Steganography of Steganographic Networks

Guobiao Li, Sheng Li*, Meiling Li, Xinpeng Zhang*, Zhenxing Qian

Fudan University
{gbli20, lisheng, mlli20, zhangxinpeng, zxqian}@fudan.edu.cn

Abstract

Steganography is a technique for covert communication between two parties. With the rapid development of deep neural networks (DNN), more and more steganographic networks are proposed recently, which are shown to be promising to achieve good performance. Unlike the traditional handcrafted steganographic tools, a steganographic network is relatively large in size. It raises concerns on how to covertly transmit the steganographic network in public channels, which is a crucial stage in the pipeline of steganography in real world applications. To address such an issue, we propose a novel scheme for steganography of steganographic networks in this paper. Unlike the existing steganographic schemes which focus on the subtle modification of the cover data to accommodate the secrets. We propose to disguise a steganographic network (termed as the secret DNN model) into a stego DNN model which performs an ordinary machine learning task (termed as the stego task). During the model disguising, we select and tune a subset of filters in the secret DNN model to preserve its function on the secret task, where the remaining filters are reactivated according to a partial optimization strategy to disguise the whole secret DNN model into a stego DNN model. The secret DNN model can be recovered from the stego DNN model when needed. Various experiments have been conducted to demonstrate the advantage of our proposed method for covert communication of steganographic networks as well as general DNN models.

Introduction

Steganography is a technique which takes advantage of the redundancy in cover media to conceal secret information. It is one of the main strategies for covert communication. Various types of cover media have been considered in literature for steganography, including image (Lu et al. 2021b; Liao et al. 2022), text (Liu et al. 2016), audio (Yi et al. 2019), video (Zhang, Cao, and Zhao 2016) or 3D Mesh (Zhou et al. 2019). Traditional steganographic schemes focus on designing handcrafted algorithms to modify the cover media subtly for data embedding, which are difficult to achieve a good balance between the undetectability and embedding capacity (termed as the capacity for short). Recently, some deep neural network (DNN)-based steganographic schemes have been proposed (Baluja 2019; Lu et al. 2021a; Jing et al. 2021), which achieve superior performance to handcrafted ones.

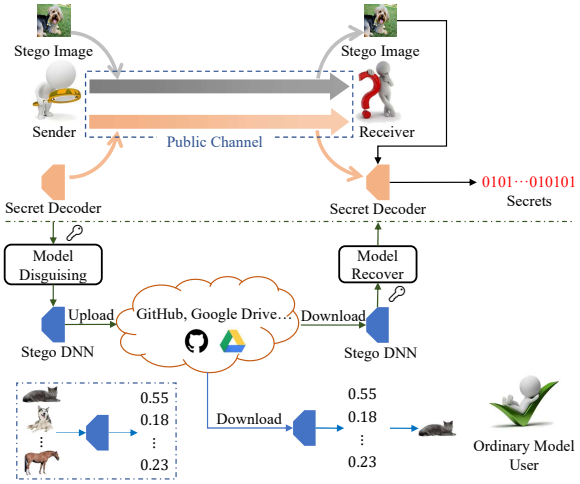

Figure 1: The application scenario of the proposed method for steganography of steganographic networks.

A DNN-based steganographic scheme usually takes advantage of the encoder-decoder structure of DNN for secret data embedding and extraction, which contains two steganographic networks: a DNN-based secret encoder and a DNN-based secret decoder (termed as the secret encoder and secret decoder for short). The secret encoder hides the secret into a cover media to obtain a stego media (i.e., media with hidden data), while the secret decoder recovers the secret from the stego media. Despite the advantage, we find that it is not convenient to deploy the DNN-based steganographic schemes in real world applications. Let's take image steganography as an example, when the receiver receives the stego image, he could only conduct the secret recovery with the possession of the secret decoder, as shown in the upper part of Fig. 1. Unlike the handcrafted steganographic tools, a DNN-based secret decoder is relatively large in size. It would not be secure for the sender and receiver to conduct another transmission for the delivery of the secret decoder.

One possible solution is to treat the secret decoder as or-

*Corresponding authors

dinary data and hide them into popular cover media such as images or videos using the existing steganographic schemes. Since the size of a secret decoder is relatively large in terms of secret information, using traditional steganographic schemes would result in the need of a large amount of cover media data due to the limited capacity, which would cause a lot of communication burden. Some DNN-based steganographic schemes (Baluja 2019; Weng et al. 2019; Jing et al. 2021) are able to achieve a high capacity. However, they are designed specifically for hiding images with errors introduced during the secret recovery. It is not appropriate to apply them on the hiding of secret decoder which will be of no use after the recovery.

To deal with the issue of covert communication of the secret decoder, in this paper, we propose a novel scheme for steganography of steganographic networks, where we disguise a secret DNN model (i.e., the secret decoder) into a stego DNN model which performs an ordinary machine learning task (termed as a stego task). As shown in the lower part of Fig. 1, the sender conducts the model disguising using our proposed method to generate a stego DNN model from the secret DNN model and a key. The stego DNN model is then uploaded to public model repositories such as the github, google drive and etc. The receiver downloads the stego DNN model and performs the model recovery using the key to restore the secret decoder. Other users (those without the key) could download the stego DNN model for an ordinary machine learning task. Compared with the existing steganographic schemes, our proposed method is tailored for the covert communication of DNN models, which has the following advantages:

1) The communication burden hardly increases before and after steganography, and there is no need to collect a large amount of cover data to accommodate the secret DNN model.
2) The behaviour of covert communication is well protected by placing another functional DNN model for an ordinary machine learning task. The stego DNN model could be placed in the model repository that is public available without being noticed.

To disguise the secret DNN model, we first select a set of filters according to the corresponding gradients when the model is back-propagated on the datasets for secret and stego tasks. Based on the selected filters, we establish a stego DNN model from the secret DNN model by a partial optimization strategy. The aforementioned process are progressively done to obtain a stego DNN model with good performance on the secret and stego tasks. The main contributions of this paper are summarized below.

1) We tackle the problem of the covert communication of the steganographic network by disguising a secret DNN model into a stego DNN model without causing notice.
2) We propose a partial optimization scheme for model disguising, where a stego DNN model is trained based on the secret DNN model for both the secret and stego tasks.
3) We propose a progressive model disguising strategy for good performance of the stego DNN model.

Related Works

According to whether DNN has been utilized for data embedding or extraction, steganographic schemes could be mainly categorized into traditional steganography and DNN-based steganography. In this section, we give a brief review regarding these two types of steganographic schemes.

Traditional Steganography

Traditional steganography usually involves the alteration of cover data using hand-craft data hiding algorithms. Liao *et al.* (Liao et al. 2022) take a batch of images as the cover data for embedding the secret message, where the image texture features are measured and considered as an indicator to determine the capacity of each image. Zhang *et al.* (Zhang, Cao, and Zhao 2016) propose a video steganography scheme by exploring the motion vector for data embedding. Lu *et al.* (Lu et al. 2021b) conduct data embedding on a halftoned image, where the histgram of the pixels is slightly modified to accommodate the secret message. Zhou *et al.* (Zhou et al. 2019) propose to convey secret data through 3D meshes, where the importance of different regions in the 3D meshes is measured for adaptive data embedding. To ensure the undetectability, the capacity of these schemes are limited. Taking the image steganography as an example, each pixel in a grayscale image (8 bits) could accommodate 1 bit of secret message at most for high undetectability.

DNN-based Steganography

Most of the DNN-based steganographic schemes are proposed for image steganography. Some works take advantage of DNN to compute the embedding cost of each pixel (Tang et al. 2017, 2020), which still use handcrafted algorithms for data embedding when the embedding costs are ready.

Recently, more and more DNN-based steganographic schemes are proposed without any handcrafted processes, which usually take advantage of the encoder-decoder structure of DNN for data embedding and extraction. Zhu *et al.* (Zhu et al. 2018) pioneer the research of such techniques, where the secrets are embedded into a cover image using an end-to-end learnable DNN. Motivated by (Zhu et al. 2018), some researchers propose to hide secret images into cover images for high capacity data embedding (Baluja 2019; Lu et al. 2021a; Jing et al. 2021; Weng et al. 2019). The concept of hiding images into images is initially proposed by Baluja *et al.* (Baluja 2019), where a secret image is concatenated with the cover image into a set of different channels to learn an image encoding network for generating the stego image. The stego image is then fed into an image decoding network to extract the secret image. The image encoding and decoding networks are jointly learnt for minimized extraction errors and image distortion. Similarly, the works in (Jing et al. 2021) and (Lu et al. 2021a) make use of the invertible neural network for image encoding and decoding, where the number of the channels in the secret image branch can be increased to remarkably improve the capacity. Weng *et al.* (Weng et al. 2019) propose to hide a secret frame into a cover frame by a temporal residual modeling technique for video steganography. Despite the advantage, the works in

(Baluja 2019; Lu et al. 2021a; Weng et al. 2019; Jing et al. 2021) are suitable for secrets in terms of images which can not be losslessly recovered in decoding.

Despite the progress in the area of steganography, little attention has been paid to the problem of covert communication of DNN models, which plays an important role in real world applications of the DNN-based steganographic schemes. In this paper, we try to tackle this problem by a network to network steganographic mechanism, which disguises the secret DNN model into a stego DNN model performing an ordinary machine learning task. The secret DNN model can be recovered from the stego DNN model with little performance degradation, while the stego DNN model is able to achieve high fidelity and undetectability.

Problem Formulation

Given a secret DNN model to be disguised, our goal is to select and tune a subset of filters to preserve the function of the secret DNN, where the remaining filters are reactivated such that the whole model works on an ordinary machine learning task (i.e., a stego task) to establish a stego DNN model for model disguising. The secret DNN model could be recovered from the stego DNN model when necessary, as shown in Fig. 2.

Let's denote the secret and stego DNN models as Θ and $\tilde{\Theta}$, where $\tilde{\Theta}$ is an optimization of Θ after the model disguising. We use two datasets, a secret dataset and a stego dataset to conduct the model disguising for the secret and stego tasks, which are denoted as $D_{se} = \{\mathbf{x}_e, \mathbf{y}_e\}$ and $D_{st} = \{\mathbf{x}_t, \mathbf{y}_t\}$, with $\mathbf{x}_e$, $\mathbf{x}_t$ being the samples and $\mathbf{y}_e$, $\mathbf{y}_t$ being the labels. We further denote the model disguising process as $\mathcal{P}$ which can be formulated below

$$\begin{aligned} &\tilde{\Theta} = \mathcal{P}(\Theta, D_{se}, D_{st}, \mathcal{K}) \\ &s.t. \quad \begin{cases} \Theta^{\mathrm{S}} \in \tilde{\Theta} \\ \mathcal{F}(\Theta^{\mathrm{S}}, \mathbf{x}_e) \rightarrow \mathbf{y}_e \\ \mathcal{F}(\tilde{\Theta}, \mathbf{x}_t) \rightarrow \mathbf{y}_t \end{cases}, \end{aligned} \tag{1}$$

where Θ^{S} is the selected subset for the secret task and $\mathcal{K}$ is a key. With the stego DNN model available, the receiver can recover the secret DNN model by

$$\Theta^{\mathrm{S}} = \mathcal{Q}(\tilde{\Theta}, \mathcal{K}), \tag{2}$$

where $\mathcal{Q}$ refers to the model recovery.

The design of $\mathcal{P}$ and $\mathcal{Q}$ should satisfy the following properties for covert communication.

1) **Recoverability**: the performance of the recovered secret DNN model should be similar to its original version on the secret task.
2) **Fidelity**: the performance of the stego DNN model should be high for the stego task.
3) **Capacity**: the size of the stego DNN model should not expand too much compared with the secret DNN model for efficient transmission.
4) **Undetectability**: it should be difficult for the attackers to identify the existence of the secret DNN model from the stego DNN model.

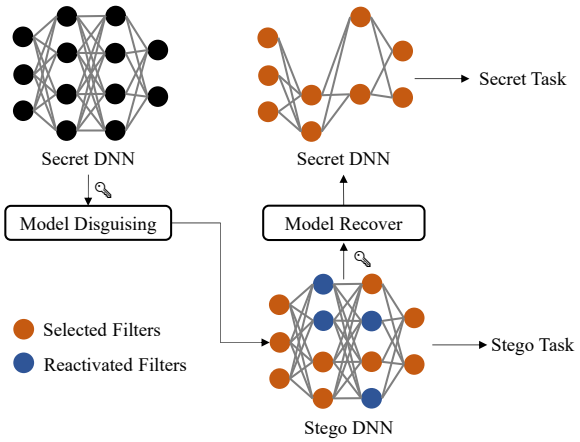

Figure 2: An illustration for the DNN model disguising and recovery.

The Proposed Method

Our model disguising starts with selecting a subset of filters from the convolutional layers in the secret DNN model, which is important on the secret task but trivial on the stego task. Then, we propose a partial optimization strategy to obtain a stego DNN model from the secret DNN model. The aforementioned process can be progressively performed for optimal performance. We also propose several strategies to deal with the parameters in other layers for real world applications.

Filter Selection

Assume the secret DNN model (i.e., Θ) contains L convolutional layers, the filters in the l-th convolutional layer can be represented as a 4-dimensional tensor: $\mathcal{W}^l \in \mathbb{R}^{d^l \times c^l \times s_1^l \times s_2^l}$, where d^l is the number of filters and each filter contains c^l 2-dimensional spatial kernels. s_1^l and s_2^l correspond to the kernel's height and width, respectively. Denote $\mathcal{W}^l_{i,:,:,:}$ as the i-th filter and $\mathcal{W}^l_{:,j,:,:}$ as the j-th channel of all the filters in the l-th layer. Ignoring fully connected layer, we have: $\Theta = \{\mathcal{W}^1, \mathcal{W}^2, \cdots, \mathcal{W}^L\}$, where $\mathcal{W}^l = \{\mathcal{W}^l_{1,:,:,:}, \mathcal{W}^l_{2,:,:,:}, \cdots, \mathcal{W}^l_{d^l,:,:,:}\}$ filter-wise or $\mathcal{W}^l = \{\mathcal{W}^l_{:,1,:,:}, \mathcal{W}^l_{:,2,:,:}, \cdots, \mathcal{W}^l_{:,c^l,:,:}\}$ channel-wise.

The purpose of the filter selection is to choose the filters that are important on the secret task but trivial on the stego task. In our discussions, we term such filters as the important filters on model disguising. For a single machine learning task, researches have indicated that the importance of the filter could be measured according to the gradients of the filter weights (Dai, Yin, and Jha 2019; Yang, Lao, and Li 2021). In our case, however, we want to identify the filters which contain weights with large gradients on the secret task and with small gradients on the stego task. For the i-th filter in the l-th convolutional layer of a secret DNN model Θ, we separately measure its importance on the secret task and stego task by GoE_i^l and GoT_i^l, where

$$\begin{cases} GoE_i^l = g(\mathcal{L}_e, \mathcal{W}_{i,:,:,:}^l, \mathbf{x}_e, \mathbf{y}_e) \\ \qquad\quad + g(\mathcal{L}_e, \mathcal{W}_{:,i,:,:}^{l+1}, \mathbf{x}_e, \mathbf{y}_e) \\ GoT_i^l = g(\mathcal{L}_t, \mathcal{W}_{i,:,:,:}^l, \mathbf{x}_t, \mathbf{y}_t) \\ \qquad\quad + g(\mathcal{L}_t, \mathcal{W}_{:,i,:,:}^{l+1}, \mathbf{x}_t, \mathbf{y}_t), \end{cases} \tag{3}$$

where

$$g(\mathcal{L}, \mathbf{w}, \mathbf{x}, \mathbf{y}) = \mathcal{AVG}\left(\left|\frac{\partial \mathcal{L}(\mathcal{F}(\Theta, \mathbf{x}), \mathbf{y})}{\partial \mathbf{w}}\right|\right), \tag{4}$$

where $\mathcal{AVG}$ returns the mean of a 3-dimensional tensor, $\mathcal{L}_e$ and $\mathcal{L}_t$ are the loss functions for the secret and stego task, respectively. Here, we take the correlation between two consecutive convolutional layers into consideration. The reason is that the feature map generated by the filter $\mathcal{W}_{i,:,:,:}^l$ in the l-th layer will be convolved with the channels $\mathcal{W}_{:,i,:,:}^{l+1}$ in the (l+1)-th layer. There is a strong correlation between the weights in $\mathcal{W}_{i,:,:,:}^l$ and $\mathcal{W}_{:,i,:,:}^{l+1}$ for a certain task. Thus, we consider both the gradients of the weights in the current filter and those in the corresponding channel of the next layer. The importance of the filter on model disguising is then computed as

$$\alpha_i^l = GoE_i^l - \lambda_g GoT_i^l, \tag{5}$$

where λ_g is a weight for balance. According to α_i^l, we select the top N most important filters for model disguising from the secret DNN model Θ, which forms the subset Θ^{S} for performing the secret task.

Partial Optimization

The purpose of partial optimization is to make Θ^{S} work on the secret task and Θ work on the stego task. First of all, we fine-tune the filters in Θ^{S} on the secret dataset by

$$\Theta^{\mathrm{S}} = \Theta^{\mathrm{S}} - \lambda_e \nabla_{\Theta^{\mathrm{S}}} \mathcal{L}_e(\Theta^{\mathrm{S}}, D_{se}), \tag{6}$$

where λ_e is the learning rate of the secret task. Then, we freeze Θ^{S}, reinitialize and reactivate the remaining filters in Θ on the stego dataset D_{st} for the following optimization problem:

$$\min_{\Theta} \sum_{(\mathbf{x}_t, \mathbf{y}_t) \in D_{st}} \mathcal{L}_t(\mathcal{F}(\Theta, \mathbf{x}_t), \mathbf{y}_t), \quad s.t.\ \Theta^{\mathrm{S}} \in \mathbb{C}, \tag{7}$$

where $\mathbb{C}$ is the constant space. To do so, we introduce M, a binary mask with the same size as that of Θ, for partial optimization, which prevents Θ^{S} from updating during the backpropagation. Let λ_t be the learning rate of the stego task and $\odot$ denotes the element-wise product, the secret DNN model is optimized below to form a stego DNN model $\tilde{\Theta}$:

$$\Theta = \Theta - \lambda_t\, \mathrm{M} \odot \nabla_{\Theta} \mathcal{L}_t(\Theta, D_{st}), \tag{8}$$

where

$$\mathrm{M}[i] = \begin{cases} 0, & \Theta[i] \in \Theta^{\mathrm{S}} \\ 1, & else \end{cases}, \tag{9}$$

where $\Theta[i]$ refers to the i-th parameter in Θ.

Algorithm 1: Progressive Model Disguising Algorithm.

Input: Secret DNN Θ, Dataset D_{se}, D_{st}, Thresholds τ_{se}, τ_{st}

Output: Stego DNN $\tilde{\Theta}$

```
t ← 1, Θ_0^S ← Θ, Θ̃_0 ← Θ, α_0^se ← 0
while α_{t-1}^se < τ_se do
    Calculate the importance of filters for filter selection from Θ_{t-1}^S by Eq. (5)
    Θ_t^S ← Top P_t important filters in Θ_{t-1}^S
    Generate mask M
    Reinitialize the remaining filters
    Conduct the model disguising according to Eq. (6) and Eq. (8) to get a stego DNN model Θ̃_t
    if α_t^st < τ_st do
        Break
    end if
    t ← t + 1
end while
Θ̃ = Θ̃_t
```

Progressive Model Disguising

To train a disguised stego DNN model from the secret DNN model with good performance for the secret and stego tasks, we propose to progressively select the important filters Θ^{S} from the secret DNN model Θ for model disguising. In particular, we iteratively select a set of important filters from Θ^{S} to train the stego DNN model, which is performed until the stego DNN model achieves satisfactory performance on the secret and stego tasks.

For simplicity, we denote the important filter set as Θ_t^{S} and the disguised stego DNN model as $\tilde{\Theta}_t$ in the t-th iteration, where we set $\Theta_0^{\mathrm{S}} = \Theta$ and $\tilde{\Theta}_0 = \Theta$ for initialization. In the t-th iteration, we select the top P_t most important filters from $\Theta_{t-1}^{\mathrm{S}}$ to form Θ_t^{S} for training a stego DNN model $\tilde{\Theta}_t$, where

$$P_t = (\lambda_p)^t \times V, \tag{10}$$

where $\lambda_p < 1$ is a decay factor and V is the number of filters in Θ. We terminate the iteration until the performance reduction of the stego DNN model is more than τ_{se} on the secret task, or less than τ_{se} and τ_{st} on the secret and stego tasks, respectively. Algorithm 1 gives the pseudo code of the aforementioned process, where α_t^{se} and α_t^{st} are the performance reduction of the stego DNN model in the t-th iteration on the secret and stego tasks.

After the progressive model disguising, we record the property of each filter in $\tilde{\Theta}$ into L binary streams $\mathbf{B} = \{b^1, b^2, \cdots, b^L\}$. Each convolutional layer corresponds to a d^l-bits binary stream b^l with d^l being the number of filters in this layer. Each bit corresponds to a filter with 0 indicating that it belongs to Θ^{S} and 1 otherwise. We consider $\mathbf{B}$ as side information and randomly select a set of parameters from the stego DNN model according to a key $\mathcal{K}$ to host $\mathbf{B}$. In particular, we embed each bit of $\mathbf{B}$ into a selected parameter as follows. We transform the parameter into an integer and flip its least significant bit according to the bit to be hidden. The parameters with hidden data are then inversely transformed

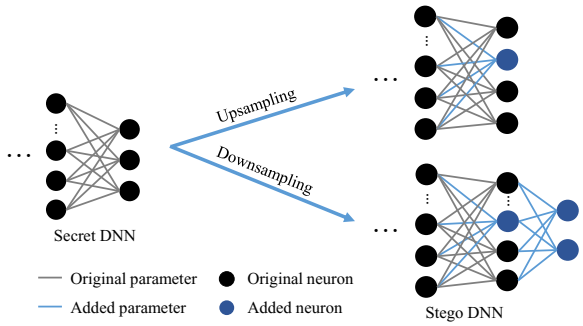

Figure 3: Output layer adaptation.

into the floating numbers to complete the embedding process.

In model recovery, the receiver uses the key $\mathcal{K}$ to recover the side information $\mathbf{B}$ from the stego DNN model, based on which we can identify the important filters Θ^{S} to establish a secret DNN model for the secret task.

Hiding Other Layers

Batch Normalization Layer. Normalization layers, such as Batch Normalization (BN) (Ioffe and Szegedy 2015), Group Normalization (GN) (Wu and He 2018), Layer Normalization (LN) (Ba, Kiros, and Hinton 2016) and Instance Normalization (IN) (Ulyanov, Vedaldi, and Lempitsky 2016), play a crucial role in accelerating the convergence of model training and performance boosting. All these normalization layers will normalize the feature maps of the intermediate layers according to some feature statistics. The BN utilizes the moving averaged mean and standard deviation of the feature maps during training, which are completely different between the secret and stego learning tasks. Therefore, when using BN, we store the corresponding feature statistics of the secret DNN model as the side information to be hidden into the stego DNN model with $\mathbf{B}$. Other types of normalization layers, *i.e.* GN, LN, IN, do not introduce additional overhead.

Fully Connected Layer. Fully connected layers are very common in DNN models, We treat them as special convolutional layers. For a fully connected layer with f neurons, we represent it as f filters with each being a $c \times 1 \times 1$ tensor, where c is the number of backward neurons that are fully connected to the layer.

Output Layer Adaptation. In real world applications, the dimension of the output layers are usually different between the secret DNN model and the stego DNN model. For simplicity, we denote the output layers of the secret and stego DNN model as $layer_e$ and $layer_t$, respectively. We further denote the number of neurons in $layer_e$ and $layer_t$ as O_e and O_t. When $O_e < O_t$, we upsample $layer_e$ to make it the same dimension as that of $layer_t$ for adaptation. When $O_e > O_t$, however, it is not appropriate to directly downsample $layer_e$ because such a strategy will inevitably distort the function of the secret DNN model. For remedy, we propose to add additional neurons into $layer_e$ to disguise it as the second to last fully connected layer. Then, we newly add a final layer for adaptation, which has the same dimension as that of $layer_t$. Fig. 3 illustrates our output layer adaptation for different cases. All the newly added parameters will not be selected into Θ^{S} to perform the secret task.

Experiments

Setup

We conduct two types of experiments for our proposed method: 1) steganography of steganographic networks (SSN), where the secret DNN model is a secret decoder, and 2) steganography of general networks (SGN), where the secret DNN model is a DNN model for a general machine learning task.

In SSN, we use the secret decoder of HiDDeN (Zhu et al. 2018) as the secret DNN model for the secret task, and we randomly select 11000 images from the COCO dataset [1] to form the secret dataset which is split into 10000/1000 images for training and testing. We use the bit error rate (BER) between the original and decoded secret information as the performance indicator of the secret decoder. We take the GTSRB (Stallkamp et al. 2012) classification task as the stego task for model disguising. We use the GTSRB dataset as the stego dataset, where 80%/20% of the images are randomly selected for training/testing. We adopt the classification accuracy (ACC) to be the indicator to evaluate the performance of the stego DNN model.

In SGN, we evaluate our method on two well-known DNN models: ResNet18 (He et al. 2016) and U-net (Ronneberger, Fischer, and Brox 2015). For ResNet18, we assume the Fashion-MNIST (Xiao, Rasul, and Vollgraf 2017) classification task as the secret task and take the CIFAR10 (Krizhevsky, Hinton et al. 2009) classification task as the stego task for model disguising. For both the secret and stego datasets, we use their default partitions for training and testing with ACC being the performance indicator.

For U-net, we assume the image denoising task as the secret task, where the U-net is trained/tested on a secret dataset with 10000/1000 images randomly selected from ImageNet (Deng et al. 2009) after adding Gaussian noise. We compute the peak single noise ratio (PSNR) to evaluate the denoising performance. We take image segmentation task as the stego task for model disguising, where the oxford-pet dataset (Parkhi et al. 2012) is adopted as the stego dataset. We randomly separate the Oxford-Pet dataset into three parts, including 6000 images for training, 1282 images for validating and 100 images for testing. We employ the Mean Intersection over Union (mIOU) as the performance indicator for the stego DNN model.

For both SSN and SGN, we set λ_g=0.01, λ_e=λ_t=0.001, λ_p=0.9, τ_{st}=0.01. For τ_{se}, we set 0.0001, 0.01, and 0.5 for the secret decoder of HiDDeN, ResNet18 and U-net, respectively. We reinitialize the remaining filters by Kaiming initialization (He et al. 2015). All the models take the BN as normalization layer and are optimized using Adam (Kingma and Ba 2014) optimizer. All our experiments are conducted on Ubuntu 18.04 system with four NVIDIA RTX 1080 Ti GPUs.

[1] https://cocodataset.org/#home

Secret DNN Model	SSN	SGN	
	HiDDeN-Secret Decoder BER ($\times 10^{-5}$)	Fashion-MNIST-Classification ACC(%)	ImageNet-Denoising PSNR(dB)
Original	6.68	93.99	28.98
Recovered	6.68	93.80	28.92

Table 1: The recoverability of the proposed method.

DNN Model	SSN	SGN	
	GTSRB-Classification ACC (%)	CIFAR10-Classification ACC (%)	Oxford-Pet-Segmentation mIOU (%)
Cover	99.97	93.46	87.68
Stego	99.81	93.02	87.05

Table 2: The fidelity of the proposed method.

Recoverability

We evaluate the recoverability of our proposed method according to the performance of the secret DNN model before and after the model disguising, i.e., the performance of the original and recovered secret DNN models (from the stego DNN model). Table 1 gives the results for different steganographic scenarios. It can be seen that, the performance of the secret decoder is not affected at all after model disguising in the SSN case. For SGN, the performance of the secret DNN models slightly decrease, which does not affect the function of the DNN models for the secret tasks.

Fidelity

To evaluate the fidelity of the stego DNN model, we train a cover DNN model which has the same architecture as the stego DNN model for the stego task. The training is conducted on the training set of the stego dataset which has been used for model disguising, where all the parameters in the cover DNN model are tuned to achieve the best performance. Furthermore, we use the same test set to evaluate the performance of the two models.

Table 2 presents the performance of the stego DNN model and the cover DNN model for both SSN and SGN. For SSN, it can be seen that the ACC of the stego DNN model is reduced with less than 0.2% after the model disguising. The results of the SGN are similar to those of SSN. In particular, after model disguising, the ACC/mIOU of the stego DNN model is slightly degraded when compared with the corresponding cover DNN model on the CIFAR100/Oxford-Pet dataset. These results indicate that the fidelity of our stego DNN model is favorable, which has little performance degradation caused by model disguising on the stego task.

Capacity

The capacity of the proposed method is evaluated in terms of the expansion rate defined as

$$e = \frac{N_{stego}}{N_{sec}} - 1, \tag{11}$$

where N_{sec} and N_{stego} refer to the number of parameters in the original secret DNN model and the corresponding stego DNN model, respectively. Thus, we have $e \geq 0$, which is closely related to the number of parameters added when adjusting the output layer. The lower the expansion rate, the higher the capacity, meaning that there is less overhead created for transmission after the model disguising. Table 3 gives the expansion rates e for different steganographic scenarios. It can be seen that, for both SSN and SGN, the expansion rates are extremely small with less that 2×10^{-3}. In the ResNet18 case, the output layers are with the same dimension for the secret and stego tasks, so the parameters do not expand at all with $e = 0$.

Expansion Rate	SSN	SGN	
	HiDDeN-Secret Decoder	ResNet18	U-Net
e ($\times 10^{-3}$)	1.65	0.00	0.14

Table 3: The capacity of the proposed method.

Classifier	Detection Accuracy (%)
SVM	47.50
MLP	50.00

Table 4: The undetectability of the proposed method.

Undetectability

To evaluate the undetectability, we first establish a model pool with different stego DNN models and cover DNN models trained. We select five benchmark datasets: MNIST (LeCun et al. 1998), Fashion-MNIST, CIFAR10, CIFAR100, GTSRB to generate $\mathrm{A}_5^2 = 20$ different secret-stego dataset pairs. Then, we train five well-known DNN models: LeNet-5 (LeCun et al. 1998), AlexNet (Krizhevsky, Sutskever, and Hinton 2012), VGG11, VGG19 (Simonyan and Zisserman 2015) and ResNet34 on the 20 dataset pairs to generate 100 different stego DNN models. Since we can only obtain 25 cover DNN models on the five datasets and five DNN models, we train each cover DNN model four times with different initialized parameters for augmentation. Take the three pairs of cover and stego DNN models (HiDDeN-Secret Decoder/ResNet18/U-net) previously trained into consideration, we obtain a total of 103 stego DNN models and 103 cover DNN models in our model pool.

Next, we compute a 100 bin-histogram from the parameters in each model to get a 100-dimensional feature vector for training a support vector machine (SVM) and a multilayer perceptron (MLP) classifier. Specifically, we randomly select 83 pairs of such features from the cover and stego DNN models for training and the remaining 20 pairs are used for testing. The detection accuracy are shown in Table 4, we can see that both the SVM and MLP classifiers are not able to detect the existence of the secret DNN models from the stego DNN models with detection accuracy close to 50%.

Comparisons

In this section, we compare our proposed method with the existing state-of-the-art steganographic schemes for hiding

Category	Method	Capacity	Recoverability BER(%) / PSNR(dB)	Undetectability P_{E}(%) / Detection accuracy (%)
Traditional steganography	Tang *et al*.(Tang et al. 2017)	200.00MB (0.40bpp)	0 / -	17.40 / -
	Yang *et al*. (Yang et al. 2019)	160.00MB (0.50bpp)	0 / -	29.71 / -
	Liao *et al.* (Liao et al. 2019)	160.00MB (0.50bpp)	0 / -	25.83 / -
	Tang *et al*. (Tang et al. 2020)	200.00MB (0.40bpp)	0 / -	32.19 / -
	Lu *et al.* (Lu et al. 2021b)	655.35MB (0.015bpp)	0 / -	34.17 / -
	Liao *et al.* (Liao et al. 2022)	160.00MB (0.50bpp)	0 / -	35.40 / -
DNN-based steganography	Zhu *et al*. (Zhu et al. 2018)	10.00MB (8.00bpp)	- / 35.70	- / 76.49
	Bakuja *et al*. (Baluja 2019)	10.00MB (8.00bpp)	- / 34.13	- / 99.67
	Weng *et al*. (Weng et al. 2019)	10.00MB (8.00bpp)	- / 36.48	- / 75.03
	Jing *et al*. (Jing et al. 2021)	10.00MB (8.00bpp)	- / 46.78	- / 55.86
	Ours	10.00MB (e=0.00)	Performance reduction 0.00% in BER, 0.19% in ACC, 0.06dB in PSNR	- / 47.50 (SVM) - / 50.00 (MLP)

Table 5: Performance comparisons among different schemes for hiding DNN models.

DNN models in terms of capacity, recoverability, and undetectability. For fair comparisons among different types of steganographic schemes, we assume the data to be hidden is a secret DNN model with size of 10MB for all the methods. The capacity is evaluated as the size of stego data, the recoverability is reported as the BER or PSNR of the recovered secrets for the existing schemes, the undetectability is indicated as the average detection error P_{E} (which is the mean of false alarm rate and missed detection rate) or the detection accuracy (50% means random guess).

Table 5 reports the comparisons among different schemes. For the traditional steganographic schemes (Liao et al. 2019; Lu et al. 2021b; Liao et al. 2022; Tang et al. 2017; Yang et al. 2019; Tang et al. 2020), we duplicate the P_{E} reported at a certain payload for undetectability from each paper, where the payload is the bit per pixel (bpp), i.e., how many bits can be embedded in an 8-bit gray-scale pixel. For the DNN-based steganographic schemes (Baluja 2019; Zhu et al. 2018; Weng et al. 2019; Jing et al. 2021), we duplicate the PSNR (on imagenet) and detection accuracy at a payload of 8bpp from the results reported in (Jing et al. 2021). Here, we assume the 10MB secret data is a natural image with meaningful content for these DNN steganographic schemes.

It can be seen from Table 5 that the size of our stego data is less than 1/10 of size of the stego data that are required using the existing traditional steganographic schemes. Compared with the existing DNN-based steganographic schemes (Zhu et al. 2018; Baluja 2019; Weng et al. 2019; Jing et al. 2021), which are not able to losslessly recover the secret and are not suitable for hiding DNN models, our scheme achieves better undetectability (closer to 50%). In addition, all the existing DNN-based steganographic schemes involve the change of data format between the secret-data (a DNN model) and the stego data (images).

To demonstrate that the existing DNN-based steganographic schemes are not suitable for hiding DNN models, we further adopt a recent DNN-based steganographic scheme (Jing et al. 2021) to hide two DNN models: the HiDDeN secret decoder and the ResNet18. Specifically, we convert the parameters of a secret DNN model to a set of binary streams to form a set of meaningless images with the size of $256\times256\times3$. Next, we embed each image into a cover image randomly selected from the COCO dataset using the encoder proposed in (Jing et al. 2021), which produces a stego image for decoding. Finally, we recover the images from the stego images using the corresponding decoder, from which we restore the DNN model. Table 6 gives the performance of the recovered DNN models using our proposed method and the method proposed in (Jing et al. 2021). It can be seen that, by using the DNN-based steganographic scheme, the recovered DNN models do not work at all.

Steganographic Method	HiDDeN-Secret Decoder BER (%)	ResNet18 ACC(%)
Jing *et al*. (Jing et al. 2021)	50.13	10.00
Ours	6.68×10^{-3}	93.80

Table 6: The performance of the recovered DNN models using different schemes.

Conclusion

In this paper, we propose a novel steganographic scheme to tackle the problem of covert communication of secret DNN models. Our scheme is able to disguise a secret DNN model into a stego DNN model which performs an ordinary machine learning task without being noticed. And the secret DNN model can be recovered from the stego DNN model on the receiver side. In model disguising, we first select a set of filters from the secret DNN model, which are important on the secret task but trivial on the stego task. We fine tune these filters on the secret task and then reactivate the other filters for the stego task according to a partial optimization strategy to train a stego DNN model. These are progressively done to obtain a stego DNN model that works well on both the secret and stego tasks. Various experiments have been conducted to demonstrate the advantage of our proposed method for covert communication of the DNN models, which achieves satisfactory performance in terms of recoverability, fidelity, capacity and undetectability.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grants 62072114, U20A20178, U20B2051, U1936214, in part by the Project of Shanghai Science and Technology Commission 21010500200.

References

Ba, J. L.; Kiros, J. R.; and Hinton, G. E. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Baluja, S. 2019. Hiding images within images. *IEEE transactions on pattern analysis and machine intelligence*, 42(7): 1685–1697.

Dai, X.; Yin, H.; and Jha, N. K. 2019. NeST: A Neural Network Synthesis Tool Based on a Grow-and-Prune Paradigm. *IEEE Transactions on Computers*, 68(10): 1487–1497.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, 248–255. Ieee.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, 1026–1034.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Ioffe, S.; and Szegedy, C. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, 448–456. PMLR.

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

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

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images. *Advances in neural information processing systems*.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11): 2278–2324.

Liao, X.; Yin, J.; Chen, M.; and Qin, Z. 2022. Adaptive Payload Distribution in Multiple Images Steganography Based on Image Texture Features. *IEEE Transactions on Dependable and Secure Computing*, 19(2): 897–911.

Liao, X.; Yu, Y.; Li, B.; Li, Z.; and Qin, Z. 2019. A new payload partition strategy in color image steganography. *IEEE Transactions on Circuits and Systems for Video Technology*, 30(3): 685–696.

Liu, F.; Luo, P.; Ma, Z.; Zhang, C.; Zhang, Y.; and Zhu, E. 2016. Security Secret Information Hiding Based on Hash Function and Invisible ASCII Characters Replacement. In *2016 IEEE Trustcom/BigDataSE/ISPA*, 1963–1969.

Lu, S.-P.; Wang, R.; Zhong, T.; and Rosin, P. L. 2021a. Large-capacity image steganography based on invertible neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10816–10825.

Lu, W.; Xue, Y.; Yeung, Y.; Liu, H.; Huang, J.; and Shi, Y.-Q. 2021b. Secure Halftone Image Steganography Based on Pixel Density Transition. *IEEE Transactions on Dependable and Secure Computing*, 18(3): 1137–1149.

Parkhi, O. M.; Vedaldi, A.; Zisserman, A.; and Jawahar, C. 2012. Cats and dogs. In *2012 IEEE conference on computer vision and pattern recognition*, 3498–3505. IEEE.

Ronneberger, O.; Fischer, P.; and Brox, T. 2015. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, 234–241. Springer.

Simonyan, K.; and Zisserman, A. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In Bengio, Y.; and LeCun, Y., eds., *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Stallkamp, J.; Schlipsing, M.; Salmen, J.; and Igel, C. 2012. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural networks*, 32: 323–332.

Tang, W.; Li, B.; Barni, M.; Li, J.; and Huang, J. 2020. An automatic cost learning framework for image steganography using deep reinforcement learning. *IEEE Transactions on Information Forensics and Security*, 16: 952–967.

Tang, W.; Tan, S.; Li, B.; and Huang, J. 2017. Automatic steganographic distortion learning using a generative adversarial network. *IEEE Signal Processing Letters*, 24(10): 1547–1551.

Ulyanov, D.; Vedaldi, A.; and Lempitsky, V. 2016. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*.

Weng, X.; Li, Y.; Chi, L.; and Mu, Y. 2019. High-capacity convolutional video steganography with temporal residual modeling. In *Proceedings of the 2019 on International Conference on Multimedia Retrieval*, 87–95.

Wu, Y.; and He, K. 2018. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, 3–19.

Xiao, H.; Rasul, K.; and Vollgraf, R. 2017. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*.

Yang, J.; Ruan, D.; Huang, J.; Kang, X.; and Shi, Y.-Q. 2019. An embedding cost learning framework using GAN. *IEEE Transactions on Information Forensics and Security*, 15: 839–851.

Yang, P.; Lao, Y.; and Li, P. 2021. Robust watermarking for deep neural networks via bi-level optimization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 14841–14850.

Yi, X.; Yang, K.; Zhao, X.; Wang, Y.; and Yu, H. 2019. AHCM: Adaptive Huffman Code Mapping for Audio Steganography Based on Psychoacoustic Model. *IEEE Transactions on Information Forensics and Security*, 14(8): 2217–2231.

Zhang, H.; Cao, Y.; and Zhao, X. 2016. A steganalytic approach to detect motion vector modification using near-perfect estimation for local optimality. *IEEE Transactions on Information Forensics and Security*, 12(2): 465–478.

Zhou, H.; Chen, K.; Zhang, W.; Yao, Y.; and Yu, N. 2019. Distortion Design for Secure Adaptive 3-D Mesh Steganography. *IEEE Transactions on Multimedia*, 21(6): 1384–1398.

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