mathematics

MDPI

Article

Null Space Properties of Neural Networks with Applications to Image Steganography

Xiang Li [1,*] and Kevin M. Short [2,*]

[1] Department of Mathematics, The Ohio State University, Columbus, OH 43210, USA
[2] Integrated Applied Mathematics Program, Department of Mathematics and Statistics, University of New Hampshire, Durham, NH 03824, USA
* Correspondence: li.15497@osu.edu (X.L.); kevin.short@unh.edu (K.M.S.)

Abstract

This paper advances beyond adversarial neural network methods by considering whether the underlying mathematics of neural networks contains inherent properties that can be exploited to fool neural networks. In broad terms, this paper will consider a neural network to be composed of a series of linear transformations between layers of the network, interspersed with nonlinear stages that serve to compress outliers. The input layer of the network is typically extremely high-dimensional, yet the final classification is in a space of a much lower dimension. This dimensional reduction leads to the existence of a null space, and this paper will explore how that can be exploited. Specifically, this paper explores the null space properties of neural networks by extending the null space definition from linear to nonlinear maps and discussing the presence of a null space in neural networks. The null space of a neural network characterizes the component of input data that makes no contribution to the final prediction so that we can exploit it to trick the neural network. One application described here leads to a method of image steganography. Through experiments on image data sets such as MNIST, it has been shown that the null space components can be used to force the neural network to choose a selected hidden image class, even though the overall image can be made to look like a completely different image. The paper concludes with comparisons between what a human viewer would see and the part of the image that the neural network is actually using to make predictions, hence showing that what the neural network "sees" is completely different than what we would expect.

Keywords: neural networks; null space; image steganography

MSC: 68T07

check for updates

Academic Editors: Janez Žerovnik, Xiaobing Feng and Francisco Chiclana

Received: 12 August 2025
Revised: 18 October 2025
Accepted: 20 October 2025
Published: 24 October 2025

Citation: Li, X.; Short, K.M. Null Space Properties of Neural Networks with Applications to Image Steganography. *Mathematics* **2025**, *13*, 3394. https://doi.org/10.3390/math13213394

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

Despite their remarkable success in various tasks, neural networks (NNs) can sometimes behave in unexpectedly fragile ways, particularly in image recognition. One well-known problem is the existence of *adversarial examples*, where even small, carefully crafted perturbations to the input can lead to incorrect outputs [1,2]. While adversarial examples highlight an important class of vulnerabilities, they are not the only cause of unreliability. Such behaviors are not merely the result of noisy inputs or training data limitations but may reflect deeper issues rooted in the architecture of the neural networks. In this work, we take a different perspective to examine a structural property of neural networks, namely their *null space*, as a potential source of fragility.

Many existing studies attribute the vulnerability of neural networks to input-level manipulations. In 2014, Szegedy et al. [3] first discovered an intriguing weakness of deep neural networks. They showed that neural networks for image classification can be easily fooled by small perturbations. Following this observation, numerous studies have been carried out to find different ways to generate adversarial examples (see, for example, [2,4–6]). It is observed that adversarial examples have good transferability across models, which suggests that the existence of adversarial examples is also a property of data sets [7]. Thus, adversarial examples are not restricted to the given model.

Unlike the adversarial examples, our study focuses on a structural property of neural networks: the null space of neural networks. The null space is a well-established concept in linear algebra, but its connection to neural networks may not be obvious. If one were to consider a standard neural network from the perspective of a mathematician, the input can be considered a vector, generally in a high-dimensional space. Considering Figure 1, each entry x_i in the input vector $\vec{x}$ is placed in an open circle in the input layer. If we ignore for the moment the unitary node indicated by the 1 at the bottom of the first layer, the weights w_{ij} are multiplied by the entries in the input vector to produce the intermediate values y_j, where $y_j = \sum_{i=1}^{n_0} w_{ij}x_i$ at the nodes in the middle layer. Hence, the mathematical structure is basically a matrix multiplication of a weight matrix W times the input vector $\vec{x}$, so the presence of a null space associated with the weight matrix is very possible. Further, in the case of an NN, the known information is in the input vector and the weight matrix is to be determined via the training process. Note that the number of degrees of freedom in W is much greater than the dimension of the input vector; thus the training process can lead to underdetermined systems and high-dimensional null spaces.

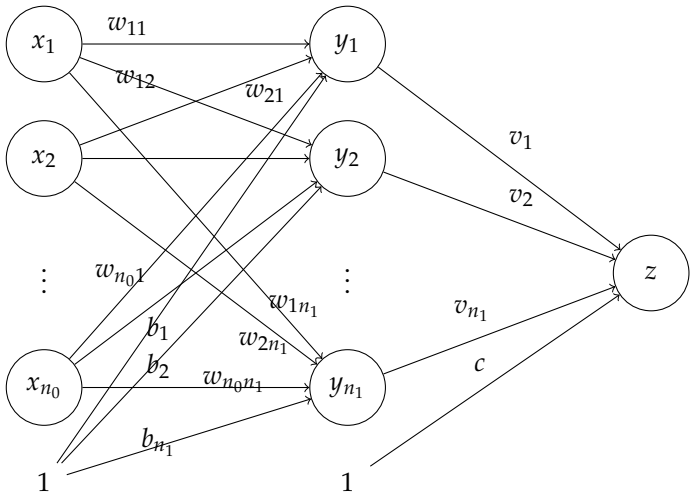


Figure 1. Basic structure of a fully connected neural network.

In the simplest examples of NNs, the transformation from one layer to the next may be a linear transformation, but a bias term, as indicated by the unitary node labeled 1 at the bottom of the input layer, is usually introduced, which then converts the transformations into affine transformations. The affine form will introduce some complications, and this paper will explore the impact of this on the null space characteristics of the linear transformation component of NNs. When all of the layers are considered, along with the activation functions, we say that we have a fully connected neural network (FCNN) with n_0 nodes in the first layer, n_1 nodes in the second layer, and so on. This FCNN is also referred to as an $(n_0, n_1, \cdots, n_k)$-FCNN. Finally, it should be mentioned that a nonlinear activation function is usually applied to the data in an intermediate layer, such as a sigmoid function or ReLU function, but these functions can be thought of as a compressing or compactifying

function that diminishes the impact of outlier results or, in the case of ReLU, negative-valued intermediate results. While these nonlinear activation functions can obfuscate the impact of the null space effects, this paper will separate out the stages to show that the null space properties, extended to consider the nonlinear effects, can be used to fool the NN and cause it to give unexpected classification results. A detailed analysis will be given in Section 2.

As a primary goal, we discuss how the null space of a neural network affects classification outcomes, revealing it to be a critical factor that may inherently limit a model's ability to make reliable predictions. The approach takes advantage of the (nonlinear) null space properties of the neural network and generates examples to fool the neural network by adding a large difference to images without changing the predictions. More importantly, to the naked eye, the large differences will look like completely different objects than those that will be recognized by the neural network, achieved via image steganography. Considered from a broader perspective, only a few studies have analyzed the null space properties of neural network functions. Cook et al. [8] integrated the null space analysis on weight matrices with the loss function to detect out-of-distribution inputs rather than analyzing the reliability of NN predictions. Rezaei et al. [9] used the null space of the last layer weight matrix to quantify overfitting. Beyond these, some studies have investigated the null space structure of the parameter space. Sonoda et al. [10] showed that the parameter spaces of overparameterized NNs inherently contain null components that do not affect training loss but may impact generalization. In addition, a few studies have worked on the null space associated with the input data itself [11–13].

However, there is still a lack of understanding regarding the global effects of null space properties for NNs. In this study, inspired by the null space of linear maps, we introduce the concept of a null space for nonlinear maps and discuss how the null space applies to fully connected neural networks (FCNNs). We show how the null space is generally an intrinsic property of NNs. Further, in most cases, once the NN's architecture is determined, the dimension of the null space of the NN is also determined. To illustrate this in concrete terms, we use null space-based methods to fool an image recognition NN, demonstrating an application of image steganography.

Image steganography is a technique to hide an image inside another image [14,15]. Traditional image steganography methods can be categorized into two main types: methods operating in the spatial domain [16] and the transform domain [17]. Among methods operating in the transform domain, discrete orthogonal moment transforms have been widely used in image analysis and image steganography [18–22]. In addition to traditional steganography methods, approaches based on NNs have also been employed for image steganography, using several different approaches [23–31]. In this paper, inspired by the null space analysis of NNs, we propose a null space-based method for image steganography. Our approach shares some similarity with discrete orthogonal moment-based steganography, as both methods decompose images with respect to a chosen basis. However, while moment methods rely on polynomial bases (e.g., Tchebichef, Krawtchouk), our method employs basis vectors from the null space of a neural network. In addition, although conceptually related to other steganographic methods, the null space-based method presented in this paper provides new degrees of freedom and a clear process for creating steganographic images for NNs. The main goal is to hide an image and recognize the correct class of the hidden image while presenting to the (human) viewer an image that is completely different to the hidden image that will be recognized by the NN.

In summary, the main contributions of this paper are as follows:

- We introduce a novel perspective on NN reliability by analyzing the role of the null space structure for the NN and showing that its dimension is architecture-dependent.

- We propose a null space-based steganographic method that generates visually different images while hiding images that are recognized with high certainty by the model.

The rest of this paper is organized as follows: Section 2 introduces the (nonlinear) null space of NNs. Inspired by null space analysis, an image steganography method based on the null space of FCNNs is introduced in Section 3. Section 4 presents a series of experiments on different image data sets. In Section 5, we offer some discussion and conclusions about the advantages and limitations of the null space method and the differences between the null space image steganography method and adversarial examples. Additionally, more examples are presented to emphasize that what the NN is seeing and using to make classification decisions is not what we humans might perceive it to be seeing.

2. Methodology and Theory

In this section, we introduce the null space of nonlinear maps and discuss the null space of neural networks (NNs). In order to keep the flow of ideas consistent, detailed proofs will be placed in the appendix.

2.1. Null Space of Linear and Nonlinear Maps

In linear algebra, the null space of an $m \times n$ matrix A is $\text{Null}(A) = \{\vec{x} \in \mathbb{R}^n : A\vec{x} = \vec{0}\}$ and $\text{Null}(A)$ is a subspace of $\mathbb{R}^n$. Consider a linear map $T : \mathbb{R}^n \to \mathbb{R}^m, T(\vec{x}) = A\vec{x}$ with $\text{Null}(A)$ nontrivial. By the definition of a null space, it is not difficult to see that for any $\vec{x} \in \mathbb{R}^n$ and $\vec{x}_{null} \in \text{Null}(A)$, $T(\vec{x} + \vec{x}_{null}) = T\vec{x}$. That is, adding any vector in $\text{Null}(A)$ to any input $\vec{x}$ will not change the output since the transformation is a linear map.

In most cases, null space in nonlinear maps is not defined as described above. The set $\{\vec{x} : f(\vec{x}) = 0\}$ for a nonlinear map $f : \mathbb{R}^n \to \mathbb{R}^m$ is not a subspace of $\mathbb{R}^n$. Moreover, the property of linearity $f(\vec{x} + \vec{y}) = f(\vec{x}) + f(\vec{y})$ does not hold for the nonlinear map f. Although we can still find the set $\{\vec{x}_\alpha : f(\vec{x}_\alpha) = 0\}$, the property $f(\vec{x} + \vec{x}_\alpha) = f(\vec{x})$ is no longer true for all $\vec{x} \in \mathbb{R}^n$. To preserve this useful property, the concept of null space can be adapted to nonlinear maps by identifying a set of vectors $\vec{x}_{null}$ that satisfy the condition $f(\vec{x} + a\vec{x}_{null}) = f(\vec{x})$ for any $\vec{x} \in \mathbb{R}^n, a \in \mathbb{R}$:

Definition 1 (Null space of a nonlinear map)**.** *The null space of a nonlinear map $f : \mathbb{R}^n \to \mathbb{R}^m$, denoted as $N(f)$, is the set of all vectors in which, by adding these vectors to any input, the image under the map, or output, will not change:*

$$N(f) := \{\vec{v} \in \mathbb{R}^n : f(\vec{x}) = f(\vec{x} + a\vec{v}) \text{ for all } \vec{x} \in \mathbb{R}^n, a \in \mathbb{R}\}. \tag{1}$$

When this definition is applied to linear maps, it is equivalent to the null space concept in linear algebra. However, for the purposes of this paper, we will use the term "null space" in a broader sense, extending its application to include affine maps, nonlinear maps, etc. Similarly to the null space in linear algebra, $N(f)$ is a subspace of $\mathbb{R}^n$ (Proposition A1). Furthermore, as demonstrated in Proposition A2, all the vectors in $N(f)$ are mapped to a single point by f. More specifically, for any $\vec{x}, \vec{y} \in N(f)$, $f(\vec{x}) = f(\vec{y})$. Further properties about null spaces of nonlinear maps are provided in Appendix B. One difference between null spaces of linear and nonlinear maps is that a nonlinear map will not always map null space vectors to zeros.

The definition of the null space for nonlinear maps is straightforward. However, in practice, it is difficult to find a null space explicitly by following this definition. Next, we will introduce an alternative yet equivalent definition for the null space of nonlinear maps, which provides a more accessible way with which to find the null space of a nonlinear map.

To begin with, given a nonlinear map $f : \mathbb{R}^n \to \mathbb{R}^m$, let us consider the subspaces of $N(f)$. Suppose f has a decomposition; then, f can be expressed as the composition of

two functions $f = f_2 \circ f_1$ and f_1 is linear. It follows that $\text{Null}(f_1)$ is a subspace of $N(f)$. This is straightforward to see, since for any $\vec{x}_{null} \in \text{Null}(f_1)$, $f_1(\vec{x}) = f_1(\vec{x} + \vec{x}_{null})$ holds for every $\vec{x} \in \mathbb{R}^n$, and so $f(\vec{x} + \vec{x}_{null}) = f_2 \circ f_1(\vec{x} + \vec{x}_{null}) = f_2 \circ f_1(\vec{x}) = f(\vec{x})$. Therefore, we now have a method to find a subspace of $N(f)$. Hence, we define a partial null space as follows:

Definition 2 (Partial null space of a map)**.** *Given a nonlinear map, $f : \mathbb{R}^n \to \mathbb{R}^m$. If f has a decomposition $f = f_2 \circ f_1$, where $f_1 : \mathbb{R}^n \to \mathbb{R}^d$ is a linear map, then the null space of f_1 is defined as a partial null space of f (given by f_1). We denote it as $PN_{f_1}(f)$.*

Then, the question of how to find the null space with partial null spaces and the decomposition of maps remains. In the following lemmas, we present some results regarding partial null spaces and the null space of a nonlinear map. Detailed proofs are provided in Appendix B.

Lemma 1. *Let $f : \mathbb{R}^n \to \mathbb{R}^m$ be a nonlinear map. Every partial null space $PN(f)$ is a subspace of $N(f)$, $\dim PN(f) \leq \dim N(f)$.*

Lemma 2. *For every nonlinear map $f : \mathbb{R}^n \to \mathbb{R}^m$, there exists a decomposition $f = f_2 \circ f_1$, where f_1 is a linear map such that $PN_{f_1}(f) = Null(f_1) = N(f)$, i.e., the partial null space given by f_1 is equal to $N(f)$.*

Note that $PN_{f_1}(f) = \text{Null}(f_1) = N(f)$ is not always true for any arbitrary decomposition $f = f_2 \circ f_1$. In the following corollary, we further discuss the conditions under which this equation holds.

Corollary 1. *Let $f : \mathbb{R}^n \to \mathbb{R}^m$ be a nonlinear map. The null space of f is the largest partial null space of f. That is, $N(f) = PN_{f_1}(f)$ if $f = f_2 \circ f_1$, f_1 is a linear map, and $PN_{f_1}(f)$ has the largest dimension among all decompositions of f.*

Directly from Lemmas 1 and 2, Corollary 1 gives an equivalent definition of $N(f)$ in terms of its partial null space. Consequently, instead of searching for all possible vectors that satisfy Equation (1), we have turned the null space problem into a problem of finding a decomposition of f that yields the largest partial null space.

2.2. Null Space of Fully Connected Neural Networks

Consider a fully connected neural network $f : \mathbb{R}^{n_0} \to \mathbb{R}^{n_{K+1}}$, where $K \in \mathbb{N}$ denotes the number of hidden layers of f, with $n_1, n_2, \ldots, n_K \in \mathbb{N}$ representing widths of the hidden layers, i.e., the number of nodes in each hidden layer. $n_0, n_{K+1} \in \mathbb{N}$ are the input and output dimensions, respectively. f is called a $(K+1)-$layer fully connected neural network (FCNN). In this paper, we refer to this FCNN architecture as a $(n_0, n_1, \ldots, n_K)-$FCNN.

As shown in Figure 2, assume the input values are $\{x_i\}$ and the activation functions are denoted by σ; then the output of the k^{th} neuron in the j^{th} layer is given by $\sigma(z_k^{[j]}) = \sum_i W_j(k,i)x_i + \vec{b}_j(k)$, and the outputs of the j^{th} layer can be written in a compact matrix form $\sigma(\vec{z}^{[j]}) = W_j\vec{x} + \vec{b}_j$. The function represented by this neural network is

$$f(\vec{x}) = W_{K+1}\,\sigma(W_K\,\sigma(\cdots W_2\,\sigma(W_1\vec{x} + \vec{b}_1) + \vec{b}_2 \cdots) + \vec{b}_K) + \vec{b}_{K+1}.$$

Define a sequence of linear and affine transformations as $T_i(\vec{x}) = W_i\vec{x}$, $A_i(\vec{x}) = \vec{x} + \vec{b}_i$ for $i = 1, \ldots, k+1$, respectively. The function of FCNN f can also be represented by a composition of maps:

$$f = A_{k+1} \circ T_{k+1} \circ \sigma \circ A_k \circ T_k \circ \cdots \circ A_2 \circ T_2 \circ \sigma \circ A_1 \circ T_1. \tag{2}$$

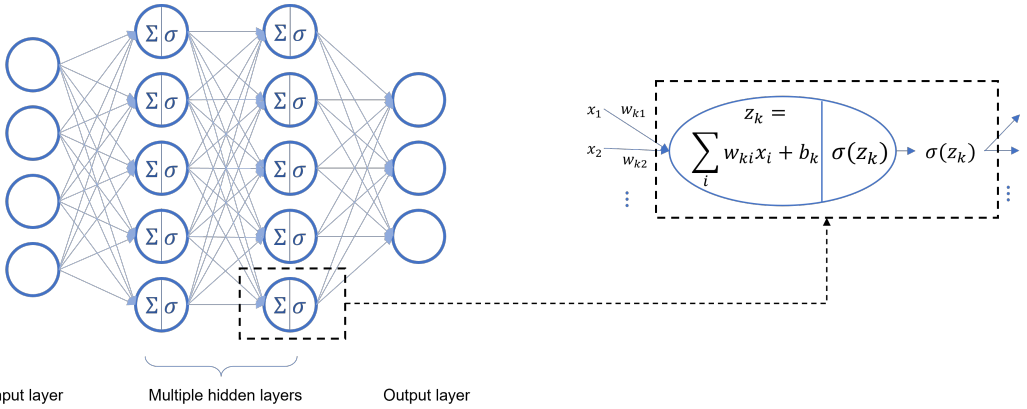


Figure 2. Architecture of a fully connected neural network.

Common activation functions include sigmoid activation function and Rectified Linear Unit (ReLU). We refer to the $(n_0, n_1, \ldots, n_K)$-FCNN with ReLU activation as a $(n_0, n_1, \ldots, n_K)$-ReLU NN.

According to Corollary 1, we can conduct null space analysis on FCNNs using the composite function form (Equation (2)). An FCNN can be naturally decomposed into two parts, $(A_{K+1} \circ T_{K+1} \circ \sigma \circ T_K \circ \cdots \circ A_2 \circ T_2 \circ \sigma \circ A_1)$ and T_1, where T_1 is linear. Thus, $\text{Null}(T_1)$ is a partial null space of FCNN f which, in most cases, is also the null space $N(f)$.

As will be shown in Section 3, input images can be decomposed into null space and non-null space components, which can be used to fool the neural networks in an image steganography application.

2.3. Null Space of Convolutional Neural Networks

Similarly, a convolutional neural network (CNN) can also be represented by the composition of maps. Instead of having only affine transformations and activation functions, the early layers of a CNN allow two additional types of computation: convolution, denoted as C, and pooling, denoted as S. To determine the null space of a CNN f, we can first decompose f into the composition of maps $f = T_{k+1} \circ \sigma \circ \cdots \circ \sigma \circ C$. So, in most cases, the first convolution operation C is the maximum linear map of f, and $N(f) = \text{Null}(C)$.

The null space of a CNN is significantly more complex than an FCNN. In this paper, we will only give an overview of the null space analysis for CNN in a simple case. Consider the case when the input image of a CNN only has one channel, for example the grayscale number images in the MNIST data set (a color image typically has three channels).

A typical convolutional layer in a CNN comprises multiple kernels. To find the null space of the entire convolutional layer $\text{Null}(C)$, we can initially determine the null space of a single kernel $\text{Null}(K)$ corresponding to one convolution operation. As shown in Appendix C, given one kernel, if the convolution operation keeps the output image with the same dimensions as the input image (same padding), then $N(f) = \{0\}$ in most cases. If the output image of the first convolution operation has fewer dimensions than the input image (valid padding), $N(f)$ can be nontrivial. According to Lemma A1, in most cases, the null space of a given kernel has a dimension that is equal to the difference between the dimensions of the input and output image. For example, given a 28×28

image and a 3×3 kernel, the dimension of the null space of this convolution operation is $28 \times 28 - 26 \times 26 = 108$ in most cases.

Assume that the first convolutional layer C has n kernels $K_1, \cdots, K_n$; then $N(f) = N(C) = \bigcap_{i=1}^{n} \text{Null}(K_i)$. For instance, if the input is a 28×28 image and the first convolutional layer has six 3×3 kernels, then the null space of the CNN is the intersection of six $109-$dimensional subspaces of $\mathbb{R}^{784}$. The likelihood of there being a non-trivial intersection between 109-dimensional subspaces of a 784-dimensional space is low, but care must be taken when designing the kernels.

Compared to FCNNs, CNNs can achieve a trivial null space with fewer unknown parameters in the first weight matrix and perhaps greater robustness.

3. Photo Steganography Based on the Null Space of a Neural Network

In considering the task of hiding secret information inside another image, a natural idea emerges from our previous discussions on the null space properties of FCNNs. To this end, in this section we propose a method for creating *steganographic images*, referred to as "stego images", to fool FCNNs and pass on secret classification information.

In terms of a pre-trained FCNN f, for any input image X, the FCNN f gives a decomposition,

$$X = \hat{X} + X_{\perp},$$

where $\hat{X} \in N(f)$ is the projection of X onto the null space of f and $X_{\perp}$ is the orthogonal complement that determines the prediction. Since $f(\hat{X}) = 0$, the output depends solely on $X_{\perp}$; that is, $f(X) = f(X_{\perp})$. This decomposition provides the idea for controlling the predicted label/classification independently of the image's appearance.

Consider two images H and C: H is the *hidden image* which carries the secret label. C is the *cover image* which determines the visual appearance. Our goal is to pass a new image S that visually looks like C but secretly hides the classification information of H, i.e., which is classified the same as the hidden image by f. This can be achieved by decomposing both H and C with respect to the null space $N(f)$: the null space projections, denoted by $\hat{H}$ and $\hat{C}$, which lie in the null space and do not affect the prediction; and the orthogonal complements, denoted by $H_{\perp}$ and $C_{\perp}$, which contribute to the output of f. Importantly, this method does not involve retraining the network; instead, the null space $N(f)$ is computed directly from the first layer of a pre-trained neural network f. By combining the null space projection of C, denoted by $\hat{C}$, with the prediction-relevant component of H, denoted by $H_{\perp}$, we construct

$$S = \alpha_1 H_{\perp} + \alpha_2 \hat{C}, \quad \alpha_1, \alpha_2 \in (0, 1),$$

so that S appears similar to C but satisfies $f(S) = f(H)$ if the f is trained on grayscale images to correctly classify the scaled inputs, such as αH for $\alpha \in (0, 1)$. The pseudocode of the proposed image steganography method is provided in Appendix A.

Figure 3 provides a visualization of the whole process. We consider a data set $\{(X_i, y_i)\}_{i=1}^{N}$, where X_i are 28×28 grayscale images normalized to the range $[-1, 1]$ and y_i are class labels. First, a fully connected ReLU network f is trained on both the original images X_i and their rescaled versions αX_i, where $\alpha \in (0, 1)$, to create "headroom" for linear combinations. Then, the null space $N(f)$ is obtained by computing the singular value decomposition (SVD) of the first hidden layer weight matrix $W_1 = USV^T$ and extracting the columns of V associated with zero singular values, which form an orthonormal basis for the null space $N(f)$. These columns form the matrix V_2. If H and C are column vectors in the form of neural network inputs, we compute $H_{\perp}$ and $\hat{C}$ as follows:

$$H_{\perp} = H - V_2 V_2^T H, \qquad \hat{C} = V_2 V_2^T C,$$

where $V_2^T H$ and $V_2^T C$ give the coefficients (dot products) of H and C along the null space basis vectors, respectively. Multiplying by V_2 reconstructs the projection within the null space. Thus, $\hat{C}$ is composed only of null space vectors, while $H_\perp$ has all null space components removed. Finally, we construct the stego image $S = \alpha_1 H_\perp + \alpha_2 \hat{C}$. Note that the resulting stego image S must remain within the normalized range $[-1, 1]$, consistent with the range of the neural network inputs. This condition constrains the choice of α_1 and α_2: α_1 cannot be too large (otherwise some entries of S may fall outside $[-1, 1]$) or too small (otherwise the hidden component becomes too weak to carry sufficient information from the hidden image). Also, guaranteeing the visual quality of the cover image requires α_2 to be as large as possible.

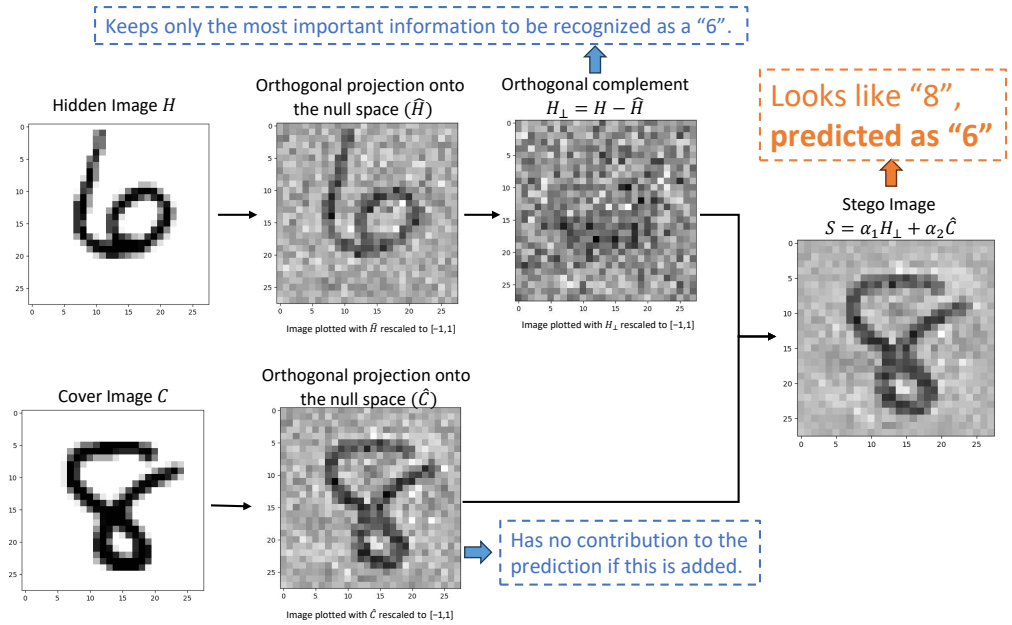


Figure 3. An example of creating a steganographic image with the null space of an ReLU NN. For a given neural network, if we combine the null space component of the cover image with the orthogonal complement component of the hidden image, we can obtain a steganographic image that visually resembles the cover image but is classified the same as the hidden image. In this example, the steganographic image appears as the number 8, similarly to the cover image, but it is classified as a 6 because of the hidden image. Note that $\hat{H}, H_\perp, \hat{C}$ are not rescaled in the algorithm. The rescaling to [−1,1] shown here is for visualization purposes only, as intermediate values may fall outside of this range and cannot be directly rendered.

In this example, the network classifies S as the digit eight based on the hidden image H while ignoring the visual features of the cover image C. So, for the example in Figure 3, the stego image looks like an "8", but when presented to the NN, it is classified as a "6".

In the next section, we will show experiments using the MNIST, Fashion-MNIST (FMNIST), and Extended MNIST (EMNIST) data sets. For simplification, we present experiment results using $\alpha_1 = 0.2$. Other values of α_1 may also be used as long as the stego image remains within the range $[-1, 1]$. Further discussion on the choice of parameters α_1 and α_2 can be found in Appendix D.

4. Experimental Results

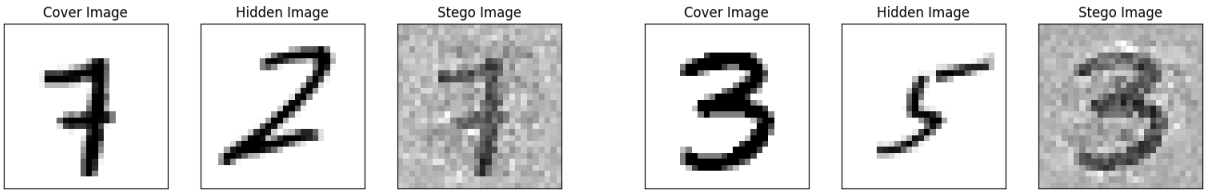
All experiments presented in this section were implemented in TensorFlow. The data sets used in these experiments include MNIST [32], Fashion-MNIST [33], Extended MNIST (EMNIST) [34], and CIFAR-10 [35].

4.1. Hide the Digits

In this section, we use the MNIST data set to conduct experiments in image steganography. For these experiments, we trained a $(784, 32, 16, 10) - \text{ReLU}$ NN, i.e., the ReLU NN has 28×28 images as the input vector $28^2 = 784$ in length, 32 nodes in the first hidden layer, 16 nodes in the second hidden layer, and 10 nodes in the output layer. The training was based on an expanded image training data set that included the original data set and the data set rescaled with $\alpha_1 = 0.2$, with a prediction accuracy of 99.79% on the original training data. For the original training data correctly predicted, the average confidence level is 99.98%. When evaluated on the rescaled training data, this ReLU NN maintained a high prediction accuracy of 99.78% with an average confidence of 99.77% on correctly predicted data. Expanding the data set to include the rescaled training data slightly reduced the training accuracy; the accuracy remains high and still provides a large pool of correctly classified images for constructing steganographic images. Notably, this ReLU NN has a 752-dimensional null space. This large null space plays a crucial role in the steganographic capabilities of the network.

We then filter out only the correctly predicted images to create a new data set. This data set comprises a total of 50,000 images, with each class having 5000 examples. The $(784, 32, 16, 10) - \text{ReLU}$ NN can correctly predict both original and rescaled data from this new data set. The main purpose of this process is to ensure that for any chosen hidden image H from the new data set, feeding $0.2H$ into the ReLU NN does not create scaling issues and yields a correct classification. Next, we will show some representative results with stego images produced by $S = 0.2H_\perp + \alpha_2\hat{C}$, where α_2 can be chosen in the range of $(0, 1)$ and is generally set to around 0.5.

Figures 4 and 5 present stego images generated using the images from the new data set. For each set of three images, the first is the cover image, the second is the hidden image, and the third, the stego image, combines parts of the cover and the hidden image. Additionally, Figure 4 annotates the combination weights α_1, α_2 of stego images and the predictions and their corresponding confidence levels. In each case, the ReLU NN predicts the hidden digit with high confidence.



(**a**) (**b**)

Figure 4. Examples of steganographic images with the MNIST data set. (**a**) The cover image is predicted ro be 7 with a confidence of nearly 100%. The hidden image is predicted to be 2 with a confidence of nearly 100%. The stego image, $S = 0.2H_\perp + 0.51\hat{C}$, is predicted to be 2 with a confidence level of 99.9%. (**b**) The cover image is predicted to be 3 with a confidence of nearly 100%. The hidden image is predicted to be 5 with a confidence of nearly 100%. The stego image, $S = 0.2H_\perp + 0.49\hat{C}$, is predicted to be 5 with a confidence level of 99.9%.

As shown in the figures, we can barely see the hidden digits in the stego images, which have patterns extremely similar to the cover images, even if the cover image is from another data set. However, these stego images are not predicted to be in the same classes as the cover images; rather, they are identified as the hidden images with high confidence. The cover image's portion within the stego image falls entirely in the null space of the ReLU NN, giving the stego image a visible pattern that resembles the cover image (plus some noise-like structure) but makes no contribution to the prediction with the ReLU NN. On

the other hand, the part of the hidden image included in the stego image appears as noise (as shown in Figure 3), but it is this portion that carries the most important information for prediction with the ReLU NN.

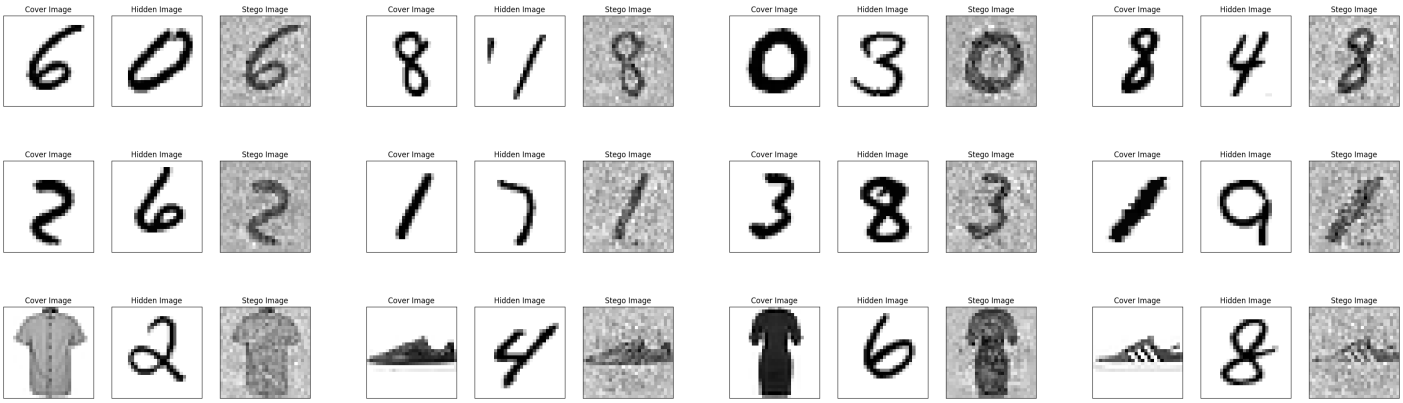
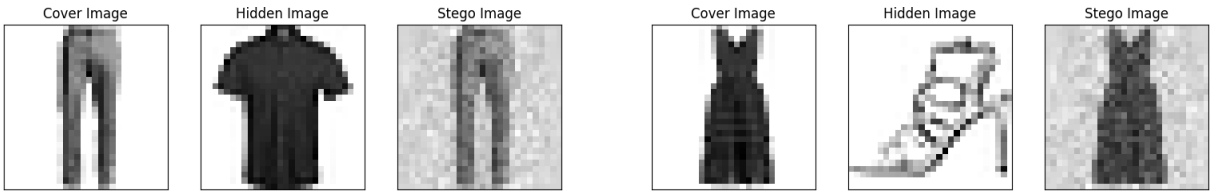


Figure 5. More examples on the MNIST data set. For each set of three images, the stego image is the combination of the null space part of the cover image and the orthogonal complement part of the hidden image. Thus, the stego image is visually similar to the cover image but is classified based on the hidden image.

4.2. Results on Other Data Sets

This proposed image steganography method is general and can be applied to any image data set. In addition to the MNIST data set, we also applied our method to other image data sets: FMNIST, EMNIST, and CIFAR-10.

For the FMNIST data set, we trained a $(784, 32, 16, 10)$-ReLU NN with a 752-dimensional null space. To generate stego images, we first created a new data set comprising both the correctly predicted original and rescaled images (also scaled by a factor of 0.2). This new data set has an average confidence of 99.74% for the original images and 97.22% for the rescaled images. Examples of stego images created from the new data set are shown in Figures 6 and 7.



(**a**) (**b**)

Figure 6. Examples of steganographic images with the FMNIST data set. (**a**) The cover image is predicted to be "Trouser" with a confidence of nearly 100%. The hidden image is predicted to be "Shirt" with a confidence of nearly 100%. The stego image, $S = 0.2H_{\perp} + 0.68\hat{C}$, is predicted to be "Shirt" with a confidence level of 86.4%. (**b**) The cover image is predicted to be "Dress" with a confidence of nearly 100%. The hidden image is predicted to be "Sandal" with a confidence of nearly 100%. The stego image, $S = 0.2H_{\perp} + 0.49\hat{C}$, is predicted to be "Sandal" with confidence of nearly 100%.

Similarly, as expected, the stego images have the look of cover images (as seen in the first column) but are recognized as the same categories as their corresponding hidden images (shown in the second column) with high confidence.

To continue this investigation, we also perform experiments on the EMNIST Balanced data set (Figure 8) and CIFAR-10 data set (Figure 9), where the same technique works. The figures suggest that the null space-based image steganography method is

also applicable to more complicated and colorful images. To guarantee the capability of concealing an image within any chosen cover image, we can train a neural network model with a sufficiently large null space. As observed in the experiments, when simple digit or letter images are used as cover images, their null space components $\hat{C}$ tend to preserve most of their visual information under projection, while more complex and colorful images, such as the images in the CIFAR-10 data set, may appear more "noisy" after being projected onto the null space, therefore requiring a larger null space to maintain visual fidelity.

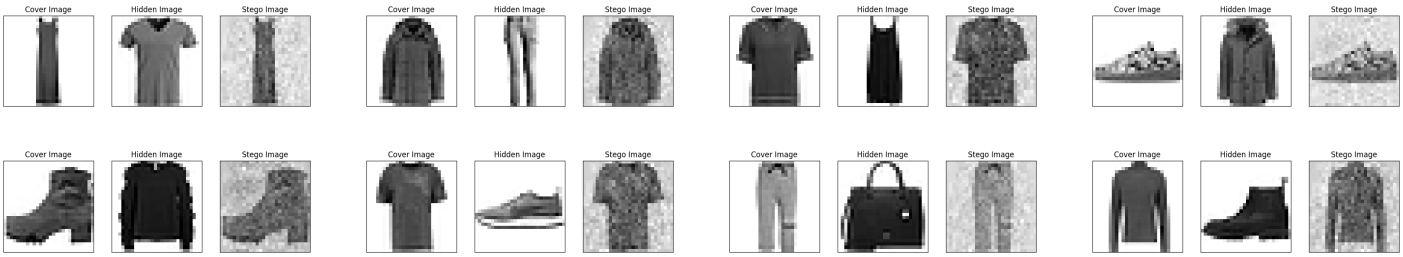


Figure 7. More examples on the FMNIST data set.

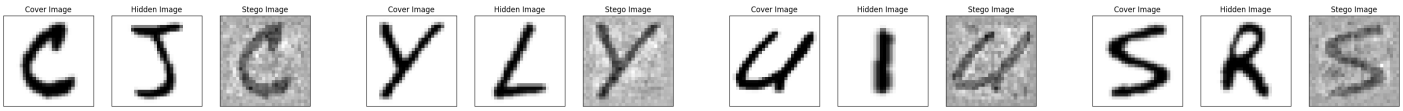


Figure 8. More examples on the EMNIST data set.



Figure 9. More examples on the CIFAR-10 data set.

4.3. Analysis of Reduced Confidence in Stego Images

While most stego images constructed by our method are classified the same as the hidden image with nearly 100% confidence, in a few cases, a reduced confidence is observed. In this regard, it is worthwhile to look at these outliers.

One such outlier result is shown in Figure 6a. While the original hidden image H is predicted to be a "shirt" with confidence close to 1, the stego image $S = 0.2H_{\perp} + 0.68\hat{C}$ is predicted to be a "shirt" with a lower confidence of 86.4%. The reduced confidence level is caused by the rescaled hidden image. According to the null space method, the prediction and confidence level for the stego image S should align with those for $0.2H_{\perp}$ and, consequently, $0.2H$. Consistent with this analysis, the rescaled hidden image $0.2H$ is also predicted to be "shirt" with 86.4% confidence. Therefore, the confidence drop reflects the effect of rescaling rather than a limitation of the null space method.

Unlike the experiment with the MNIST data set, it is more common to see lower confidence stego image examples in the experiment with the FMNIST data set. In Figure 10, we compare the original hidden image H and rescaled hidden image $0.2H$ in Figure 6a. For the rescaled image $0.2H$, the NN gave a reduced confidence of 86.4% for the correct prediction of "shirt", but it also predicted the related category of "T-shirt/Top" with a confidence of 13.4%. As shown in Figure 10, the rescaling operation reduces contrast and leads to the loss of some visual details, which causes lower confidence. For instance, the buttons and collar cannot be clearly observed in the rescaled image, so it is also more likely to be identified as a "T-shirt/Top".

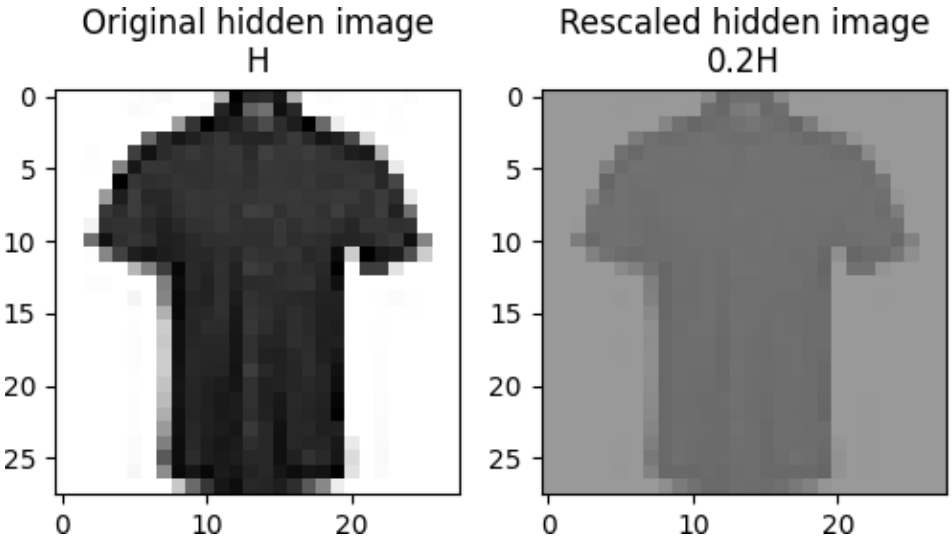


Figure 10. Original and rescaled images.

Therefore, the achievement of high confidence in steganographic images depends on having "good" hidden images with high confidence on both original and rescaled data. Essentially, the prediction confidence level of the hidden images will be no better than the prediction confidence of the original images after rescaling has been applied.

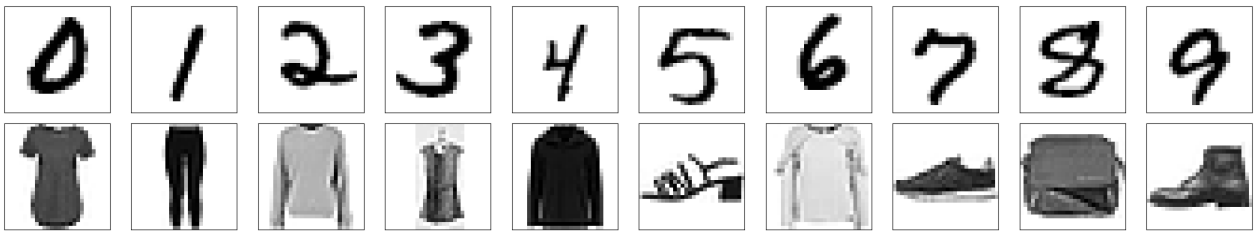
5. Discussions and Conclusions

This study discusses the fundamental role of null space in NNs, the reliability issues it may introduce, and how it can be exploited. The null space is a structural property inherent to the architecture of the NN itself, which is independent of the input data. By extending the null space concept from linear transformations to complex NN architectures, we demonstrate that this hidden structure can lead to serious reliability concerns. More specifically, if a neural network has a high-dimensional null space, it remains vulnerable, no matter how well it is trained, because one can always construct inputs to fool the model. In this regard, CNNs tend to be more robust than FCNNs from the null space perspective, as their architectures typically induce a much smaller or even trivial null space. However, the null space property can also be used for beneficial applications, including but not limited to image steganography.

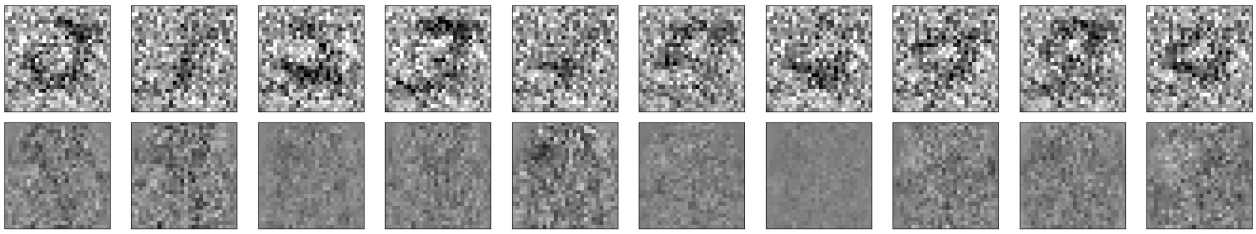
These findings have implications for model interpretability and reliability. The following discussion elaborates on what the NN actually "sees" and broader concerns like reliability in AI systems.

5.1. What We See and What NNs See

From the analysis and experimental results presented above, it is evident that NNs do not perceive visual information in the same way as humans. Figure 11 compares what we see and what the NN "sees". After removing null space components, the remaining crucial parts for prediction have fewer visual patterns than their initial appearance. Without this, distinct neural networks would see the same image differently, even when they have null spaces of the same dimensions, as illustrated in Figure 12, where the null space orthogonal components of the same images are shown as they are seen by different NNs.



(**a**)



(**b**)

Figure 11. Comparison of what is visualized by humans and NNs. (**a**) What we see. (**b**) What NN sees.

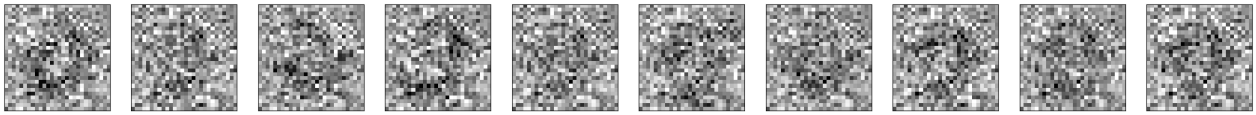


Figure 12. What another NN sees given the same set of images of digits in the top row of Figure 11a.

5.2. Null Space of NNs and Reliability Issue

Ball [36] pointed out that there are reproducibility and reliability issues with AI. Previously, studies have found that neural network models could pick up on irrelevant features to succeed in classification tasks, raising questions about their reliability. In this paper, the null space analysis on neural networks provides additional insights into these reliability concerns and shows that the NN may pick up on hidden features that are not just irrelevant but that are crafted to confuse the NN. In addition to being too focused on aligning with the particular patterns in the training data, the design of neural network architectures may also raise risks that cannot be solved by merely increasing the data set size. Importantly, it is the architecture that leads to the null space weaknesses shown in this paper.

In addition, stego images created by null space methods cannot be used to improve training. As described in [1], the adversarial example/image is a modified version of a clean image that is intentionally perturbed to mislead machine learning models, such as deep neural networks. Some studies have shown that it may be possible to harden NNs against adversarial attacks. Further, it was observed by Szegedy et al. that the robustness of deep neural networks against adversarial examples could be improved by adversarial training, where the idea is to include adversarial examples in the training data [3]. It is crucial to note that while adversarial examples using previous techniques may improve the training, they cannot solve the problem caused by the null space of a neural network. There are some interesting applications in adversarial attacks [4,37] that might be combined with null space ideas to produce an efficient, reliable way to fool neural networks, and this could be a fruitful direction for future research.

The null space analysis of NNs is not limited to the NNs in image classification tasks. In this study, image steganography is selected as an application to better visualize the impact of the null space. The existence of the null space, inherent in the neural network's

architecture, implies that one can always use the null space vectors to fool a neural network or the user of a neural network, at least when the input data projected onto the null space is close enough to the original to fool a human viewer. Recently, the development of large language models has advanced rapidly. However, these models sometimes produce incorrect or unreasonable outputs. Many researchers have focused on detecting and quantifying the extent of hallucinations in such models [38–40]. Through the analysis of the null space of neural networks, the null space may potentially be a contributing cause for these hallucinations. This also suggests a potential future direction for exploring the underlying reasons for hallucinations in large language models.

Author Contributions: Conceptualization, K.M.S.; Methodology, X.L.; Software, X.L.; Validation, K.M.S.; Formal analysis, X.L.; Data curation, X.L.; Writing—original draft, X.L.; Writing—review & editing, K.M.S.; Visualization, X.L.; Supervision, K.M.S.; Project administration, K.M.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The code and trained models supporting this study are available from the corresponding author upon request.

Conflicts of Interest: The authors declare that they have no conflicts of interest related to this work.

Appendix A. Pseudocode of the Image Stegonography Method

In this section, the pseudocode of the image stegonography method proposed in Section 3 is provided as follows:

Algorithm A1 Photo Steganography Method Based on the Null Space of a Neural Network

Input: Pre-trained fully connected neural network f with first-layer weight matrix W, hidden image H, cover image C, scaling parameter $\alpha_1 \in (0,1)$
Output: Stego image S

1. Normalize H and C to $[-1,1]$.
2. Compute the Singular Value Decomposition of W_1: $W_1 = U\Sigma V^T$.
3. Extract null space basis V_2 from V (columns with zero singular values).
4. Project the cover image C onto the null space: $\hat{C} = V_2 V_2^T C$.
5. Compute orthogonal complement of the hidden image H: $H_\perp = H - V_2 V_2^T H$.
6. Fix $\alpha_1 = 0.2$, choose the largest α_2 such that $S = \alpha_1 H_\perp + \alpha_2 \hat{C}$ remains in $[-1,1]$.

return S.

Appendix B. Proofs and Additional Results from Section 2.1

Appendix B.1. Properties of the Null Space in Nonlinear Maps

Proposition A1. *Let $f : \mathbb{R}^n \to \mathbb{R}^m$ be a nonlinear map. Then the set $N(f) := \{\vec{v} \in \mathbb{R}^n : f(\vec{x}) = f(\vec{x} + a\vec{v})$ for all $\vec{x} \in \mathbb{R}^n$ and $a \in \mathbb{R}\}$ is a subspace of $\mathbb{R}^n$.*

Proof. First, by definition, $\vec{0} \in N(f)$, and for every $\vec{v} \in N(f)$ and for every $c \in \mathbb{R}$, $c\vec{v} \in N(f)$. Second, when $\vec{v}_1, \vec{v}_2 \in N(f)$, for every $\vec{x} \in \mathbb{R}^n$ and $a \in \mathbb{R}$, we have $f(\vec{x} + a(\vec{v}_1 + \vec{v}_2)) = f(\vec{x} + a\vec{v}_1 + a\vec{v}_2) = f(\vec{x} + a\vec{v}_1) = f(\vec{x})$; hence $\vec{v}_1 + \vec{v}_2 \in N(f)$. Therefore $N(f)$ is a subspace of $\mathbb{R}^n$. □

Proposition A2. *Given a nonlinear map, $f : \mathbb{R}^n \to \mathbb{R}^m$. For any $\vec{x} \in N(f)$ and $\vec{y} \in N(f)$, $f(\vec{x}) = f(\vec{y}) = f(\vec{0})$.*

Proof. For any $\vec{x}, \vec{y} \in N(f)$, $f(\vec{x}) = f(\vec{x} + \vec{0}) = f(\vec{0}) = f(\vec{y} + \vec{0}) = f(\vec{y})$. □

Given a nonlinear map $f : \mathbb{R}^n \to \mathbb{R}^m$, as well as its null space $N(f)$, we also considered its null set, $\mathcal{N}(f) := \{\vec{v} \in \mathbb{R}^n : f(\vec{x}) = f(\vec{x} + \vec{v}) \text{ for all } \vec{x} \in \mathbb{R}^n\}$. It is easy to see that $N(f)$ is a subset of $\mathcal{N}(f)$, and $\mathcal{N}(f)$ may not be a vector space in general. However, $\mathcal{N}(f)$ also has some structure; any vector in $\mathcal{N}(f)$ that is not in the subspace $N(f)$ is part of a set of integer periodic null elements.

Proposition A3. *Given a nonlinear map $f : \mathbb{R}^n \to \mathbb{R}^m$, consider the set $\mathcal{N} = \{\vec{v} \in \mathbb{R}^n : f(\vec{x}) = f(\vec{x} + \vec{v})$ for all $\vec{x} \in \mathbb{R}^n\}$. If $\vec{v} \in \mathcal{N}$ but $\vec{v} \notin N(f)$, then for any integer k, $k\vec{v} \in \mathcal{N}$.*

Proof. Let $\vec{v_1}, \vec{v_2} \in \mathcal{N}(f)$. We prove that $-\vec{v_1}$ and $\vec{v_1} + \vec{v_2} \in \mathcal{N}(f)$. Indeed, $\vec{v_1} \in \mathcal{N}(f)$ implies that $f(\vec{x} - \vec{v_1}) = f(\vec{x} - \vec{v_1} + \vec{v_1}) = f(\vec{x})$ for every $\vec{x} \in \mathbb{R}^n$; hence $-\vec{v_1} \in \mathcal{N}(f)$. And $\vec{v_1}, \vec{v_2} \in \mathcal{N}(f)$ implies that $f(\vec{x}) = f(\vec{x} + \vec{v_1}) = f(\vec{x} + \vec{v_1} + \vec{v_2})$ for every $\vec{x} \in \mathbb{R}^n$; hence $\vec{v_1} + \vec{v_2} \in \mathcal{N}(f)$. Therefore, if $\vec{v} \in \mathcal{N}(f)$, $k\vec{v} \in \mathcal{N}$ for any integer k. □

Appendix B.2. Proofs of Lemmas from Section 2.1

Proof of Lemma 1. Assume the nonlinear map $f : \mathbb{R}^n \to \mathbb{R}^m$ has a decomposition $f = f_2 \circ f_1$ where f_1 is linear. For any $\vec{x}_{null} \in \text{Null}(f_1)$, $f_1(\vec{x}) = f_1(\vec{x} + \vec{x}_{null})$ holds for every $\vec{x} \in \mathbb{R}^n$, $f(\vec{x} + \vec{x}_{null}) = f_2 \circ f_1(\vec{x} + \vec{x}_{null}) = f_2 \circ f_1(\vec{x}) = f(\vec{x})$. Therefore, $\vec{x}_{null} \in N(f)$ and $PN_{f_1} = \text{Null}(f_1)$ is a subspace of $N(f)$ with $\dim PN(f) \leq \dim N(f)$. □

Proof of Lemma 2. Given the nonlinear map $f : \mathbb{R}^n \to \mathbb{R}^m$, consider the quotient space $\mathbb{R}^n / N(f) := \{[\vec{v}] : \vec{v} \in \mathbb{R}^n\}$, where $[\vec{v}] = \{\vec{v} + \vec{y} : \vec{y} \in N(f)\}$. The addition $[\vec{v}] + [\vec{u}] = [\vec{v} + \vec{u}]$, scalar multiplication $\lambda[\vec{v}] = [\lambda\vec{v}]$, and $[\vec{0}] = N(f) \in \mathbb{R}^n / N(f)$ make $\mathbb{R}^n / N(f)$ a vector space (of dimension $n - \dim(N(f))$). Denote f_1 as the quotient map from $\mathbb{R}^n$ to $\mathbb{R}^n / N(f)$ (i.e., $f_1(\vec{v}) = [\vec{v}]$). Then, f_1 is linear and $\text{Null}(f_1) = N(f)$. Define $f_2 : \mathbb{R}^n / N(f) \to \mathbb{R}^m$ as the map $f_2([\vec{v}]) = f(\vec{v})$. To show that f_2 is well-defined, notice that when $[\vec{v}] = [\vec{u}]$, $\vec{u} = \vec{v} + \vec{y}$ for some $\vec{y} \in N(f)$; hence $f(\vec{u}) = f(\vec{v} + \vec{y}) = f(\vec{v})$ by the definition of $N(f)$. Now, $f = f_2 \circ f_1$ is the decomposition needed. □

Appendix C. Related Results in Section 2.3

The convolution operation is linear. The convolution of an image and a kernel can also be written in a matrix-vector multiplication form. For example, consider the case of a $5 \times 5 \times 1$ image convolved with a $3 \times 3 \times 1$ kernel. Figure A1 and Equation (A1) show the convolution with valid padding and its matrix form . A $5 \times 5 \times 1$ image convolved with a $3 \times 3 \times 1$ kernel with valid padding can be written as a 9×25 matrix (kernel) multiplied with a 25×1 vector (image).

The convolution with the same padding keeps the input dimension and the output dimension the same by appending zero values in the outer frame of the images. Equation (A2) shows the matrix form of a convolution operation with the same padding; that is, a 25×25 matrix (kernel) multiplied with a 25×1 vector.

We can prove that for almost all cases, the convolution operations with valid padding or the same padding are full rank, i.e., the dimension of the null space is exactly the difference of the dimensions of the input and output. More precisely, consider all the convolution operations convolving $n_1 \times n_2$ inputs with $k_1 \times k_2$ kernels. The set of all $k_1 \times k_2$ kernels can be identified with $\mathbb{R}^{k_1 \times k_2}$, and we may equip the Lebesgue measure on the set. The following Lemma shows that, for all but a zero-measure subset of all kernels, the convolution operation has full rank.

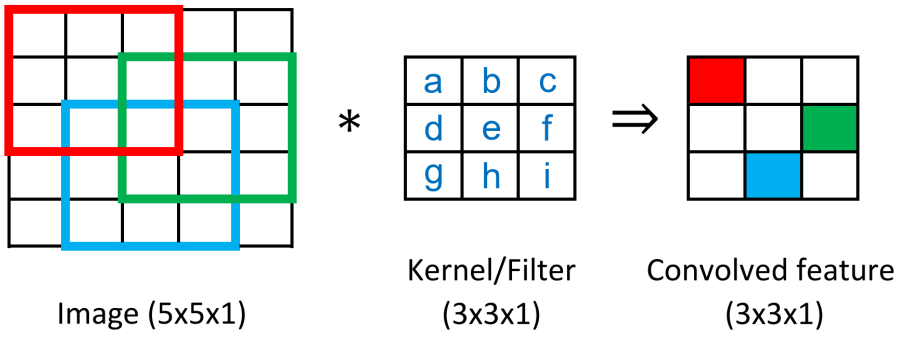


Figure A1. Convolution example.

Lemma A1. *Let n_1, n_2, k_1, k_2 be positive integers and $n_1 \geq k_1$, $n_2 \geq k_2$. For almost all $k_1 \times k_2$ kernels, the (valid padding or same padding) convolution operation of the kernel acting on $n_1 \times n_2$ images has full rank.*

Proof. Let M be the matrix representation of the convolution operation (see Equations (A1) and (A2) for both valid padding and same-padding cases).

To begin with, let us focus on the convolution with valid padding (Equation (A1)). The null space of a convolution operation is Null(M). By switching columns in the matrix M, we may assume that there is a weight a of the kernel that appears and only appears on every entry of the diagonal of M. As an example, for the matrix in Equation (A1), we switch the columns with the new order $(1, 2, 3, 6, 7, 8, 11, 12, 13, 4, 5, 9, 10, 14, 15, \cdots)$. Then, the diagonal has entries all equal to a. Let M_1 be the matrix consisting of the first $\min(k_1, k_2)$ columns and the first $\min(k_1, k_2)$ rows of M, which is a square matrix with diagonal entries all equal to a. In the following, we show that M_1 has full rank for all but a finite number of choices of a; hence M also has full rank for almost all kernels. We have that M_1 is a square matrix and $M_1 = aI + B$, where I is the identity matrix and B is a square matrix independent of a with diagonal entries all equal to zero. Therefore, M_1 is invertible (i.e., has full rank) if and only if $-a$ is not an eigenvalue of B. Fix the other values in the kernel and only change the value of a; then the matrix B is fixed. While B has many but finite eigenvalues, for every B, for all but a finite number of choices of a, M_1 has full rank. Thus, for all but a zero-measure subset of all $k_1 \times k_2$ kernels, M_1 and M have full rank.

For the convolution with the same padding, let M be the matrix representation of a given kernel (see Equation (A2)). The diagonal entries are all equal. Following the proof above, we know that M is full rank for almost all kernels. □

$$\left[\begin{array}{ccccc|ccccc|ccccc|ccccc|ccccc}
a & b & c & 0 & 0 & d & e & f & 0 & 0 & g & h & i & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & a & b & c & 0 & 0 & d & e & f & 0 & 0 & g & h & i & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & a & b & c & 0 & 0 & d & e & f & 0 & 0 & g & h & i & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & 0 & 0 & a & b & c & 0 & 0 & d & e & f & 0 & 0 & g & h & i & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & a & b & c & 0 & 0 & d & e & f & 0 & 0 & g & h & i & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & a & b & c & 0 & 0 & d & e & f & 0 & 0 & g & h & i & 0 & 0 & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a & b & c & 0 & 0 & d & e & f & 0 & 0 & g & h & i & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a & b & c & 0 & 0 & d & e & f & 0 & 0 & g & h & i & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a & b & c & 0 & 0 & d & e & f & 0 & 0 & g & h & i
\end{array}\right] \tag{A1}$$

$$\left[\begin{array}{ccccc|ccccc|ccccc|ccccc|ccccc}
e & f & 0 & 0 & 0 & h & i & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
d & e & f & 0 & 0 & g & h & i & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & d & e & f & 0 & 0 & g & h & i & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & d & e & f & 0 & 0 & g & h & i & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & d & e & 0 & 0 & 0 & g & h & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
b & c & 0 & 0 & 0 & e & f & 0 & 0 & 0 & h & i & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
a & b & c & 0 & 0 & d & e & f & 0 & 0 & g & h & i & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & a & b & c & 0 & 0 & d & e & f & 0 & 0 & g & h & i & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & a & b & c & 0 & 0 & d & e & f & 0 & 0 & g & h & i & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & a & b & 0 & 0 & 0 & d & e & 0 & 0 & 0 & g & h & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & 0 & 0 & b & c & 0 & 0 & 0 & e & f & 0 & 0 & 0 & h & i & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & a & b & c & 0 & 0 & d & e & f & 0 & 0 & g & h & i & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & a & b & c & 0 & 0 & d & e & f & 0 & 0 & g & h & i & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & a & b & c & 0 & 0 & d & e & f & 0 & 0 & g & h & i & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a & b & 0 & 0 & 0 & d & e & 0 & 0 & 0 & g & h & 0 & 0 & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & b & c & 0 & 0 & 0 & e & f & 0 & 0 & 0 & h & i & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a & b & c & 0 & 0 & d & e & f & 0 & 0 & g & h & i & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a & b & c & 0 & 0 & d & e & f & 0 & 0 & g & h & i & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a & b & c & 0 & 0 & d & e & f & 0 & 0 & g & h & i \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a & b & 0 & 0 & 0 & d & e & 0 & 0 & 0 & g & h \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & b & c & 0 & 0 & 0 & e & f & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a & b & c & 0 & 0 & d & e & f & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a & b & c & 0 & 0 & d & e & f & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a & b & c & 0 & 0 & d & e & f \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a & b & 0 & 0 & 0 & d & e
\end{array}\right] \tag{A2}$$

Appendix D. Discussion on Parameters in the Null Space-Based Image Steganography Method

This appendix provides theoretical analyses and empirical evidence for the feasible ranges of the parameters α_1, α_2 used in the construction of the steganographic image

$$S = \alpha_1 H_\perp + \alpha_2 \hat{C},$$

where $H_\perp$ denotes the orthogonal complements of the hidden image to the model's null space and $\hat{C}$ denotes the null space projections of the cover image. The design of these parameters is guided by three conditions:

1. Every entry of S lies within the normalized range $[-1, 1]$.
2. S is classified as the hidden image H by the trained model.
3. The visual appearance of S is similar to the cover image C and does not reveal the true digit of H.

Appendix D.1. Condition (1): Range Constraint and Parameter Bounds

Although both the hidden image H and the cover image C are normalized within the range $[-1, 1]$, their components after the null space decomposition,

$$H = H_\perp + \hat{H}, \quad C = C_\perp + \hat{C},$$

do not necessarily remain within this range. More specifically, some entries of $H_\perp$ and $\hat{C}$ may exceed $[-1, 1]$ even if the original inputs were normalized.

To ensure that every entry of the stego image $S = \alpha_1 H_\perp + \alpha_2 \hat{C}$ lies within $[-1, 1]$, one can fix one parameter and obtain the feasible range of the other parameter accordingly. Without loss of generality, if we first fix the value of α_1, the following must hold to satisfy the range constraint of S:

$$|\alpha_1 H_\perp(i,j)| < 1 \quad \text{for all } i,j, \tag{A3}$$

where $H_\perp(i,j)$ is the (i,j) entry of $H_\perp$.

Equation (A3) is a sufficient condition to satisfy the range constraint of S. When this condition is satisfied, it is always possible to choose at least one feasible value of α_2 such that each pixel of S is in the range $[-1, 1]$. However, if any entry of $\alpha_1 H_\perp$ already exceeds $[-1, 1]$, there may not exist a value of α_2 that can satisfy the range constraint on S. By symmetry, the same applies if α_2 is fixed first.

The specific numerical bound for α_1 (or α_2) may vary across data sets. As an illustrative example, we computed the bound of α_1 for all training samples in the MNIST data set:

$$|\alpha_1| < \frac{1}{\max_{i,j} |H_\perp(i,j)|}.$$

Across the 60,000 training images, the upper bound of $|\alpha_1|$ has a mean value of 0.7077, with a minimum of 0.4012 and a maximum of 1.0129. Thus, the representative choice $\alpha_1 = 0.2$ used in our experiments with the MNIST data set lies in the safe and numerically valid region for all samples.

Appendix D.2. Condition (2): Classification Consistency

The second condition guarantees the classification consistency of the stego image. More specifically, given a trained neural network f, the stego image should satisfy

$$f(S) = f(\alpha_1 H_\perp) = f(\alpha_1 H).$$

However, when α_1 becomes too small, the hidden image component $\alpha_1 H$ may be misclassified and fail to reveal the hidden information. Thus, a sufficiently large α_1 is required to ensure a sufficiently large pool of candidate hidden images.

As an illustrative example, we evaluated one trained MNIST network by scaling the training inputs with different values of $\alpha_1 \in [0.1, 1.0]$. The accuracies are shown in Figure A2. In this example, the accuracy remains above 0.95 for $\alpha_1 \geq 0.3$, but it drops sharply when $\alpha_1 < 0.2$. This is consistent with the example provided in Section 4.3, where a very small value for α_1 can cause the images to lose contrast, and this loss of contrast contributes to classification with diminished confidence or potential misclassifications. For more complex data sets, a larger α_1 may be required.

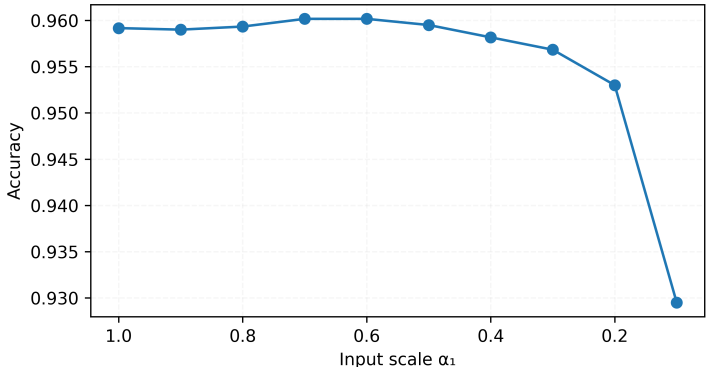


Figure A2. Comparison of accuracy with different input scaling (α_1).

Appendix D.3. Condition (3): Visual Requirements

The third condition considers the visual requirements of the stego image, which also provides an idea to fool the neural network. While the stego image $S = \alpha_1 H_\perp + \alpha_2 \hat{C}$ must be correctly recognized by the neural network as the hidden image H, its visual appearance should be similar to the cover image C to the human eye.

The visual similarity between the projected cover component $\hat{C}$ and the original cover image C is determined by the dimension of the network's null space. A larger null space dimension makes $\hat{C}$ more visually similar to C. The scaling factor α_2 controls the contrast level of $\hat{C}$. A larger α_2 makes $\hat{C}$ appear darker and more dominant in the stego image.

For example, Figure A3 shows the stego images with different scaling factors α_1, α_2. All stego images are correctly classified by the model as the hidden digit "5" with 100% confidence. However, as α_1 increases, α_2 decreases, and the projected cover image gradually becomes noisier and less visible.

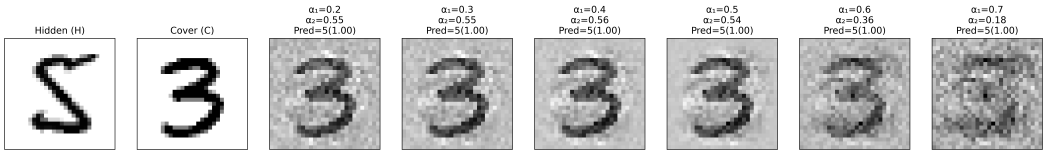


Figure A3. Comparison of stego images with different input scaling (α_1).

References

1. Akhtar, N.; Mian, A. Threat of Adversarial Attacks on Deep Learning in Computer Vision: A Survey. *IEEE Access* **2018**, *6*, 14410–14430. [CrossRef]
2. Su, J.; Vargas, D.V.; Sakurai, K. One pixel attack for fooling deep neural networks. *IEEE Trans. Evol. Comput.* **2019**, *23*, 828–841. [CrossRef]
3. Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.J.; Fergus, R. Intriguing properties of neural networks. *arXiv* **2014**, arXiv:1312.6199. [CrossRef]
4. Goodfellow, I.J.; Shlens, J.; Szegedy, C. Explaining and harnessing adversarial examples. *arXiv* **2015**, arXiv:1412.6572. [CrossRef]
5. Moosavi-Dezfooli, S.M.; Fawzi, A.; Fawzi, O.; Frossard, P. Universal Adversarial Perturbations. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 1765–1773.
6. Fowl, L.; Goldblum, M.; Chiang, P.y.; Geiping, J.; Czaja, W.; Goldstein, T. Adversarial Examples Make Strong Poisons. In *Advances in Neural Information Processing Systems* ; Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., Vaughan, J.W., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2021; Volume 34, pp. 30339–30351.
7. Ilyas, A.; Santurkar, S.; Tsipras, D.; Engstrom, L.; Tran, B.; Madry, A. Adversarial Examples are Not Bugs, They are Features. In *Advances in Neural Information Processing Systems*; Wallach, H., Larochelle H., Beygelzimer A., d'Alché-Buc F., Fox E., Garnett R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2019; Volume 32.
8. Cook, M.; Zare, A.; Gader, P. Outlier Detection Through Null Space Analysis of Neural Networks. *arXiv* **2020**, arXiv:2007.01263. [CrossRef]
9. Rezaei, H.; Sabokrou, M. Quantifying Overfitting: Evaluating Neural Network Performance through Analysis of Null Space. *arXiv* **2023**, arXiv:2305.19424. [CrossRef]
10. Sonoda, S.; Ishikawa, I.; Ikeda, M. Ghosts in neural networks: Existence, structure and role of infinite-dimensional null space. *arXiv* **2021**, arXiv:2106.04770. [CrossRef]
11. Schwab, J.; Antholzer, S.; Haltmeier, M. Deep null space learning for inverse problems: convergence analysis and rates. *Inverse Probl.* **2019**, *35*, 025008. [CrossRef]
12. Kissel, M.; Gottwald, M.; Diepold, K. Neural Network Training with Safe Regularization in the Null Space of Batch Activations. In *Artificial Neural Networks and Machine Learning, Proceedings of the ICANN 2020: 29th International Conference on Artificial Neural Networks, Bratislava, Slovakia, 15–18 September 2020*; Proceedings, Part II; Springer: Cham, Switzerland, 2020; pp. 217–228.
13. Wang, S.; Li, X.; Sun, J.; Xu, Z. Training Networks in Null Space of Feature Covariance for Continual Learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 184–193.

14. Subramanian, N.; Elharrouss, O.; Al-Maadeed, S.; Bouridane, A. Image steganography: A review of the recent advances. *IEEE Access* **2021**, *9*, 23409–23423. [CrossRef]
15. Hu, K.; Wang, M.; Ma, X.; Chen, J.; Wang, X.; Wang, X. Learning-based image steganography and watermarking: A survey. *Expert Syst. Appl.* **2024**, *249*, 123715. [CrossRef]
16. Hussain, M.; Wahab, A.W.A.; Idris, Y.I.B.; Ho, A.T.; Jung, K.H. Image steganography in spatial domain: A survey. *Signal Process. Image Commun.* **2018**, *65*, 46–66. [CrossRef]
17. Arunkumar, S.; Subramaniyaswamy, V.; Vijayakumar, V.; Chilamkurti, N.; Logesh, R. SVD-based robust image steganographic scheme using RIWT and DCT for secure transmission of medical images. *Measurement* **2019**, *139*, 426–437. [CrossRef]
18. Radeaf, H.S.; Mahmmod, B.M.; Abdulhussain, S.H.; Al-Jumaeily, D. A Steganography Based on Orthogonal Moments. In Proceedings of the International Conference on Information and Communication Technology, Baghdad, Iraq, 15–16 April 2019; pp. 147–153.
19. Xiao, B.; Luo, J.; Bi, X.; Li, W.; Chen, B. Fractional discrete Tchebyshev moments and their applications in image encryption and watermarking. *Inf. Sci.* **2020**, *516*, 545–559. [CrossRef]
20. Soria-Lorente, A.; Berres, S.; Díaz-Nuñez, Y.; Avila-Domenech, E. Hiding data inside images using orthogonal moments. *J. Inf. Secur. Appl.* **2022**, *67*, 103192. [CrossRef]
21. Yamni, M.; Daoui, A.; Abd El-Latif, A.A. Efficient color image steganography based on new adapted chaotic dynamical system with discrete orthogonal moment transforms. *Math. Comput. Simul.* **2024**, *225*, 1170–1198. [CrossRef]
22. Hassan, G.; Hosny, K.M.; Fathi, I.S. Optimized bio-signal reconstruction and watermarking via enhanced fractional orthogonal moments. *Sci. Rep.* **2025**, *15*, 30337. [CrossRef] [PubMed]
23. Wu, P.; Yang, Y.; Li, X. Image-Into-Image Steganography Using Deep Convolutional Network. In *Advances in Multimedia Information Processing, Proceedings of the PCM 2018: 19th Pacific-Rim Conference on Multimedia, Hefei, China, 21–22 September 2018*; Proceedings, Part II; Springer: Cham, Switzerland, 2018; pp. 792–802.
24. Duan, X.; Jia, K.; Li, B.; Guo, D.; Zhang, E.; Qin, C. Reversible image steganography scheme based on a U-Net structure. *IEEE Access* **2019**, *7*, 9314–9323. [CrossRef]
25. Baluja, S. Hiding images within images. *IEEE Trans. Pattern Anal. Mach. Intell.* **2019**, *42*, 1685–1697. [CrossRef]
26. Lu, S.P.; Wang, R.; Zhong, T.; Rosin, P.L. Large-Capacity Image Steganography Based on Invertible Neural Networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 10816–10825.
27. Liu, L.; Tang, L.; Zheng, W. Lossless image steganography based on invertible neural networks. *Entropy* **2022**, *24*, 1762. [CrossRef]
28. Shang, F.; Lan, Y.; Yang, J.; Li, E.; Kang, X. Robust data hiding for JPEG images with invertible neural network. *Neural Netw.* **2023**, *163*, 219–232. [CrossRef]
29. Yang, Z.; Wang, Z.; Zhang, X. A general steganographic framework for neural network models. *Inf. Sci.* **2023**, *643*, 119250. [CrossRef]
30. Martín, A.; Hernández, A.; Alazab, M.; Jung, J.; Camacho, D. Evolving Generative Adversarial Networks to improve image steganography. *Expert Syst. Appl.* **2023**, *222*, 119841. [CrossRef]
31. Wang, K.; Zhu, Y.; Chang, Q.; Wang, J.; Yao, Y. High-accuracy image steganography with invertible neural network and generative adversarial network. *Signal Process.* **2025**, *234*, 109988. [CrossRef]
32. LeCun, Y.; Cortes, C.; Burges, C.J. *MNIST Handwritten Digit Database*; AT&T Labs: Atlanta, GA, USA, 2010.
33. Xiao, H.; Rasul, K.; Vollgraf, R. Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv* **2017**, arXiv:1708.07747. [CrossRef]
34. Cohen, G.; Afshar, S.; Tapson, J.; Schaik, A.V. EMNIST: Extending MNIST to Handwritten Letters. In Proceedings of the 2017 International Joint Conference on Neural Networks (IJCNN), Anchorage, AK, USA, 14–19 May 2017.
35. Krizhevsky, A. *Learning Multiple Layers of Features from Tiny Images*; Technical Report TR-2009; University of Toronto: Toronto, ON, Canada, 2009.
36. Ball, P. Is AI leading to a reproducibility crisis in science? *Nature* **2023**, *624*, 22–25. [CrossRef]
37. Moosavi-Dezfooli, S.M.; Fawzi, A.; Frossard, P. Deepfool: A Simple and Accurate Method to Fool Deep Neural Networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 2574–2582.
38. Maynez, J.; Narayan, S.; Bohnet, B.; McDonald, R. On faithfulness and factuality in abstractive summarization. *arXiv* **2020**, arXiv:2005.00661 . [CrossRef]

39. Ji, Z.; Lee, N.; Frieske, R.; Yu, T.; Su, D.; Xu, Y.; Ishii, E.; Bang, Y.J.; Madotto, A.; Fung, P. Survey of hallucination in natural language generation. *ACM Comput. Surv.* **2023**, *55*, 1–38. [CrossRef]
40. Jesson, A.; Beltran Velez, N.; Chu, Q.; Karlekar, S.; Kossen, J.; Gal, Y.; Cunningham, J.P.; Blei, D. Estimating the hallucination rate of generative ai. *Adv. Neural Inf. Process. Syst.* **2024**, *37*, 31154–31201.

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.