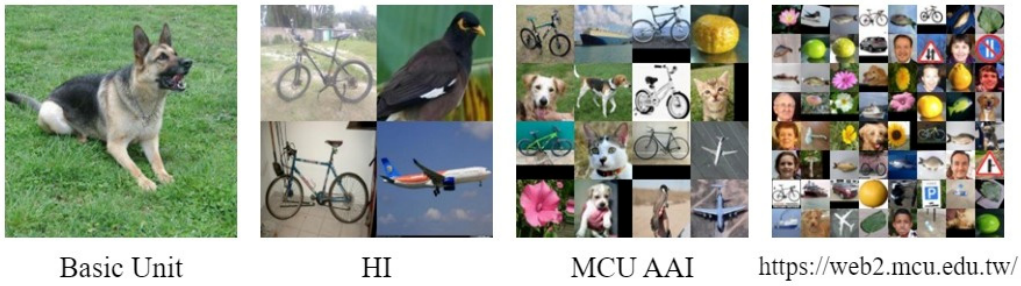


Figure 8. Confidential images of different levels.

5.2.2. Results for Steganography and Inpainting

In this paper, four representative and stylistically distinct images were selected as cover images for the style transfer process to generate stego images. The chosen covers include two well-known artworks, Van Gogh's Starry Night and Monet's Water Lilies, which demonstrate the potential of the proposed method in the domain of artistic styles. In addition, two photographic images were used: a natural scene featuring a bird and the classic test image known as Lenna, as shown in Figure 9. The experimental results clearly show that regardless of the type of cover image used, both natural visual quality and the original style can be preserved on the stego images. At the same time, the results also demonstrate that the diffusion model is highly effective in inpainting the secret image embedded within the stego image. Notably, even when the secret image consists of complex compositions such as multi-object collages, the system successfully embeds the content into various covers. This indicates the model's strong capacity for handling high-complexity inputs. These results not only confirm the applicability of style transfer techniques in the field of image steganography but also offer a novel and adaptable approach for digital content protection and privacy-preserving communication.

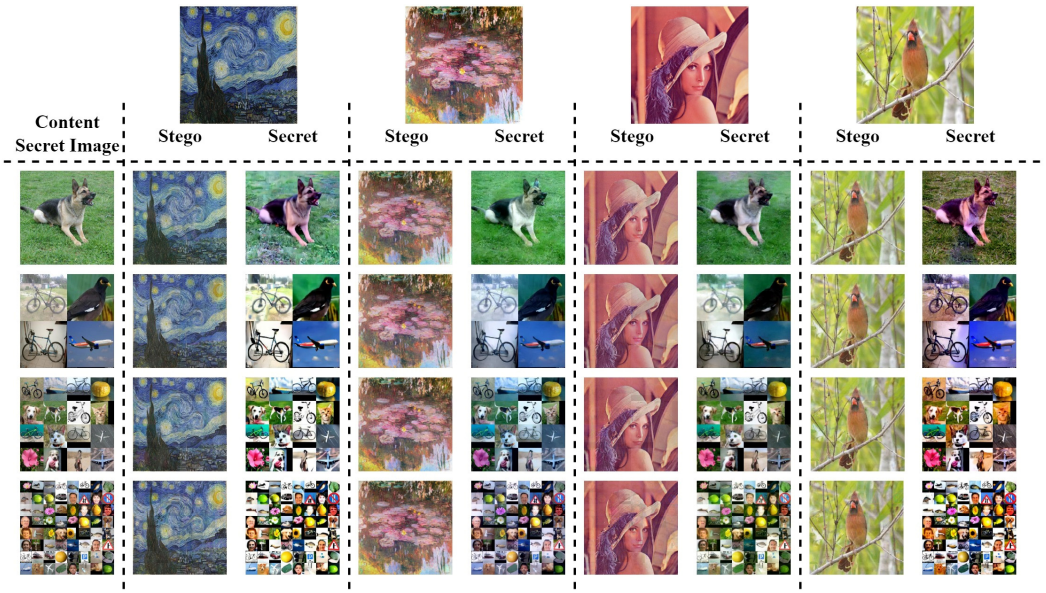


Figure 9. Results of stego image generation and secret image inpainting.

5.2.3. Object Detection and Message Decoding

Our research focused on the development and optimization of a YOLO-based object detection model capable of detecting inpainted secret images after style transfer. The model gained robustness in complex scenarios through our implementation of a hybrid training strategy. The training dataset consisted of unprocessed secret images together with various inpainted secret images that resulted from different style–content weight combinations. The

model learned to handle different levels of image distortion through this training approach. The experimental results show that the model successfully detected and identified all the embedded objects within complex composite images containing multiple objects, as shown in Figure 10.

The system starts decoding the hidden image after detection by the YOLO model. The conversion rules presented in Figure 3 enable the system to transform detected objects into their corresponding message category, which results in the inpainting of the embedded secret image message. The decoding accuracy of our system reached 100% according to the results presented in Table 2. Two evaluation metrics were used. The message accuracy shows the ratio of correctly identified valid secret objects, which measure the system's precision for extracting meaningful information. The total system accuracy demonstrates the ability to process both valid and invalid or noisy objects. The evaluation assesses the system's performance regarding valid and irrelevant object recognition, which demonstrates the reliability of our approach for real-world applications.

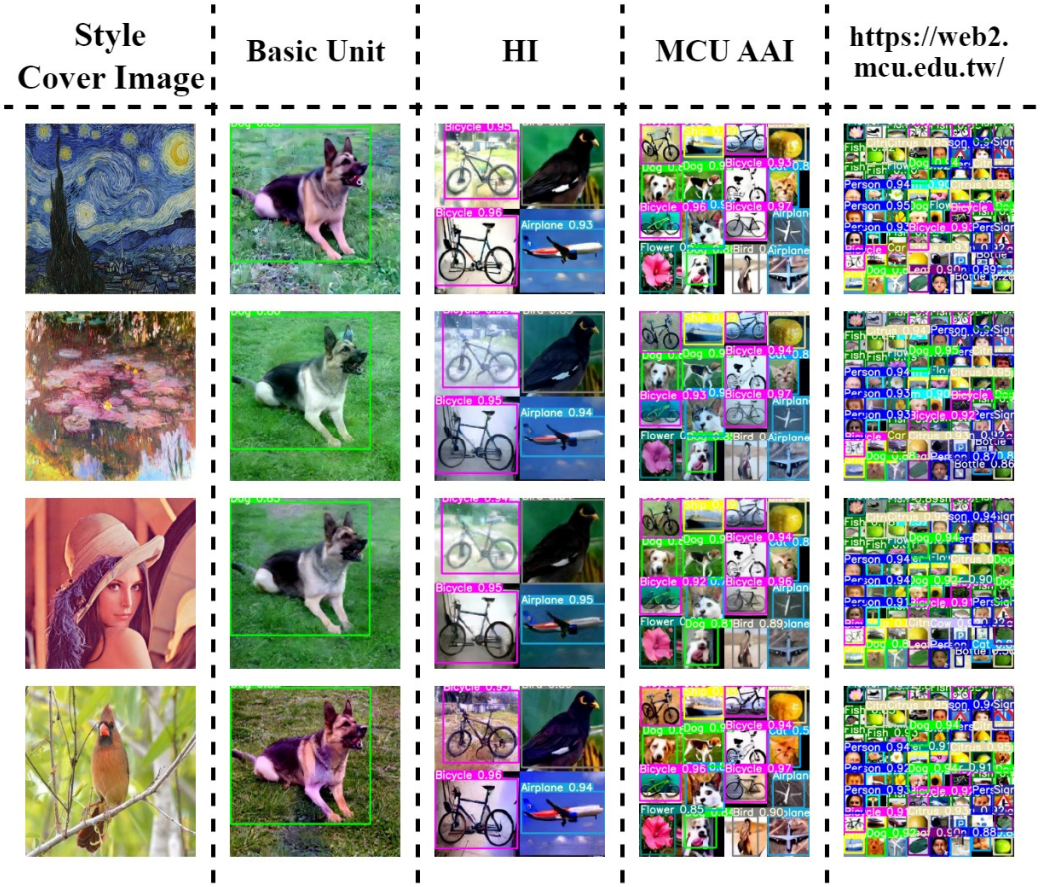


Figure 10. Results of YOLO detection.

Table 2. Results of message decoding.

Different Levels		Starry Night	Water Lilies	Lenna	Bird
Basic Unit	Message Accuracy	100%	100%	100%	100%
	Total Accuracy	100%	100%	100%	100%
HI	Message Accuracy	100%	100%	100%	100%
	Total Accuracy	100%	100%	100%	100%

Table 2. *Cont.*

Different Levels		Starry Night	Water Lilies	Lenna	Bird
MCU AAI	Message Accuracy	100%	100%	100%	100%
	Total Accuracy	100%	100%	100%	100%
https://web2.mcu.edu.tw/	Message Accuracy	100%	100%	100%	100%
	Total Accuracy	100%	100%	100%	100%

5.3. Computational Efficiency

To evaluate the real-world viability of our proposed method, the analysis of the computational efficiency is performed on the key components using the experimental setup described in Section 5.1. The DDPM inference process, which performs the image inpainting to recover the secret image from the stego image, requires an average of 10 s for each 512 × 512 image. The YOLOv7 object detection component demonstrates significantly faster performance at an average speed of 0.006 s. The pipeline, including style transfer encoding with 2.3 s, DDPM decoding with 10 s, and YOLO detection with 0.006 s, totally approximates 12.3 s per image.

5.4. Robustness Evaluation

This section reports on the experiments simulating image attacks during transmission, evaluating four common types of image transmission attacks and interference: Gaussian Noise, JPEG Compression, Resize, and Gaussian Blur. The experiments are designed with three intensity levels, Low Level, Medium Level, and High Level, to simulate varying degrees of image degradation. The specific attack parameters for each level are detailed in Table 3.

Table 3. Attack parameters across different intensity levels.

Strength	Gaussian Noise (σ)	JPEG Compression (%)	Resize	Gaussian Blur (Kernel, σ)
Low Level	10	90	0.9	3/1.0
Medium Level	20	80	0.8	5/1.5
High Level	30	70	0.7	7/2.0

The results of the image inpainting after various attacks are illustrated in Figure 11, showing the different levels of quality degradation caused by four attack types. In Gaussian noise attacks, as the intensity level increases, the image is progressively covered by extensive noise, making the inpainting image content difficult to recognize at high levels. JPEG compression attacks exhibit a different degradation pattern, in which compression artifacts become apparent as the compression ratio increases. Resize attacks primarily affect image resolution and clarity, with image details becoming progressively blurred and edges losing sharpness as the scaling intensity increases. Gaussian blur attacks create an overall blurring effect on the image to simulate considerably hazy conditions.

This research demonstrates the proposed framework's robustness through target detection tests across different attack types and levels, as presented in Table 4. The method shows outstanding resistance to JPEG compression and resizing attacks because high detection accuracy can be maintained during intense attacks. The strong noise interference from Gaussian noise and Gaussian blur attacks damages the hidden information structure, which negatively affects the feature extraction accuracy. The performance degradation

under high-level attacks is primarily caused by external noise added to the stego image and subsequently affects object detection accuracy. The proposed method shows strong robustness to image transmission issues with multiple types of attacks.

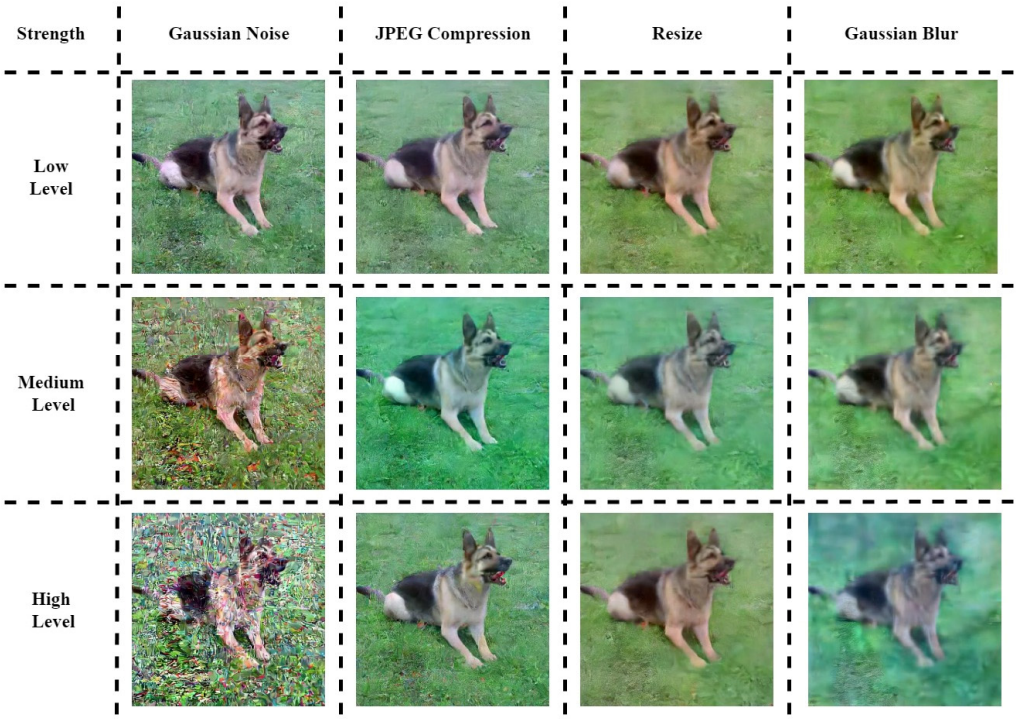


Figure 11. Image inpainting results following different attack types and intensity levels.

Table 4. Robustness evaluation of the proposed method against various image attacks.

Different Levels		Basic Unit	HI	MCU AAI	https://web2.mcu.edu.tw/
Gaussian Noise	Low Level	100%	100%	100%	100%
	Medium Level	100%	100%	75%	84.37%
	High Level	0%	0%	31.25%	48.44%
JPEG Compression	Low Level	100%	100%	100%	100%
	Medium Level	100%	100%	100%	100%
	High Level	100%	100%	100%	98.44%
Resize	Low Level	100%	100%	100%	100%
	Medium Level	100%	100%	100%	98.43%
	High Level	100%	100%	100%	96.88%
Gaussian Blur	Low Level	100%	100%	100%	100%
	Medium Level	100%	75%	81.25%	79.69%
	High Level	0%	50%	50%	42.19%

5.5. Qualitative Analysis

Multiple image restoration models perform global inpainting operations to restore the concealed secret images from stego images, as demonstrated in Figure 12. In this study, we introduce the first global image recovery approach to inpainting tasks. The De-noising Diffusion Probabilistic Model (DDPM) enables the step-by-step recovery of the original secret image from the stylized stego input. This approach is different from most

existing methods, which often rely on local inpainting or traditional generative models for inpainting.

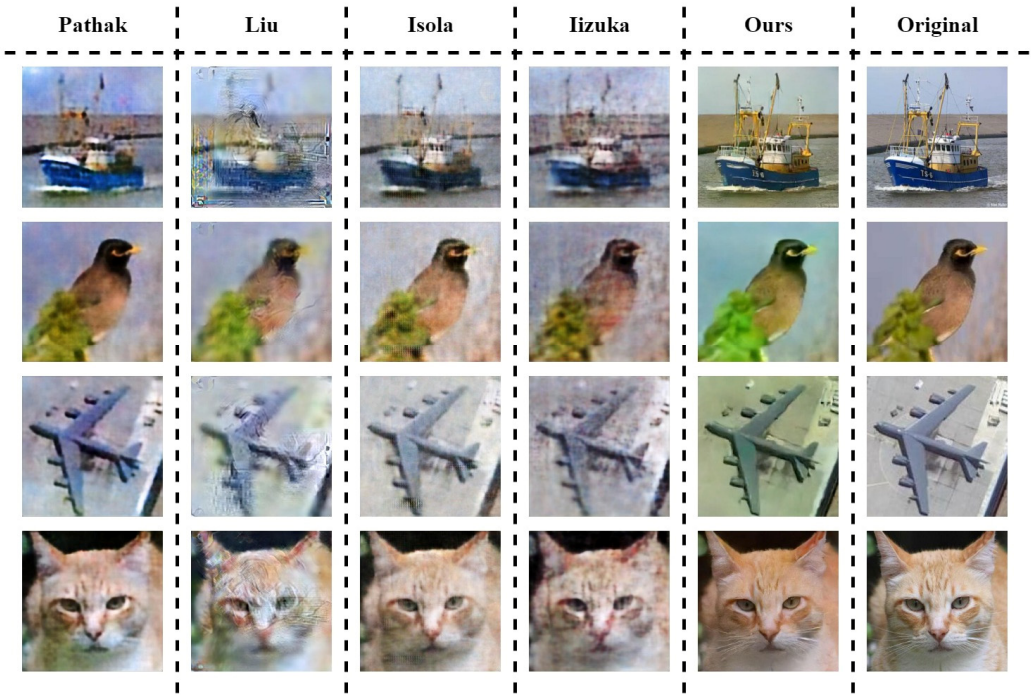


Figure 12. Comparison of secret image restoration results using different inpainting models.

The performance evaluation of multiple image inpainting models, which uses a dataset we created for this research, appears in Table 5. The results show that although there is still room for improvement in our *PSNR*, *SSIM*, *LPIPS*, and *FID* metrics, our method provides stable and acceptable performance compared to traditional models. Our method provides more than image quality enhancement. The method unites a two-layer protection system with state-of-the-art generative modeling technology. This approach offers promising directions for future development and scalability improvements.

Table 5. Comparison of *PSNR*, *SSIM*, *LPIPS*, and *FID* for different methods.

Method	*PSNR* (dB)	*SSIM*	*LPIPS*	*FID*
Pathak [26]	17.4457	0.2786	0.3988	177.3536
Liu [27]	16.5800	0.2259	0.4560	286.8824
Isola [28]	17.1155	0.3316	0.3687	197.8397
Iizuka [29]	18.6662	0.3103	0.5122	224.9508
Ours	21.1740	0.6231	0.2768	100.4428

6. Conclusions

In this paper, we propose the Steganographic Message Transmission Method Based on Style Transfer and Denoising Diffusion Probabilistic Model Technologies. The encoding process begins by converting the secret message into an image format, which is then encoded and assembled. The secret image undergoes style transfer embedding into a cover image, which produces the stego image. The DDPM model conducts global image inpainting and extracts the secret image. YOLOv7 is used for detecting and extracting embedded secret messages. This entire process forms a dual-layer security framework that ensures safe message delivery. The experimental findings demonstrate the better

performance of the proposed method compared to existing techniques in terms of PSNR, SSIM, LPIPS, and FID, but there is still room for improvement. Future research may consider using a discriminator from adversarial networks or applying a joint loss strategy. This could help synchronize the optimization of the style transfer and diffusion models, which would result in better overall performance.

The system was tested with only a limited number of images. The accuracy of the YOLO object detection possibly decreases when detecting objects with abnormal lighting conditions, partial occlusion, or a different viewing angle than the training data. This detection system faces challenges in processing images with complex backgrounds and multiple overlapping objects and struggles to accurately identify and locate hidden objects. The current results suggest that our method is effective, but further testing on various datasets and more images may be required to demonstrate the robustness of the system. In summary, this study provides a foundation for a secure message transmission framework based on style transfer and diffusion models and suggests promising directions for the future development of steganographic techniques.

Author Contributions: C.-P.H.: Conceptualization, Methodology, and Supervision. Y.-H.L.: Software and experiments, Writing—original Chinese draft. P.-S.H.: Data curation, Visualization, Writing—review and editing. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The datasets generated and/or analyzed during the current study are available from the corresponding author on reasonable request.

Acknowledgments: The authors would like to thank the editors and reviewers for their valuable work.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

References

1. Subramanian, N.; Fawad; Nasralla, M.M.; Esmail, M.A.; Mostafa, H.; Jia, M. Image steganography: A review of the recent advances. *IEEE Access* **2021**, *9*, 23409–23423. [CrossRef]
2. Gatys, L.A.; Ecker, A.S.; Bethge, M. Image style transfer using convolutional neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016.
3. Saharia, C.; Chan, W.; Saxena, S.; Li, L.; Whang, J.; Salimans, T.; Ho, J.; Fleet, D.J.; Norouzi, M. Palette: Image-to-image diffusion models. In Proceedings of the ACM SIGGRAPH Conference, Vancouver, BC, Canada, 8–11 August 2022.
4. Ho, J.; Jain, A.; Abbeel, P. Denoising diffusion probabilistic models. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 6840–6851.
5. Bender, W.; Gruhl, D.; Morimoto, N.; Lu, A. Techniques for data hiding. *IBM Syst. J.* **1996**, *35*, 313–336. [CrossRef]
6. Lin, E.T.; Delp, E.J. A review of fragile image watermarks. In Proceedings of the ACM Multimedia Security Workshop, Orlando, FL, USA, 2–3 October 1999.
7. Kumar, A.; Pooja, K. Steganography—A data hiding technique. *Int. J. Comput. Appl.* **2010**, *9*, 19–23. [CrossRef]
8. Huang, C.-P.; Hsieh, C.-H. Delivering messages over user-friendly code images grabbed by mobile devices with error correction. In Proceedings of the IEEE International Conference on Multimedia and Expo Workshops, Turin, Italy, 29 June–3 July 2015.
9. Rahim, R.; Nadeem, S. End-to-end trained CNN encoder-decoder networks for image steganography. In Proceedings of the European Conference on Computer Vision Workshops, Munich, Germany, 8–14 September 2018.
10. Baluja, S. Hiding images in plain sight: Deep steganography. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 2066–2076.
11. Zheng, S.; Gao, Z.; Wang, Y.; Fang, Y.; Huang, X. Image steganography based on style transfer. In Proceedings of the 2024 3rd International Conference on Image Processing and Media Computing (ICIPMC), Xi'an, China, 12–14 January 2024.
12. Zhang, B.; He, L.; Li, M.; Liu, W.; Li, G. The research and analysis of robust image steganography for arbitrary style transfer. In Proceedings of the 2024 5th International Conference on Artificial Intelligence and Computer Engineering (ICAICE), Beijing, China, 22–24 March 2024.
13. Goodfellow, I.J.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative adversarial nets. *Adv. Neural Inf. Process. Syst.* **2014**, *27*.
14. Volkhonskiy, D.; Nazarov, I.; Burnaev, E. Steganographic generative adversarial networks. In Proceedings of the International Conference on Machine Vision, Amsterdam, The Netherlands, 16–18 November 2019; Volume 11433.

15. Fu, Z.; Wang, F.; Cheng, X. The secure steganography for hiding images via GAN. *EURASIP J. Image Video Process.* **2020**, *2020*, 46. [CrossRef]
16. Song, Y.; Sohl-Dickstein, J.; Kingma, D.P.; Kumar, A.; Ermon, S.; Poole, B. Score-based generative modeling through stochastic differential equations. *arXiv* **2020**, arXiv:2011.13456.
17. Dhariwal, P.; Nichol, A. Diffusion models beat GANs on image synthesis. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 8780–8794.
18. Song, J.; Meng, C.; Ermon, S. Denoising diffusion implicit models. *arXiv* **2020**, arXiv:2010.02502.
19. Rombach, R.; Blattmann, A.; Lorenz, D.; Esser, P.; Ommer, B. High-resolution image synthesis with latent diffusion models. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 19–24 June 2022.
20. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.
21. Johnson, J.; Alahi, A.; Fei-Fei, L. Perceptual losses for real-time style transfer and super-resolution. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 8–16 October 2016.
22. Ubhi, J.S.; Aggarwal, A.K. Neural style transfer for image within images and conditional GANs for destylization. *J. Vis. Commun. Image Represent.* **2022**, *85*, 103483. [CrossRef]
23. Zhu, J.-Y.; Park, T.; Isola, P.; Efros, A.A. Unpaired image-to-image translation using cycle-consistent adversarial networks. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017.
24. Chen, H.; Bai, Y.; Zhang, W.; Zhao, J. Dualast: Dual style-learning networks for artistic style transfer. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 19–25 June 2021.
25. Bertalmio, M.; Sapiro, G.; Caselles, V.; Ballester, C. Image inpainting. In Proceedings of the ACM SIGGRAPH Conference, New Orleans, LA, USA, 23–28 July 2000.
26. Pathak, D.; Krahenbuhl, P.; Donahue, J.; Darrell, T.; Efros, A.A. Context encoders: Feature learning by inpainting. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016.
27. Liu, G.; Reda, F.A.; Shih, K.J.; Wang, T.C.; Tao, A.; Catanzaro, B. Image inpainting for irregular holes using partial convolutions. In Proceedings of the European Conference on Computer Vision, Munich, Germany, 8–14 September 2018.
28. Isola, P.; Zhu, J.-Y.; Zhou, T.; Efros, A.A. Image-to-image translation with conditional adversarial networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017.
29. Iizuka, S.; Simo-Serra, E.; Ishikawa, H. Globally and locally consistent image completion. *ACM Trans. Graph.* **2017**, *36*, 1–14. [CrossRef]
30. Saharia, C.; Ramesh, A.; Ge, S.; Gharbi, M.; Norouzi, M.; Fleet, D.J.; Salimans, T. Image super-resolution via iterative refinement. *IEEE Trans. Pattern Anal. Mach. Intell.* **2023**, *45*, 4713–4726. [CrossRef] [PubMed]
31. Lugmayr, A.; Danelljan, M.; Timofte, R. Repaint: Inpainting using denoising diffusion probabilistic models. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 19–24 June 2022.
32. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014.
33. Girshick, R. Fast R-CNN. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 11–18 December 2015.
34. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 1137–1149. [CrossRef] [PubMed]
35. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016.
36. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. SSD: Single shot multibox detector. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 8–16 October 2016.
37. Redmon, J.; Farhadi, A. YOLO9000: Better, faster, stronger. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017.
38. Redmon, J.; Farhadi, A. YOLOv3: An incremental improvement. *arXiv* **2018**, arXiv:1804.02767. [CrossRef]
39. Bochkovskiy, A.; Wang, C.-Y.; Liao, H.-Y.M. YOLOv4: Optimal speed and accuracy of object detection. *arXiv* **2020**, arXiv:2004.10934. [CrossRef]
40. Wang, C.-Y.; Bochkovskiy, A.; Liao, H.-Y.M. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Vancouver, BC, Canada, 18–24 June 2023.
41. Ronneberger, O.; Fischer, P.; Brox, T. U-Net: Convolutional networks for biomedical image segmentation. In Proceedings of the Medical Image Computing and Computer-Assisted Intervention, Munich, Germany, 5–9 October 2015.
42. Roboflow Universe. Available online: https://universe.roboflow.com/ (accessed on 20 January 2024).
43. Zhang, R.; Isola, P.; Efros, A.A.; Shechtman, E.; Wang, O. The unreasonable effectiveness of deep features as a perceptual metric. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018.

44. Heusel, M.; Ramsauer, H.; Unterthiner, T.; Nessler, B.; Hochreiter, S. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NeurIPS), Long Beach, CA, USA, 4–9 December 2017.
45. Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; Wojna, Z. Rethinking the Inception architecture for computer vision. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.