An Automatic Cost Learning Framework for Image Steganography Using Deep Reinforcement Learning

Weixuan Tang, *Member, IEEE*, Bin Li, *Senior Member, IEEE*, Mauro Barni, *Fellow, IEEE*, Jin Li, *Senior Member, IEEE*, and Jiwu Huang, *Fellow, IEEE*

***Abstract*— Automatic cost learning for steganography based on deep neural networks is receiving increasing attention. Steganographic methods under such a framework have been shown to achieve better security performance than methods adopting hand-crafted costs. However, they still exhibit some limitations that prevent a full exploitation of their potentiality, including using a function-approximated neural-network-based embedding simulator and a coarse-grained optimization objective without explicitly using pixel-wise information. In this article, we propose a new embedding cost learning framework called SPAR-RL (Steganographic Pixel-wise Actions and Rewards with Reinforcement Learning) that overcomes the above limitations. In SPAR-RL, an agent utilizes a policy network which decomposes the embedding process into pixel-wise actions and aims at maximizing the total rewards from a simulated steganalytic environment, while the environment employs an environment network for pixel-wise reward assignment. A sampling process is utilized to emulate the message embedding of an optimal embedding simulator. Through the iterative interactions between the agent and the environment, the policy network learns a secure embedding policy which can be converted into pixel-wise embedding costs for practical message embedding. Experimental results demonstrate that the proposed framework achieves state-of-the-art security performance against various modern steganalyzers, and outperforms existing cost learning frameworks with regard to learning stability and efficiency.**

***Index Terms*— Steganography, steganalysis, reinforcement learning, embedding policy, automatic cost learning.**

Manuscript received February 19, 2020; revised June 12, 2020 and August 4, 2020; accepted August 27, 2020. Date of publication September 25, 2020; date of current version October 7, 2020. This work was supported in part by NSFC under Grant 61872244, Grant U1636202, Grant 61772349; in part by the Guangdong Basic and Applied Basic Research Foundation under Grant 2019B151502001; in part by the Guangdong Research and Development Program in Key Areas under Grant 2019B010139003; and in part by the Shenzhen Research and Development Program under Grant JCYJ20180305124325555 and Grant GJHZ20180928155814437. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Dinu Coltuc. *(Corresponding author: Bin Li.)*

Weixuan Tang and Jin Li are with the Institute of Artificial Intelligence and Blockchain, Guangzhou University, Guangdong 510006, China (e-mail: tweix@gzhu.edu.cn; jinli71@gmail.com).

Bin Li and Jiwu Huang are with the Guangdong Key Laboratory of Intelligent Information Processing and the Shenzhen Key Laboratory of Media Security, Shenzhen University, Shenzhen 518060, China (e-mail: libin@szu.edu.cn; jwhuang@szu.edu.cn).

Mauro Barni is with the Department of Information Engineering and Mathematics, University of Siena, 53100 Siena, Italy (e-mail: barni@dii.unisi.it).

Digital Object Identifier 10.1109/TIFS.2020.3025438

I. Introduction

IMAGE steganography is a technology aiming at concealing secret information within digital images. State-of-the-art steganographic methods are designed using a distortion minimization strategy [1] to modify image elements with least detectable artifacts. The distortion of the stego image is usually expressed in a simple additive form, corresponding to the sum of the embedding costs of modified elements. Several effective steganographic cost functions such as [2]–[11] have been proposed in the past decade, based on either heuristic principles [2]–[8] or statistical models [9]–[11]. Furthermore, non-additive distortion measures can also be approximated by additive costs by considering the correlations among neighboring image elements [12]–[15].

On the other hand, image steganalysis tries to detect the presence of a secret message within a stego image by looking for the artifacts caused by the data hiding process. Most modern steganalytic methods rely on hand-crafted high dimensional image statistical features [16]–[19] analyzed by classifiers trained in a supervised learning fashion [20], [21]. The detection performance can be further improved by using selection-channel information [22]–[25]. In recent years, steganalytic schemes based on deep learning, noticeably on CNNs (Convolutional Neural Networks) [26]–[32], have made tremendous progresses, owing to delicate network designs and by exploiting the domain knowledge inherited from hand-crafted feature based steganalysis.

While deep learning based steganalysis has developed rapidly, advances in deep learning based steganography are relatively slower due to the challenges associated to information hiding based on neural networks. In most of the recently proposed schemes, the competition between steganography and steganalysis can be simulated by alternatively updating a generator and a discriminator in a GAN (Generative Adversarial Network) [33]. However, the majority of the existing GAN-based information hiding schemes [34]–[36] can not guarantee error-free recovery of the hidden message, thus limiting their possible applications. A GAN-based steganographic method with reliable message extraction, named ASDL-GAN (Automatic Steganographic Distortion Learning framework with Generative Adversarial Network), has been proposed in [37]. It automatically learns the embedding costs by letting the generator output simulated stego images according to the learned embedding change probabilities, while a discriminator tries to differentiate between cover and simulated stego images. A dedicated mini neural network, called Ternary Embedding Simulator (TES), is used as a special embedding simulator so that the gradients obtained from the discriminator can be backpropagated to the generator to learn the embedding change probabilities. Under the same framework, an improved scheme called UT-GAN (U-Net and Double-Tanh Framework using GAN) is described in [38] with improved network designs, including a generator based on U-Net [39], a discriminator based on an enhanced version of Xu-Net [27], and a Double-Tanh unit to replace TES. The security performance of UT-GAN outperform those methods relying on hand-crafted cost functions [4], [5], [10].

Despite their good performance, both ASDL-GAN and UT-GAN have some potential limitations. Firstly, their neural-network-based embedding simulator cannot perfectly match an actual optimal embedding simulator. Since Sigmoid/Tanh activation functions are used in the output layer of the neural-network-based embedding simulator, the amplitude of its embedding modification is a continuous floating-point value, and does not perfectly fit the modification with discrete value in an actual optimal embedding simulator. Also, such embedding simulator needs to be manually designed and carefully trained. This procedure depends on the empirical knowledge and is rather time-consuming. Besides, the saturated behavior of Sigmoid/Tanh activation function could cause some potential issues in propagating gradients. Secondly, their optimization objectives may not fully utilize the learning ability towards pixel-level embedding costs. Note that the adversarial loss for the generator is associated to the negative loss of the discriminator, which is calculated at the coarse image level. Therefore, some important fine-grained pixel-level information may be ignored and cannot be explicitly and effectively exploited by the generator.

Built on the theory of Markov Decision Process, reinforcement learning (RL) [40] is a machine learning technique enabling algorithms to make optimized decisions. In a nutshell, RL consists of an agent taking actions within an environment, and getting rewards from the environment according to the chosen actions. The agent iteratively improves its actions based on the feedback provided by the environment. With respect to GAN, which aims at learning an underlying data distribution and whose convergence can not be proved, RL can be more appropriately used for decision making. In addition, its convergence to an optimal policy has been mathematically proved under certain conditions.

In this article, we propose a new embedding cost learning framework called SPAR-RL (Steganographic Pixel-wise Actions and Rewards with Reinforcement Learning), which avoids the limitations in ASDL-GAN/UT-GAN. In SPAR-RL, an *agent*, playing the role of the steganographer, aims at learning an optimal *embedding policy* associated to an optimal choice of the embedding costs. At the same time, an *environment* gives reward feedbacks to the actions taken by the agent. Learning is obtained by the iterative interaction between the agent and the environment according to their sampling actions and reward feedbacks. In existing image-related RL problems, the agent would typically learn the image-level policy [41]. However, such policy would lead to tremendously high search space and inefficient learning in steganographic problem. To solve the above issue, we utilize the policy network to learn embedding policy applied on individual pixel, which adapts the RL technique to practical steganographic method. In our case, the agent consists of a policy network that decomposes the embedding process into *pixel-wise actions* and aims at maximizing their overall rewards, while the role of the environment is played by an environment network giving *pixel-wise rewards* to the agent. The environment network is updated based on the simulated stego images generated by the agent. Experimental results show that under the proposed SPAR-RL framework with a well-defined reward function, the agent can obtain better embedding costs in a dynamic, interactive, and automatic way.

The main contributions of our work can be summarized as follows.

- A novel cost learning framework called SPAR-RL based on deep reinforcement learning is proposed, wherein an agent tries to learn the optimal embedding policy that maximizes the rewards from the steganalytic environment so as to yield content-adaptive costs. The costs automatically learned by SPAR-RL outperform HILL [5] and ASDL-GAN/UT-GAN with respect to security.
- An embedding policy suitable in steganography domain, which decomposes the image-level action into parallel pixel-wise actions, is proposed. Compared to conventional policies used in a general reinforcement learning framework, the proposed policy greatly reduces the complexity of cost learning.
- A *pixel-wise reward function* is designed, in which both the modification direction and the gradient backpropagated from a steganalytic neural network, i.e., environment network, with respect to each modification are considered. The proposed reward function enables the agent to learn an effective embedding policy in the SPAR-RL framework.

The rest of the paper is organized as follows. In Section II, we give the foundation of the proposed steganographic scheme, including the concept of distortion minimization and reinforcement learning. In Section III, we introduce the SPAR-RL framework detailing its components and implementations. In Section IV, we conduct extensive experiments to demonstrate the performance of steganographic methods under SPAR-RL framework. In Section V, we further investigate the effectiveness of SPAR-RL from the perspective of reward function and embedding patterns. The paper ends in Section VI, where we draw our conclusions.

II. Preliminaries

In this section, to make the paper self-contained, we first briefly review image steganography under distortion minimization strategy, given that SPAR-RL also works under such a strategy. Then, we introduce the basic ideas behind RL, including definitions and methodologies, since SPAR-RL relies on RL to learn the embedding costs.

In the rest of the paper, matrices are denoted by capital bold letters, while elements within matrices are indicated by the corresponding lowercase letters. Specifically, $\mathbf{X} = (x_{i,j})^{H\times W}$ and $\mathbf{Y} = (y_{i,j})^{H\times W}$ are used, respectively, to denote a cover image and stego image, where H and W are the height and width of the image. Calligraphic fonts are used to denote sets.

A. Image Steganography With Distortion Minimization

The distortion minimization strategy formulates the message embedding process under a given target payload capacity C as a constrained optimization problem defined by

$$\min_{\mathbf{Y}} D(\mathbf{X}, \mathbf{Y}), \quad \text{s.t. } \psi(\mathbf{Y}) = C, \tag{1}$$

where $D(\mathbf{X}, \mathbf{Y})$ is the distortion incurred by transforming $\mathbf{X}$ into $\mathbf{Y}$, and $\psi(\mathbf{Y})$ is the payload capacity of $\mathbf{Y}$ normally measured in bits.

Assuming that the embedding distortions associated to different image elements are independent, the overall distortion can be computed in an additive form as follows

$$D(\mathbf{X}, \mathbf{Y}) = \sum_{i=1}^{H}\sum_{j=1}^{W} \rho_{i,j}^{(m_{i,j})}\mathcal{I}(m_{i,j}), \tag{2}$$

where $\rho_{i,j}^{(m_{i,j})}$ is the embedding cost associated to the modification $m_{i,j} = y_{i,j} - x_{i,j}$ of the (i, j)-th image element, and the indicator function $\mathcal{I}(z)$ is defined as

$$\mathcal{I}(z) = \begin{cases} 1, & \text{if } z \neq 0, \\ 0, & \text{if } z = 0. \end{cases} \tag{3}$$

In a ternary embedding scheme, the modification is $m_{i,j} \in \mathcal{M} = \{-1, 0, +1\}$. Using practical steganographic codes such as STC (Syndrome-Trellis Codes) [1], message bits can be embedded into the image according to the embedding costs. The embedding change probabilities for a given payload capacity can be computed from the embedding costs with an optimal embedding simulator [1] as

$$p_{i,j}^{(m)} = \frac{e^{-\lambda\rho_{i,j}^{(m)}}}{\sum_{\tilde{m}\in\mathcal{M}} e^{-\lambda\rho_{i,j}^{(\tilde{m})}}}, \quad m \in \mathcal{M}, \tag{4}$$

where λ is a parameter determined by the following constraint:

$$-\sum_{i=1}^{H}\sum_{j=1}^{W}\sum_{m\in\mathcal{M}} p_{i,j}^{(m)}\log_2 p_{i,j}^{(m)} = C. \tag{5}$$

B. Reinforcement Learning

1) Definition: RL is a machine learning paradigm, in which an *agent* performs *actions* and aims to learn an optimal *policy* that maximizes the *rewards* assigned by an *environment*. The RL problem can be mathematically modeled as an MDP (Markov Decision Process) consisting of the following elements:

- the state of the agent $s \in \mathcal{S}$, where $\mathcal{S}$ is a finite set;
- the action of the agent $a \in \mathcal{A}$, where $\mathcal{A}$ is a finite set;
- the agent's reward for taking action $a \in \mathcal{A}$ in state $s \in \mathcal{S}$, denoted as $R(a, s)$.

In RL, the agent's behavior is determined by a *policy, which is a distribution over the actions given the state*, i.e., $\pi(a|s)$ abbreviated as π $(a \in \mathcal{A}, s \in \mathcal{S})$. The agent plays an episode [42] by taking stochastic actions according to the policy π and obtains a sequence of states as

$$s_0 \xrightarrow[R(a_0,s_0)]{a_0} s_1 \xrightarrow[R(a_1,s_1)]{a_1} s_2 \cdots\cdots s_{T-1} \xrightarrow[R(a_{T-1},s_{T-1})]{a_{T-1}} s_T,$$

where s_t, a_t, and $R(a_t, s_t)$ are the state, the action, and the corresponding reward at time $t \in \{0, 1, 2, \cdots\}$, respectively. A *state value function* $V_\pi(s)$ is defined as the expected total discounted rewards starting at state s and following a policy π, i.e.,

$$V_\pi(s) = \mathbb{E}_\pi\left[\sum_{t=0}^{t=T-1} \gamma^t R(a_t, s_t)|s_0 = s\right], \tag{6}$$

where $\gamma \in [0, 1]$ is the discount factor. For two policies π_1 and π_2, we say that $\pi_1 \geq \pi_2$ if $V_{\pi_1}(s) \geq V_{\pi_2}(s), \forall s$. The goal of RL is to obtain the optimal policy π^* such that

$$\pi^* \geq \pi, \quad \forall\pi. \tag{7}$$

2) Learning With Policy Network: In deep reinforcement learning, a neural network called *policy network*, can be used to learn the optimal policy. Let $\boldsymbol{\theta}$ denote the parameters of the policy network. The policy can be learned by updating the parameters $\boldsymbol{\theta}$, therefore, we indicate it as $\pi^{\boldsymbol{\theta}}$. In this article, we focus on solving a one-step MDP problem ($T = 1$), where the agent receives state s_0 (corresponding to the cover image) and takes action a_0 (corresponding to data embedding) according to the learned policy $\pi_{\boldsymbol{\theta}}$ to reach state s_1 (corresponding to the stego image), and finally obtains the reward $R(a_0, s_0)$ from the environment to update the policy network parameters $\boldsymbol{\theta}$.

In order to learn the optimal parameters $\boldsymbol{\theta}$, an objective function $J(\boldsymbol{\theta})$ is defined based on the state value function as

$$\begin{aligned} J(\boldsymbol{\theta}) &= \sum_{s\in S} d(s)[V_{\pi^{\boldsymbol{\theta}}}(s)] \\ &= \sum_{s\in S} d(s) \sum_{a\in\mathcal{A}} \pi^{\boldsymbol{\theta}}(a|s)R(a, s), \end{aligned} \tag{8}$$

where $d(s)$ is the distribution of state s. There are several ways to maximize $J(\boldsymbol{\theta})$. In our case, we apply the Policy Gradient Theorem [42] to express the derivative of the objective function without introducing the derivative of the state distribution:

$$\begin{aligned} \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) &\propto \sum_{s\in S} d(s) \sum_{a\in\mathcal{A}} \nabla_{\boldsymbol{\theta}}\pi^{\boldsymbol{\theta}}(a|s)R(a, s) \\ &= \mathbb{E}_{\pi^{\boldsymbol{\theta}}}\left[\nabla_{\boldsymbol{\theta}} \log \pi^{\boldsymbol{\theta}}(a|s)R(a, s)\right]. \end{aligned} \tag{9}$$

The parameters can then be updated by gradient ascend method:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \eta\, \nabla_{\boldsymbol{\theta}} \log\pi^{\boldsymbol{\theta}}(a|s)R(a, s), \tag{10}$$

where η is the learning rate.

After initializing the network with random parameters $\boldsymbol{\theta}$, the learning process of the policy network with *stochastic policy* [41] can be carried out iteratively as follows: 1) given a state, sample an action according to $\pi^{\boldsymbol{\theta}}$; 2) receive a reward $R(a, s)$ from the environment for the chosen action; 3) update the

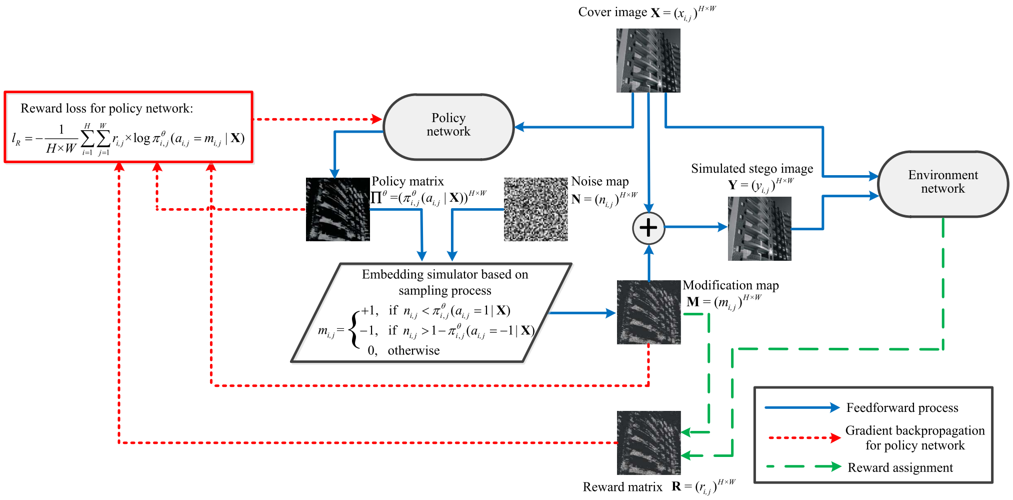

Cover image $\mathbf{X} = (x_{i,j})^{H\times W}$
Reward loss for policy network:
$l_R = -\frac{1}{H\times W}\sum_{i=1}^{H}\sum_{j=1}^{W} r_{i,j}\times\log\pi_{i,j}^{\theta}(a_{i,j} = m_{i,j} \mid \mathbf{X})$
Policy network
Environment network
Simulated stego image $\mathbf{Y} = (y_{i,j})^{H\times W}$
Policy matrix $\Pi^{\theta} = (\pi_{i,j}^{\theta}(a_{i,j} \mid \mathbf{X}))^{H\times W}$
Noise map $\mathbf{N} = (n_{i,j})^{H\times W}$
+
Embedding simulator based on sampling process
$m_{i,j} = \begin{cases} +1, & \text{if } n_{i,j} < \pi_{i,j}^{\theta}(a_{i,j} = 1 \mid \mathbf{X}) \\ -1, & \text{if } n_{i,j} > 1 - \pi_{i,j}^{\theta}(a_{i,j} = -1 \mid \mathbf{X}) \\ 0, & \text{otherwise} \end{cases}$
Modification map $\mathbf{M} = (m_{i,j})^{H\times W}$
Feedforward process
Gradient backpropagation for policy network
Reward assignment
Reward matrix $\mathbf{R} = (r_{i,j})^{H\times W}$

Fig. 1. Illustration of the proposed SPAR-RL framework. On the agent's side, the policy network takes a cover image $\mathbf{X}$ as input and outputs a policy matrix $\mathbf{\Pi}^{\boldsymbol{\theta}}$. Afterwards, modification actions are sampled according to an embedding policy and the modification map $\mathbf{M}$ is formed. The simulated stego image is obtained as $\mathbf{Y} = \mathbf{X} + \mathbf{M}$. On the other side, the environment network tries to distinguish between the cover and the simulated stego images, and returns pixel-wise rewards $\mathbf{R}$ for the sampled modification actions. The reward loss function l_R of the policy network takes the rewards, the sampled modification actions, and the embedding policy of the current iteration into account.

parameters $\boldsymbol{\theta}$ and the policy $\pi^{\boldsymbol{\theta}}$. In this way, the action with a higher reward value has a higher probability to be sampled in the next iteration round, and the optimal policy can be learned without using any supervised label.

III. The SPAR-RL Framework

A. Basic Idea

In our work, we use RL for image steganography. Specifically, the agent takes a cover image $\mathbf{X}$ as the state s_0, and then applies the embedding action a_0 according to an embedding policy learned by a policy network, and outputs a stego image $\mathbf{Y}$ representing the state s_1. An environment is set up by taking the steganalytic performance of the generated stego images into consideration to provide a reward to the agent.

The main goal of the RL framework is to derive the optimal embedding policy. In doing so, we may face two kinds of challenges. First, it is very hard for a neural network to learn lossless message encoding and decoding, as shown in previous GAN-based data hiding methods [34]–[36]. Therefore, we do not let the policy network directly learn the encoding action, instead, we let it learn the embedding costs. Such costs will then be used in conventional distortion minimization strategy wherein practical steganographic codes [1] are available. In fact, similar to the ASDL-GAN/UT-GAN framework, embedding change probabilities are actually learned in our framework and then embedding costs are obtained by solving the inverse problem in (4). Secondly, in a conventional RL framework, the action a_0 taken by the agent in state s_0 targets the entire image. This means that the modification map is sampled at the image level, and hence the amount of possible actions is tremendously high, making the entire stochastic sampling process inefficient. To cope with this problem, we decompose the image-level action into independent pixel-level actions. In this way, the optimal embedding change probabilities of each pixel can be learned efficiently by sampling them in a parallel way.

B. Framework Overview

By following the methodology described above, we now introduce a framework, called SPAR-RL, to automatically learn the embedding costs for spatial image steganography, through the interaction between an agent and an environment as depicted in Fig. 1 and described in the following.

There are two deep neural networks in our framework: one playing the role of the agent and the other the role of the environment. The policy network, parameterized by $\boldsymbol{\theta}$, takes an image $\mathbf{X} = (x_{i,j})^{H\times W}$ as input. Since an embedding policy applied on individual pixel is adopted, rather than outputting a policy for the whole image, the policy network outputs a policy matrix $\mathbf{\Pi}^{\boldsymbol{\theta}} = (\pi_{i,j}^{\boldsymbol{\theta}})^{H\times W}$, where $\pi_{i,j}^{\boldsymbol{\theta}}$ is the policy of each image element. The agent, then, samples the pixel-wise actions and determines the pixel modifications according to $\pi_{i,j}^{\boldsymbol{\theta}}$. In this way, it obtains a modification map $\mathbf{M} = (m_{i,j})^{H\times W}$, where $m_{i,j}$ is the sampled modification of each image element. A simulated stego image can then be obtained as $\mathbf{Y} = \mathbf{X} + \mathbf{M}$. The exact details of this process will be given in Section III-C.

A second deep neural network, called *environment network* and parameterized by $\boldsymbol{\omega}$, applies steganalysis to give the reward feedbacks. Typical CNN steganalyzer architecture, such as Xu-Net [27], Yedroudj-Net [28] Ye-Net [30], SRNet [32] etc., can be employed for this purpose. The environment network is trained with cover images and the corresponding simulated stego images produced by the agent. To compute the rewards, we design a reward function which takes into account both the modification direction and the gradient backpropagated from the environment network with respect to each modification. More details on reward computation will be given in Section III-D.

Note that the proposed SPAR-RL framework works in a fashion wherein the sampled actions are assigned rewards from the environment for policy adjustment and then those actions with higher rewards would have larger probabilities to be sampled in the next iteration round, while those actions

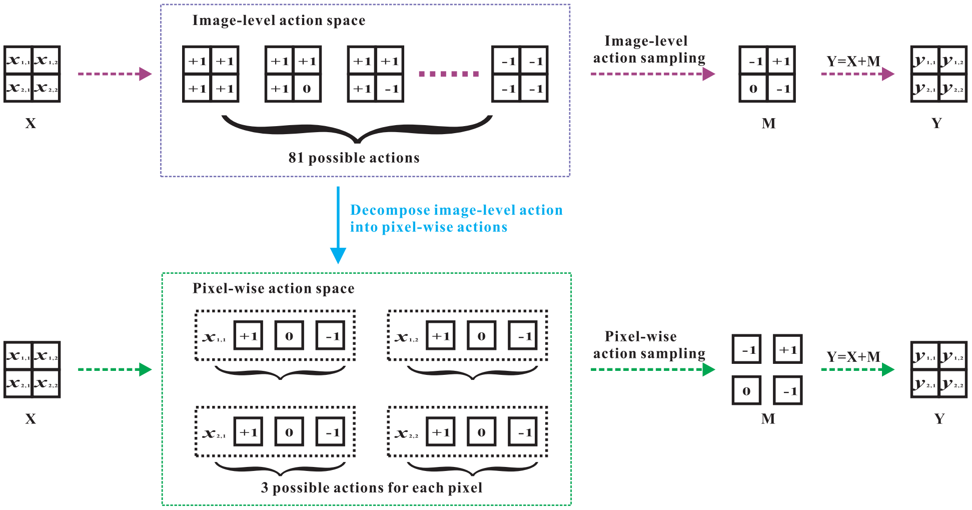


Image-level action space
81 possible actions
Image-level action sampling
Y=X+M
X
M
Y
Decompose image-level action into pixel-wise actions
Pixel-wise action space
3 possible actions for each pixel
Pixel-wise action sampling
+1
0
-1

Fig. 2. Illustration of the process of decomposing image-level sampling actions into pixel-wise sampling actions. An image of size 2 × 2 is used as an example.

with lower rewards still have some probabilities to be sampled. In this way, although the embedding process in SPAR-RL is a one-step MDP problem ($T = 1$), it can take benefit of the trial-and-error exploration nature of RL.

C. Learning the Steganographic Embedding Policy With the Policy Network

As stated above, rather than learning the policy at the image level as done in conventional RL, we propose to learn a steganographic embedding policy. This can greatly reduce the learning complexity. Let us consider, for example, the case of $\pm k$ embedding for an image of size $H \times W$. In conventional RL, the agent can only take one image-level action by sampling a modification map of size $H \times W$ at each time, and the amount of possible actions is $(2k+1)^{H \times W}$. By decomposing the image-level action into pixel-wise actions, the agent can simultaneously take $H \times W$ independent pixel-wise actions by sampling the modifications within an image in a parallel manner, each with $(2k+1)$ possible choices. Later on, the reward is assigned to each individual pixel-wise action, which is more fine-grained and content-adaptive compared to image-level rewards. In this way, the action space is greatly reduced, and the rewards provide more meaningful feedbacks, which benefit the agent to learn the optimal embedding policy more easily. The decomposition process in the case of ternary embedding ($k = 1$) is illustrated in Fig. 2.

We denote the pixel-wise action with $a_{i,j} \in \mathcal{A}$ ($\mathcal{A} = \mathcal{M}$), and its corresponding policy with $\pi^{\boldsymbol{\theta}}_{i,j}(a_{i,j}|s)$. Given an image $\mathbf{X}$ as the input, the policy network outputs a policy matrix $\boldsymbol{\Pi}^{\boldsymbol{\theta}} = (\boldsymbol{\pi}^{\boldsymbol{\theta}}_{i,j}(a_{i,j}|\mathbf{X}))^{H \times W}$, where $\boldsymbol{\pi}^{\boldsymbol{\theta}}_{i,j}(a_{i,j}|\mathbf{X})$ is the probability distribution over the action $a_{i,j} \in \mathcal{A} = \{+1, 0, -1\}$ on image element $x_{i,j}$. The aim of the policy network is to learn an optimal $\boldsymbol{\Pi}^{\boldsymbol{\theta}}$ in an element-wise manner. To this end, by adapting the RL procedures to a steganographic scenario, we let the agent sample the pixel-wise modifications according to $\boldsymbol{\pi}^{\boldsymbol{\theta}}_{i,j}(a_{i,j}|\mathbf{X})$. In this way, the embedding change probability of each pixel is

$$p^{(m)}_{i,j} = \pi^{\boldsymbol{\theta}}_{i,j}(a_{i,j} = m|\mathbf{X}), \quad m \in \mathcal{M}. \tag{11}$$

Note that in RL, the sampling process to generate the modification actions for each pixel is very similar to the optimal embedding simulator [1] used in steganography.

The entropy (measured in bits) that can be conveyed by such a simulated embedding operation is:

$$\tilde{C} = -\sum_{i=1}^{H}\sum_{j=1}^{W}\sum_{m \in \mathcal{M}} p^{(m)}_{i,j} \log_2 p^{(m)}_{i,j}. \tag{12}$$

As a result, we set a capacity objective in the policy network to let $\tilde{C}$ approximate the target capacity C. Let us indicate with $J(\boldsymbol{\theta})$ the reward optimization objective and with $O(\boldsymbol{\theta})$ the capacity optimization objective. The ultimate learning objective of the policy network can be formulated by the following optimization problem

$$\min_{\boldsymbol{\theta}}\{-\alpha J(\boldsymbol{\theta}) + \beta O(\boldsymbol{\theta})\}, \tag{13}$$

where

$$J(\boldsymbol{\theta}) = \frac{1}{HW}\sum_{s \in \mathcal{S}} d(s) \sum_{i=1}^{H}\sum_{j=1}^{W} \log\pi^{\theta}_{i,j}(a_{i,j} = m_{i,j}|s) R_{i,j}(a_{i,j} = m_{i,j}, s), \tag{14}$$

and

$$O(\boldsymbol{\theta}) = \sum_{s \in \mathcal{S}} d(s) \left(-\sum_{i=1}^{H}\sum_{j=1}^{W} \sum_{a_{i,j} \in \mathcal{A}} \pi^{\theta}_{i,j}(a_{i,j}|s) \log_2 \pi^{\theta}_{i,j}(a_{i,j}|s) - C\right)^2, \tag{15}$$

and $\alpha > 0$ and $\beta > 0$ are used to tradeoff between the reward objective and the capacity objective. Note that $d(s)$

represents the probability density function of images, and in our case it is approximated by the distribution of images in the training cover image set. $R_{i,j}(a_{i,j} = m_{i,j}, s)$ in (14) is the pixel-wise reward that will be discussed in Section III-D. The above problem can be solved by the gradient descend method:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta_{\boldsymbol{\theta}}(-\alpha \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) + \beta \nabla_{\boldsymbol{\theta}} O(\boldsymbol{\theta})). \tag{16}$$

D. Returning Rewards With Environment Network

Since the embedding policy is given at a pixel level, the reward should also be provided in a pixel-wise manner. Let $\mathbf{R} = (r_{i,j})^{H\times W}$ be the reward matrix, where $r_{i,j} = R_{i,j}(a_{i,j} = m_{i,j}, s)$ is the pixel-wise reward value. Heuristically, a reward function should assign rewards based on the capability of the modifications to resist steganalysis. We may use a neural-network based environment network to provide pixel-wise rewards. In fact, the SPAR-RL framework is flexible and different techniques including deep learning and traditional machine learning methods may be employed to assign the rewards, assuming that they can provide contribution evaluation for each action. For a fair comparison with ASDL-GAN/UT-GAN framework, we can use the discriminator network of ASDL-GAN/UT-GAN as the environment network. For example, a steganalytic architecture such as Xu-Net [27] can be used as the environment network.

Let us assume that the environment network is parameterized by $\boldsymbol{\omega}$, and that a loss function $L_{\boldsymbol{\omega}}$. Different kinds of loss functions can be used for the classification task in the environment network, such as cross-entropy loss [43]. We can derive two types of gradient information. The first one is the *parameter gradient* $\partial L_{\boldsymbol{\omega}}/\partial \boldsymbol{\omega}$ which is used to update the network parameters to improve its ability to distinguish between cover and simulated stego images. The updating process is crucial as a well-performed environment network can provide better rewards. The second type is the *modification gradient* defined as follows:

$$g_{i,j} = \partial L_{\boldsymbol{\omega}}/\partial m_{i,j}, \tag{17}$$

which is somehow similar to the gradient information used for generating adversarial examples [44] to deceive a classifier. When the modification $\mathbf{M} = (m_{i,j})^{H\times W}$ is updated in the direction of $\mathbf{G} = (g_{i,j})^{H\times W}$, the loss would increase. In other words, suppose $\overline{\mathbf{M}} = \mathbf{M} + \upsilon\mathbf{G}$ ($\upsilon > 0$), $\mathbf{X} + \overline{\mathbf{M}}$ would lead the steganalytic classifier to make a less accurate classification than $\mathbf{X} + \mathbf{M}$. Note that the modification gradient used for generating adversarial examples has been exploited in [45] for designing adversarial embedding operations. It has been shown that better steganographic security performance can be achieved even if only part of the image elements are modified according to the gradient direction. Based on the second gradient information, we design a reward function, called *DG (Direction of modification and Gradient) function*, as follows:

$$r_{i,j} = \xi \cdot \mathrm{sign}(m_{i,j}) \cdot g_{i,j}, \tag{18}$$

where ξ is a parameter controlling the reward magnitude and $\mathrm{sign}(\cdot)$ is the sign function. It can be noted that $r_{i,j} > 0$ when $m_{i,j}$ and $g_{i,j}$ have the same sign, and $r_{i,j} < 0$ when $m_{i,j}$ and $g_{i,j}$ have different signs. With such a reward function, a positive reward is assigned when the direction of the modification is exactly the same direction that increases the steganalyzer's loss function, and vice versa for the case of a negative reward. Since $g_{i,j}$ measures the sensitivity of the loss function with respect to a perturbation on $m_{i,j}$, a large reward amplitude is assigned when $|g_{i,j}|$ is large. In this way, the policy network can learn effective embedding policy over modification actions.

E. Implementation Details

There are two neural networks in the SPAR-RL framework, i.e., a policy network parameterized by $\boldsymbol{\theta}$ and an environment network parameterized by $\boldsymbol{\omega}$. Note that since the SPAR-RL framework is general, different models can be used. For example, the 25 groups of convolutional network used in [37] or the U-Net used in [38] can be adopted as the policy network, while the Xu-Net used in [37] or the enhanced Xu-Net with multiple high pass filters used in [38] can be applied as the environment network. Also note that the networks in SPAR-RL are updated in a mini-batch manner [41], where a number of N images are used in a batch for updating the parameters $\boldsymbol{\theta}$ and $\boldsymbol{\omega}$. For a given payload capacity, the detailed iterative steps of automatic embedding cost learning for ternary embedding ($\mathcal{M} = \mathcal{A} = \{+1, -1, 0\}$) with SPAR-RL are shown in the following.

- *Step 1. Generate the policy matrix by means of a policy network.*
  The policy network takes the cover image $\mathbf{X} = (x_{i,j})^{H\times W}$ as input and outputs a temporary matrix $\mathbf{Q}^{\boldsymbol{\theta}} = (q^{\boldsymbol{\theta}}_{i,j})^{H\times W}$, where $q^{\boldsymbol{\theta}}_{i,j} \in [0, 1]$. The elements of the policy matrix $\boldsymbol{\Pi}^{\boldsymbol{\theta}} = (\pi^{\boldsymbol{\theta}}_{i,j}(a_{i,j}|\mathbf{X}))^{H\times W}$ can be obtained by:
  $$\pi^{\boldsymbol{\theta}}_{i,j}(a_{i,j} = 1|\mathbf{X}) = \pi^{\boldsymbol{\theta}}_{i,j}(a_{i,j} = -1|\mathbf{X}) = q_{i,j}/2, \tag{19}$$
  $$\pi^{\boldsymbol{\theta}}_{i,j}(a_{i,j} = 0|\mathbf{X}) = 1 - q_{i,j}. \tag{20}$$
- *Step 2. Generate a simulated stego image by sampling pixel-wise modification actions according to the policy.*
  Generate a noise map $\mathbf{N} = (n_{i,j})^{H\times W}$, whose elements follow an uniform distribution in $[0, 1]$. Then, compare the noise element with the action probability defined by the policy, and obtain the pixel-wise modification as:
  $$m_{i,j} = \begin{cases} +1, & \text{if } n_{i,j} < \pi^{\boldsymbol{\theta}}_{i,j}(a_{i,j} = 1|\mathbf{X}), \\ -1, & \text{if } n_{i,j} > 1 - \pi^{\boldsymbol{\theta}}_{i,j}(a_{i,j} = -1|\mathbf{X}), \\ 0, & \text{otherwise.} \end{cases} \tag{21}$$
  In this way, the modification map $\mathbf{M} = (m_{i,j})^{H\times W}$ can be formed, and the simulated stego image can be obtained as $\mathbf{Y} = \mathbf{X} + \mathbf{M}$.
- *Step 3. Assign rewards by means of an environment network.*
  Assign pixel-wise rewards $\mathbf{R} = (r_{i,j})^{H\times W}$ according to (18) with modification map $\mathbf{M} = (m_{i,j})^{H\times W}$ obtained in Step 2 and gradient matrix $\mathbf{G} = (g_{i,j})^{H\times W}$ obtained from the environment network according to (17).

- *Step 4. Update the environment network.*
  Update the parameters $\boldsymbol{\omega}$ of the environment network with the cross-entropy loss function l_E defined as follows:
  $$l_E = -z_0' log(z_0) - z_1' log(z_1), \tag{22}$$
  where z_0 and z_1 are the environment network's Softmax outputs for the cover image $\mathbf{X}$ and the simulated stego images $\mathbf{Y}$, respectively, z_0' and z_1' are their corresponding ground-truth labels.
- *Step 5. Update the policy network.*
  Update the parameters $\boldsymbol{\theta}$ of the policy network with the total loss function l_A weighted by the reward loss l_R and the capacity loss l_C, which are defined as follows:

$$l_A = \alpha \cdot l_R + \beta \cdot l_C, \tag{23}$$

$$l_R = -\frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} r_{i,j} \cdot \log \pi_{i,j}^{\boldsymbol{\theta}}(a_{i,j} = m_{i,j} | \mathbf{X}), \tag{24}$$

$$l_C = \left(- \sum_{i=1}^{H} \sum_{j=1}^{W} \sum_{a_{i,j} \in \mathcal{A}} \pi_{i,j}^{\boldsymbol{\theta}}(a_{i,j} | \mathbf{X}) \log_2 \pi_{i,j}^{\boldsymbol{\theta}}(a_{i,j} | \mathbf{X}) - C \right)^2 . \tag{25}$$

Once the learning process is completed, the embedding policy for an input image can be obtained by the well-trained policy network, and then the corresponding embedding costs can be computed as in (4) by:

$$\rho_{i,j}^{+1} = \ln\left(\frac{1}{\pi_{i,j}^{\boldsymbol{\theta}}(a_{i,j} = +1 | \mathbf{X})} - 2 \right), \tag{26}$$

$$\rho_{i,j}^{-1} = \ln\left(\frac{1}{\pi_{i,j}^{\boldsymbol{\theta}}(a_{i,j} = -1 | \mathbf{X})} - 2 \right), \tag{27}$$

$$\rho_{i,j}^{0} = 0. \tag{28}$$

F. Comparison With Related Methods

1) Compared With ASDL-GAN/UT-GAN: The proposed SPAR-RL framework and ASDL-GAN/UT-GAN have the same goal of learning the embedding costs via deep neural networks, and may share some network components. However, by comparing Fig. 1 and Fig. 3, the proposed SPAR-RL shows two major advantages over ASDL-GAN/UT-GAN.

First, the proposed SPAR-RL uses the sampling process to implement message embedding, which can perfectly match the message embedding procedure in an optimal embedding simulator. In the ASDL-GAN/UT-GAN framework, a pre-trained neural network is used as an embedding simulator. However, since the output layer in neural-network-based embedding simulator uses Sigmoid/Tanh activation functions, modification deviation cannot be avoided, where the modification deviation is defined as the absolute difference between the target modification and the generated modification. Note that the amplitude of the generated embedding modification is a continuous floating-point value, while the amplitude of the actual embedding modification is a discrete value. Although a discrete value is a special case of a floating-point value, in the situation of generating modification by Sigmoid/Tanh function in ASDL-GAN/UT-GAN, the discrete ± 1 value is always unachievable by the generated floating-point value. An illustration of the curve of Sigmoid function in generating $+1$ approximated modification is given in the sub-figure of Fig. 3. We can observe that, the generated floating-point value can approximate $+1$, but never exactly be $+1$. Besides, it is difficult to control the tradeoff between large gradient and small modification deviation. From Fig. 3, it can be observed that the embedding modification can be generated at any point on the curve. At Point A, a large slope of the activation function can be used to avoid gradient vanishing, but the deviation from $+1$ modification is large. On the contrary, at Point B, the generated modification is close to $+1$, but the slope of the activation function at that point is correspondingly small, which may lead to gradient vanishing issue [46], [47].

Second, the proposed SPAR-RL employs fine-grained pixel-level information in the optimization objective function of the policy network, which can distinguish different modifications' contributions on deceiving the steganalyzer and can be directly used to adjust the embedding policy. By contrast, as shown in Fig. 3, in ASDL-GAN/UT-GAN, the loss function of the generator is the negative of the discriminator's Softmax loss, which is constructed by coarse-grained image-level information.

2) Compared With ADV-EMB: We remind that the gradient information is used in the DG function (18) to get effective rewards in SPAR-RL. Gradient information is also used in ADV-EMB (Adveresarial Embedding) [45] to adjust embedding costs. In addition to their different working mechanism where SPAR-RL employs reinforcement learning and ADV-EMB employs adversarial example, these two frameworks differ in the following aspects.

First, the usages of their CNN steganalyzers are different. In ADV-EMB, the CNN steganalyzer is used as the attacking target for generating adversarial images. In SPAR-RL, the CNN steganalyzer is a part of the environment to measure contributions for different modifications, and assign rewards for policy adjustment. In fact, it is not necessary to use a CNN steganalyzer as the environment in SPAR-RL. Any scheme that can evaluate the contributions of different modifications can be used to replace the CNN steganalyzer for reward assignment.

Second, their cost learning strategies are different. ADV-EMB depends on an existing steganography scheme for cost initialization, and then asymmetrically adjusts the embedding costs from different modification directions according to the CNN steganalyzer's gradients. SPAR-RL is an end-to-end framework, which automatically learns the embedding costs at different locations from scratch without using any initial embedding costs. In fact, since SPAR-RL and ADV-EMB are applied in different stages, they can be used together. We will show the performance of their combination in the experiments.

IV. Experimental Performance

In order to evaluate the performance of the proposed SPAR-RL framework, the following experiments were carried out.

- The security performance and the stability performance of SPAR-RL were compared with other cost based methods,

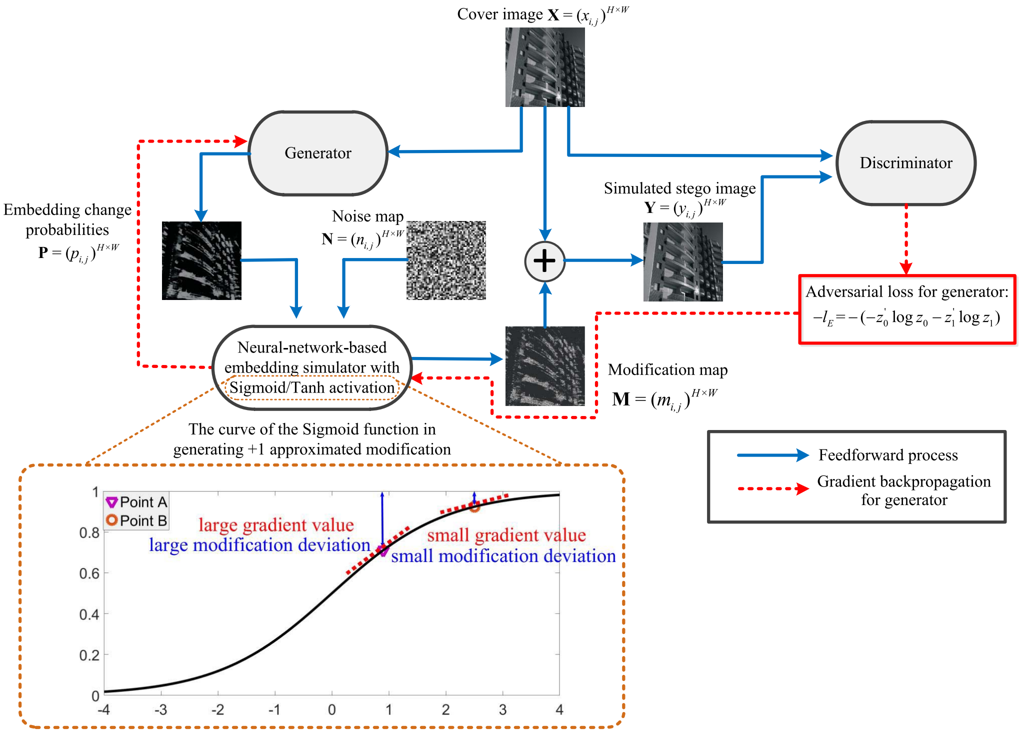

Cover image $\mathbf{X} = (x_{i,j})^{H\times W}$
Generator
Discriminator
Embedding change probabilities $\mathbf{P} = (p_{i,j})^{H\times W}$
Noise map $\mathbf{N} = (n_{i,j})^{H\times W}$
Simulated stego image $\mathbf{Y} = (y_{i,j})^{H\times W}$
Adversarial loss for generator: $-l_E = -(-z_0' \log z_0 - z_1' \log z_1)$
Neural-network-based embedding simulator with Sigmoid/Tanh activation
Modification map $\mathbf{M} = (m_{i,j})^{H\times W}$
The curve of the Sigmoid function in generating +1 approximated modification
Feedforward process
Gradient backpropagation for generator
Point A
Point B
large gradient value
large modification deviation
small gradient value
small modification deviation
1
0.8
0.6
0.4
0.2
0
-4
-3
-2
-1
0
1
2
3
4

Fig. 3. Illustration of the ASDL-GAN/UT-GAN frameworks, and the curve of Sigmoid function in generating +1 approximated modification. The generator takes a cover image **X** as input and outputs the embedding change probabilities **P**. Afterwards, a neural-network-based embedding simulator is used to output a modification map **M** according to the learned embedding change probabilities. The simulated stego image is obtained by $\mathbf{Y} = \mathbf{X} + \mathbf{M}$. The discriminator aims at distinguishing between the cover and the simulated stego images and its loss function l_E is defined in the same way as (22). The adversarial loss function of the generator is $-l_E$.

including a heuristic method and two GAN-based methods. The computational complexity of SPAR-RL was compared with that of two GAN-based methods. The general settings are detailed in Section IV-A, and the results are given in Section IV-B, IV-C, and IV-D, respectively.

- The performance when the initial costs of ADV-EMB were obtained by SPAR-RL was investigated and the results are given in Section IV-E.
- An extra image dataset, namely ALASKA dataset, was used for performance verification. The results are shown in Section IV-F.

A. Settings

1) Image Set: Following the settings in [37] and [38], two image sets were used in our experiments as described in the following.

- 256×256 *SZUBase*: it consists of 41385 images, which were firstly converted from the full-resolution raw images (in Canon CR2 format) to 512×512 grayscale images by using the same script that created the BOSSBase v1.01 [48] image set, and then down-sampled to 256×256 using the "imresize" Matlab function with the default setting.
- 256×256 *BOSSBase*: it consists of 10000 images, taken from *BOSSBase* v1.01 image set [48] and down-sampled from 512×512 to 256×256 using the "imresize" Matlab function with the default setting.

In the experiments, the 256×256 *SZUBase* was used for training the GAN-based and RL-based models, and the 256×256 *BOSSBase* was used to evaluate the security performance. In other words, once the steganographic models were trained on 256×256 *SZUBase*, they were applied to 256×256 *BOSSBase* to generate the stego images. The steganalyzers were trained and tested on 256×256 *BOSSBase*.

2) Steganographic Methods: Five steganographic methods were tested in our experiments, including four automatic cost learning methods and one hand-crafted method. The payload is measured in payload rate, defined as the payload capacity over the number of image elements, i.e., $C/(H \cdot W)$, and therefore the unit is bit per pixel (bpp). STC [1] was employed for actual embedding for all the methods. For those images that could not be successfully embedded with STC, the optimal embedding simulator [1] was used instead.

- *HILL* [5]: it utilizes one high-pass filter and two low-pass filters to heuristically generate embedding costs. HILL is very simple and is one of the best hand-crafted spatial steganographic methods.
- *ASDL-GAN* [37]: it is designed under a GAN framework, using 25 groups of convolutional neural network as the generator, Xu-Net as the discriminator, and a TES based embedding simulator. The models at the 180000-th training iteration under different payload rates were respectively used to calculate the embedding costs as in [37].
- *UT-GAN* [38]: it is designed under a GAN framework, adopting U-Net as the generator, Xu-Net enhanced with

various high-pass filters as the discriminator, and a Double-Tanh based embedding simulator. UT-GAN relies on curriculum learning for different payload rates. To be specific, the model was first trained at a payload rate of 0.4 bpp, and then fine-tuned for other payload rates. We fine-turned the best models for each payload rate as recommended by the authors [38].

- *SPAR-RL-v1*: it is designed under the proposed SPAR-RL framework, using 25 groups of convolutional neural network as the policy network and Xu-Net as the environment network. The models at the 150000-th training iteration under different payload rates were respectively used to calculate the embedding costs.
- *SPAR-RL-v2*: it is designed under the proposed SPAR-RL framework, adopting U-Net as the policy network and Xu-Net with various high-pass filters as the environment network. The models at the 150000-th training iteration under different payload rates were respectively used to calculate the embedding costs.

Note that the architecture of the policy networks and the environment networks in SPAR-RL-v1 and SPAR-RL-v2 were respectively the same as the generators and the discriminators in ASDL-GAN and UT-GAN. For a fair comparison, all the four automatic cost learning methods adopted the settings indicated in [38] by using batch size $N = 24$ and an Adam optimizer with learning rate 0.0001. The weighting parameters of the adversarial loss and capacity loss in ASDL-GAN and UT-GAN were set to $\alpha = 1$ and $\beta = 10^{-7}$. Similarly, the weighting parameters in (23) were also set to $\alpha = 1$ and $\beta = 10^{-7}$ in SPAR-RL-v1 and SPAR-RL-v2. The amplitude parameter in (18) was set to $\xi = 10^7$. Further discussions on the impact of ξ are given in Section V-B.

3) Steganalyzers: Four different steganalyzers, including two based on hand-crafted feature and two based on deep learning, were used to evaluate the security performance of the steganographic methods.

- *SRM* [16]: it uses various high-pass filters to obtain image residuals, and then extract the fourth-order co-occurrence matrix features from the quantized and truncated residuals. An FLD-based ensemble classifier was used for classification.
- *maxSRMd2* [23]: it uses selection channel information in such a way that the bin of feature is calculated as the sum of the maximum values of the four embedding change probabilities at the corresponding residuals. An FLD-based ensemble classifier was used for classification.
- *Xu-Net* [27]: it uses 6 groups of convolutional layers with a fixed high-pass filter as the convolutional mask for the first layer. 1 × 1 convolution, global pooling, and batch normalization were used to facilitate automatic feature extraction.
- *Yedroudj-Net* [28]: it utilizes 30 high-pass filters to obtain different kinds of residuals and more convolutional kernels for improving detection performance. Besides, a truncation function was also applied on the feature maps to prevent modeling outlier values in deep layers.
- *Ye-Net* [30]: it utilizes a new activation function called truncated linear unit to better capture the structure of embedding signals, and utilizes the knowledge of selection channel to boost the detection performance.

The steganalyzers were used to test the performance of the steganographic methods on 256 × 256 *BOSSBase*. For methods based on handcrafted features, namely SRM and maxSRMd2, the cover images and the corresponding stego images were pair-wisely and randomly split into a training set and a testing set with 1 : 1 proportion. For deep learning methods, including Xu-Net, Yedroudj-Net, and Ye-Net, the images were split into a training set, a validation set, and a testing set with proportions 5 : 1 : 4, in which the validation set was used for selecting the best performing steganalytic model. Security performance is evaluated as the detection error rate P_{E} on the testing set, which is obtained by the false alarm rate P_{FA} and the missed detection rate P_{MD} as follows:

$$P_{\mathrm{E}} = \min_{P_{\mathrm{FA}}} \frac{1}{2}(P_{\mathrm{FA}} + P_{\mathrm{MD}}(P_{\mathrm{FA}})). \tag{29}$$

B. Security Performance

In this part, we compare the security performance of different steganographic methods. The experimental results are shown in Table I and II. Being a conventional heuristic cost method, HILL can be regarded as a kind of baseline method. We can observe that in most cases ASDL-GAN is less secure than HILL against the SRM, maxSRMd2, Xu-Net, and Ye-Net steganalyzer. ASDL-GAN is comparable to and sometimes slightly better than HILL with respect to the Yedroudj-Net steganalyzer. UT-GAN is comparable to or better than HILL against different steganalyzers. When these methods are compared with SPAR-RL, we can draw the following conclusions:

1) Although SPAR-RL-v1 and SPAR-RL-v2 share similar network components with ASDL-GAN and UT-GAN respectively, their performance are better than their counterparts. For example, at a payload of 0.4 bpp, SPAR-RL-v1 improves 1.90%, 0.44%, 3.65%, 2.73%, and 1.60% over ASDL-GAN against five steganalyzers, respectively. SPAR-RL-v2 improves 1.00%, 0.27%, 2.47%, 3.93%, and 0.92% over UT-GAN, respectively. The experimental results indicate that the SPAR-RL framework can improve security performance over the GAN-based framework.
2) SPAR-RL-v2 outperforms SPAR-RL-v1 under all conditions significantly. For example, the improvement at 0.4 bpp against different steganalyzers is 4.45%, 4.09%, 2.20%, 2.28%, and 5.08% respectively. The main differences between SPAR-RL-v2 and SPAR-RL-v1 are the structure of the policy network and the environment network, we can conclude that an effective network design has a significant positive impact on security performance.
3) When faced with the SRM, Xu-Net, Yedroudj-Net, and Ye-Net steganalyzers, SPAR-RL-v2 always achieves the best performance among the five steganographic schemes. In particular, if we consider the maxSRMd2 steganalyzer with the best steganalytic performance due

TABLE I

P_{E} of Different Steganographic Methods Against Hand-Crafted Feature Based Steganalyzers. The Numbers in Parentheses Show the Performance Difference Between SPAR-RL and ASDL-GAN/UT-GAN With the Same Neural Network Components

Steganalyzer	Steganographic method	0.1 bpp	0.2 bpp	0.3 bpp	0.4 bpp	0.5 bpp
SRM	HILL	44.16%	37.23%	31.15%	26.20%	21.91%
	ASDL-GAN	36.19%	31.24%	25.41%	21.95%	18.04%
	SPAR-RL-v1	41.25%(↑**5.06%**)	33.51%(↑**2.27%**)	27.58%(↑**2.17%**)	23.85%(↑**1.90%**)	19.91%(↑**1.87%**)
	UT-GAN	43.53%	36.87%	32.26%	27.30%	22.52%
	SPAR-RL-v2	45.15%(↑**1.62%**)	38.43%(↑**1.56%**)	32.68%(↑**0.42%**)	28.30%(↑**1.00%**)	23.80%(↑**1.28%**)
maxSRMd2	HILL	37.63%	30.97%	26.02%	22.05%	18.83%
	ASDL-GAN	30.63%	23.85%	19.80%	17.83%	14.43%
	SPAR-RL-v1	32.18%(↑**1.55%**)	24.96%(↑**1.11%**)	19.96%(↑**0.16%**)	18.27%(↑**0.44%**)	14.98%(↑**0.55%**)
	UT-GAN	38.66%	31.04%	25.75%	22.09%	18.90%
	SPAR-RL-v2	37.71%(↓0.95%)	30.33%(↓0.71%)	25.78%(↑**0.03%**)	22.36%(↑**0.27%**)	19.54%(↑**0.64%**)

TABLE II

P_{E} of Different Steganographic Methods Against Deep Learning Based Steganalyzers. The Numbers in Parentheses Show the Performance Difference Between SPAR-RL and ASDL-GAN/UT-GAN With the Same Neural Network Components

Steganalyzer	Steganographic method	0.1 bpp	0.2 bpp	0.3 bpp	0.4 bpp	0.5 bpp
Xu-Net	HILL	43.48%	37.23%	31.38%	26.73%	22.91%
	ASDL-GAN	38.70%	37.39%	31.66%	26.76%	21.78%
	SPAR-RL-v1	45.67%(↑**6.97%**)	39.53%(↑**2.14%**)	32.78%(↑**1.12%**)	30.41%(↑**3.65%**)	26.49%(↑**4.71%**)
	UT-GAN	44.92%	38.42%	34.89%	30.14%	25.44%
	SPAR-RL-v2	46.76%(↑**1.84%**)	42.26%(↑**3.84%**)	36.42%(↑**1.53%**)	32.61%(↑**2.47%**)	29.31%(↑**3.87%**)
Yedroudj-Net	HILL	42.70%	35.41%	29.87%	24.77%	20.52%
	ASDL-GAN	39.96%	36.80%	30.02%	25.03%	19.82%
	SPAR-RL-v1	45.15%(↑**5.91%**)	37.89%(↑**1.09%**)	32.26%(↑**2.24%**)	27.76%(↑**2.73%**)	24.83%(↑**5.01%**)
	UT-GAN	43.32%	34.62%	31.83%	26.11%	23.26%
	SPAR-RL-v2	46.23%(↑**0.91%**)	39.80%(↑**5.81%**)	34.20%(↑**2.37%**)	30.04%(↑**3.93%**)	25.18%(↑**1.92%**)
Ye-Net	HILL	44.98%	40.79%	36.82%	32.01%	27.93%
	ASDL-GAN	35.96%	34.26%	30.59%	26.77%	20.36%
	SPAR-RL-v1	39.83%(↑**3.87%**)	36.01%(↑**1.75%**)	31.26%(↑**0.67%**)	28.37%(↑**1.60%**)	23.88%(↑**3.52%**)
	UT-GAN	44.52%	40.14%	36.98%	32.53%	29.23%
	SPAR-RL-v2	49.97%(↑**5.45%**)	42.53%(↑**2.39%**)	40.13%(↑**3.15%**)	33.45%(↑**0.92%**)	32.19%(↑**2.96%**)

to the exploitation of selection channel information, SPAR-RL-v2 performs comparably to UT-GAN when the payload rate is low and outperforms UT-GAN when the payload rate is no less than 0.3 bpp. It should be noted that there is no need for SPAR-RL-v2 to perform curriculum learning for different payload rates, meaning that the training process complexity for the proposed SPAR-RL framework is less than or equal to that for UT-GAN.

C. Stability Performance

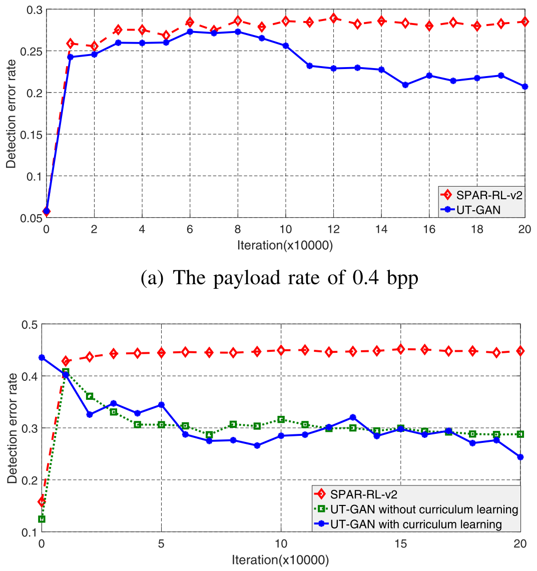
In this subsection, we evaluate the stability of the proposed framework. Specifically, we considered UT-GAN and SPAR-RL-v2 as examples, and tested their stability under two different payload rates. Experimental results are given in Fig. 4, where the x-axis denotes the training iteration and the y-axis denotes the P_{E} of the SRM steganalyzer. The following observations can be made.

1) For a payload of 0.4 bpp, both UT-GAN and SPAR-RL-v2 start performing well after the 10000-th iteration. UT-GAN obtains the best performance at the 60000-th iteration, but its performance decline rapidly after the 80000-th. SPAR-RL-v2 achieves the best performance at the 120000-th iteration and its performance are stable afterwards.
2) For a low payload of 0.1 bpp, we have investigated UT-GAN with and without curriculum learning [38]. Specifically, when curriculum learning is enabled, the model trained at 0.4 bpp is used for initialization and then it is fine-tuned to work at 0.1 bpp. In this case, the best performance of UT-GAN is obtained by

TABLE III

COMPARISON OF THE NUMBER OF PARAMETERS AND COMPUTATIONAL COST FOR DIFFERENT MODELS

Module	Metrics	ASDL-GAN	SPAR-RL-v1	UT-GAN	SPAR-RL-v2
Generator/	Parameters	1.63×10^5	1.63×10^5	2.59×10^6	2.59×10^6
Policy network	FLOPs	1.07×10^{10}	1.07×10^{10}	2.70×10^8	2.70×10^8
Embedding	Parameters	3×10^2	–	1	–
simulator	FLOPs	1.97×10^7	6.55×10^4	6.55×10^4	6.55×10^4
Discriminator/	Parameters	1.44×10^4	1.44×10^4	1.56×10^4	1.56×10^4
Environ-ment network	FLOPs	7.35×10^7	7.35×10^7	1.47×10^8	1.47×10^8
Overall	Parameters	1.77×10^5	1.77×10^5	2.60×10^6	2.60×10^6
	FLOPs	1.08×10^{10}	1.07×10^{10}	4.17×10^8	4.17×10^8



(a) The payload rate of 0.4 bpp

(b) The payload rate of 0.1 bpp

Fig. 4. Stability performance of SPAR-RL-v2 and UT-GAN. Performance is evaluated by SRM steganalyzer.

directly using the model trained at 0.4 bpp without any further fine-turning. When curriculum learning is not used, UT-GAN obtains the best performance at the 10000-th iteration, but performance declines afterwards. Under the two different cases, UT-GAN achieves similar performance after the 60000-th iteration. In contrast, SPAR-RL-v2 starts performing well soon after the 10000-th iteration. Note that, as we said, SPAR-RL-v2 does not need curriculum learning and is trained directly at the required payload rate.

The experimental results indicate that SPAR-RL is more stable and it converges quite rapidly. The reason may be due to the fact that SPAR-RL uses stochastic policy to directly sample modifications according to the learned policy, which avoids the gradient vanishing problems associated to the neural-network-based embedding simulator in UT-GAN and ASDL-GAN.

D. Computational Complexity

We have compared the number of parameters and computational cost for different deep learning models in Table III, wherein the computational cost is measured in terms of FLOPs (floating point operations). Specifically, we counted FLOPs for all convolutional layers and fully-connected layers in the corresponding network architecture. The following observations can be made.

1) The policy network and environment network in SPAR-RL-v1/SPAR-RL-v2 are respectively the same as the generator and discriminator in ASDL-GAN/UT-GAN. As a result, they have almost equal overall computational cost.
2) The computational cost of the embedding simulator is non-negligible. The reason is that each element is independently processed by the embedding simulator and thus increasing the computational cost.
3) Although the policy network in SPAR-RL-v2 has more parameters than that in SPAR-RL-v1, its computational cost is lower. The reason is that the policy network in SPAR-RL-v2 adopts the architecture of U-Net, wherein the computational cost for those feature maps with smaller sizes are largely reduced.

E. Combination With SPAR-RL and ADV-EMB

In this subsection, we show the performance of SPAR-RL, ADV-EMB [45], and their combination. According to the settings in [45], we performed experiments as follows. The $10,000$ 512×512 images from BOSSBase were randomly split into three disjoint cover subsets according to proportion $2 : 1 : 1$, which are denoted as $\mathcal{C}^0$, $\mathcal{C}^{1trn}$, and $\mathcal{C}^{1tst}$, respectively. Their corresponding stego subsets are denoted as $\mathcal{S}^0$, $\mathcal{S}^{1trn}$, and $\mathcal{S}^{1tst}$, respectively. In ADV-EMB, two steganographic methods, namely, S-UNIWARD applied in [45] and the proposed SPAR-RL-v2, were used to obtain initial costs. All stego images were generated by optimal embedding simulator. Xu-Net was used as the target steganalyzer and was trained with $\mathcal{C}^0$ and $\mathcal{S}^0$. Aiming at fooling the trained Xu-Net steganalyzer $\phi_{\mathcal{C}^0,\mathcal{S}^0}$, adversarial embedding was applied on $\mathcal{C}^{1trn}$, and $\mathcal{C}^{1tst}$ to generate adversarial stego subsets $\mathcal{Z}^{1trn}$, and $\mathcal{Z}^{1tst}$. The SRM steganalyzer for performance evaluation was trained with $\mathcal{C}^{1trn}$ and $\mathcal{S}^{1trn}/\mathcal{Z}^{1trn}$, and was tested on $\mathcal{C}^{1tst}$ and $\mathcal{S}^{1tst}/\mathcal{Z}^{1tst}$. The training set and the testing set were randomly split for 10 times to compute the averaged detection accuracy with the SRM steganalyzer.

From Table IV, we can observe that our proposed SPAR-RL-v2 outperforms ADV-EMB (initialized with

TABLE IV

SECURITY PERFORMANCE COMPARISON BETWEEN ADV-EMB AND SPAR-RL

Steganography	Testing set	0.2 bpp	0.4 bpp
S-UNIWARD	$\{\mathcal{C}^{1tst},\mathcal{S}^{1tst}\}$	32.71%	21.49%
ADV-EMB (with S-UNIWARD)	$\{\mathcal{C}^{1tst},\mathcal{Z}^{1tst}\}$	34.24%	22.56%
SPAR-RL-v2	$\{\mathcal{C}^{1tst},\mathcal{S}^{1tst}\}$	36.05%	25.85%
ADV-EMB (with SPAR-RL-v2)	$\{\mathcal{C}^{1tst},\mathcal{Z}^{1tst}\}$	36.98%	27.30%

S-UNIWARD) [45]. Specifically, it enhances the security performance by 1.81% and 3.29% on 0.2 bpp and 0.4 bpp, respectively. Since SPAR-RL and ADV-EMB are applied in different stages of cost generation, they can also be used together. By initializing ADV-EMB with SPAR-RL-v2, the performance can be further improved by approximately 0.9% $\sim$ 1.5%.

F. Performance on ALASKA Dataset

In this subsection, we performed experiments on 37511 images of ALASKA dataset. The full resolution raw images were demosaicked using executable program *DCRAW*, and then central cropped along the longer dimension to obtain square images. The images obtained in this way were converted to grayscale and then down-sampled to 256 × 256 using the "imresize" Matlab function with the default setting. The settings of training the deep learning models and its evaluation were the same as the ones used in previous experiments, except that we replaced SZUBase with ALASKA in training SPAR-RL-v1/SPAR-RL-v2 and ASDL-GAN/UT-GAN. The training set and testing set on 256 × 256 *BOSSBase* were randomly split for 10 times to compute the average detection accuracy with SRM steganalyzer. Experimental results are shown in Fig. 5. It can be observed that similar to the results obtained on SZUBase, SPAR-RL-v1 and SPAR-RL-v2 have better security and stability performance than ASDL-GAN and UT-GAN, respectively.

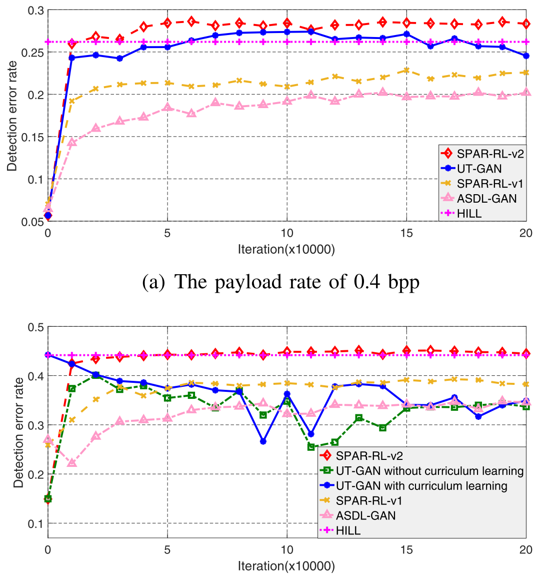
V. FURTHER INVESTIGATION

In order to further investigate the proposed SPAR-RL framework from different perspectives, the following experiments were carried out.

- The effectiveness of the reward function in SPAR-RL was investigated. The experimental results are given in Section V-A.
- The impact of reward amplitude on security performance was explored. The experimental results are given in Section V-B.
- The pixel-level embedding patterns were analyzed to better understand the effect of the proposed SPAR-RL. The statistical results are shown in Section V-C.

A. Investigation on Reward Function

The DG function defined in (18) is used in the SPAR-RL framework as the reward function for pixel-wise reward assignment. In this subsection, we further investigate the effectiveness of the DG function. Specifically, we took SPAR-RL-v2 as an example, and tested its security performance with different types of reward functions. In particular, we tested the following two variants.



(a) The payload rate of 0.4 bpp

(b) The payload rate of 0.1 bpp

Fig. 5. Security performance on ALASKA dataset with SRM steganalyzer.

1) Variant Type I, Reward With Uniform Amplitude: In the DG function, the amplitude of the reward is different for each element and it is set to be proportional to the amplitude of the modification gradient. To investigate the effectiveness of content-adaptive amplitude, we used an uniform amplitude as Variant Type I for comparison. In particular, we let the amplitude be equal to the average of the derivative amplitude over the whole image:

$$r'_{i,j} = \xi \cdot \mathrm{sign}(m_{i,j}) \cdot \bar{g}, \tag{30}$$

where $\bar{g} = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} |g_{i,j}|$.

2) Variant Type II, Reward With Inverse Sign: In the DG function, the reward is positive if the modification and the modification gradient have the same sign, and negative vice versa. For comparison, we used the negative of the modification gradient in the reward function as Variant Type II:

$$r''_{i,j} = \xi \cdot \mathrm{sign}(m_{i,j}) \cdot (-g_{i,j}) = -r_{i,j}. \tag{31}$$

The experimental results are given in Table V. Upon inspection of the results, we see that the performance of Variant Type I is inferior to the original DG function with a drop of more than 3% against the SRM steganalyzer and 2% against Xu-net (at 0.4 bpp). The security performance of Variant Type II also drops significantly. As Variant Type I still utilizes the reward direction, it is much better than Variant Type II. These phenomena imply that the sign of the modification gradient is the most important part in the reward function, while its amplitude can be used to further improve the security performance.

TABLE V

P_E OF DIFFERENT SETTINGS OF THE REWARD FUNCTION IN SPAR-RL-v2

Reward Function	**0.1 bpp**		**0.4 bpp**	
	SRM	Xu-Net	SRM	Xu-Net
Original DG function	45.15%	46.76%	28.30%	32.61%
Variant Type I	42.09%	45.03%	25.21%	30.19%
Variant Type II	5.82%	1.43%	2.73%	1.02%

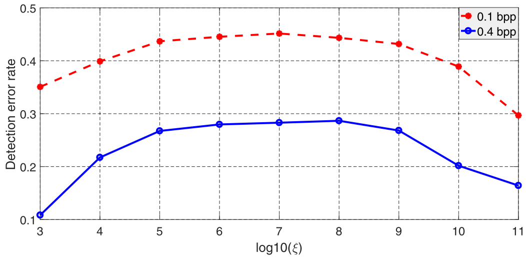


Fig. 6. P_E dependence on ξ in SPAR-RL-v2. Performance is evaluated by SRM steganalyzer.

B. Investigation on Reward Amplitude

As shown in (18), ξ is a parameter to control the reward amplitude. In this subsection, we study how the security performance changes with ξ. Experimental results for SPAR-RL-v2 against the SRM steganalyzer are shown in Fig. 6. It can be observed that the performance are stable for $10^5 \leq \xi \leq 10^9$ and the best security performance are achieved when $10^7 \leq \xi \leq 10^8$ at both 0.1 and 0.4 bpp. If ξ is too small, the reward signal would be rather weak, and the agent cannot efficiently update the embedding costs according to the feedback from the environment. On the contrary, if ξ is too large, the capacity loss in (25) may be overwhelmed by the reward loss in (24) thus making it difficult to optimize the two kinds of loss simultaneously.

C. Investigation on Embedding Patterns

In this subsection, we further analyze the pixel-level embedding patterns of the proposed SPAR-RL from three aspects.

Firstly, we investigated where the embedding modifications took place. We computed a metric called *averaged residual amplitude for modified pixels* according to the following steps.

1) For a specific pair of cover image $\mathbf{X} = (x_{i,j})^{H\times W}$ and stego image $\mathbf{Y} = (y_{i,j})^{H\times W}$, obtain the corresponding modification map $\mathbf{M} = (m_{i,j})^{H\times W}$, where $m_{i,j} = y_{i,j} - x_{i,j}$.
2) Apply high-pass filtering on the cover image $\mathbf{X}$, and then take the residual as

$$\mathbf{T} = (t_{i,j})^{H\times W} = \mathbf{X} \otimes \mathbf{H}, \tag{32}$$

where $\mathbf{H}$ denotes the high-pass filter, and $\otimes$ denotes convolution operation. Without loss of generality, KV filter [16] was used as the high-pass filter $\mathbf{H}$ in our experiment.

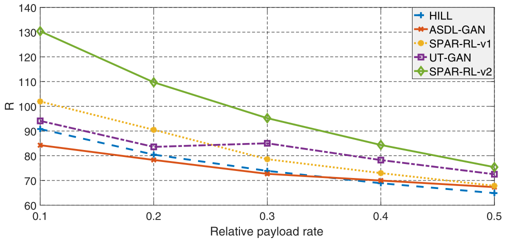


Fig. 7. Averaged residual amplitude for modified pixels.

3) Compute the averaged absolute residual amplitude for the modified pixels as

$$R = \frac{\sum_{i=1}^{H}\sum_{j=1}^{W} |t_{i,j}| \times \delta(|m_{i,j}| - 1)}{\sum_{i=1}^{H}\sum_{j=1}^{W} \delta(|m_{i,j}| - 1)}, \tag{33}$$

where

$$\delta(x) = \begin{cases} 1, & x = 0, \\ 0, & otherwise. \end{cases} \tag{34}$$

It is expected that the larger the R value, the more complex areas the embedding happens in. We performed experiments on 10,000 256×256 BOSSBase images and got the averaged results for different payload rates. The results are given in Fig. 7. It can be observed that SPAR-RL-v2 and SPAR-RL-v1 have larger R values than UT-GAN and ASDL-GAN, respectively, indicating that SPAR-RL tends to modify pixels in high-frequency areas.

Secondly, we investigated the concentration of modifications. We computed the *frequencies of occurrences of the amount of modifications within a fixed size region*. The experiments were conducted with the following steps.

1) Obtain the modification map $\mathbf{M} = (m_{i,j})^{H\times W}$.
2) For all 3×3 patches (obtained in an overlapping fashion) within the modification map, compute the frequencies of occurrences of the amount of modifications by

$$Q(k) = \frac{\sum_{i=1}^{H-2}\sum_{j=1}^{W-2} \delta(\sum_{p=0}^{2}\sum_{q=0}^{2} |m_{i+p,j+q}| - k)}{(H-2)\times(W-2)}, \tag{35}$$

where $k = 0, 1, 2, \cdots, 9$.

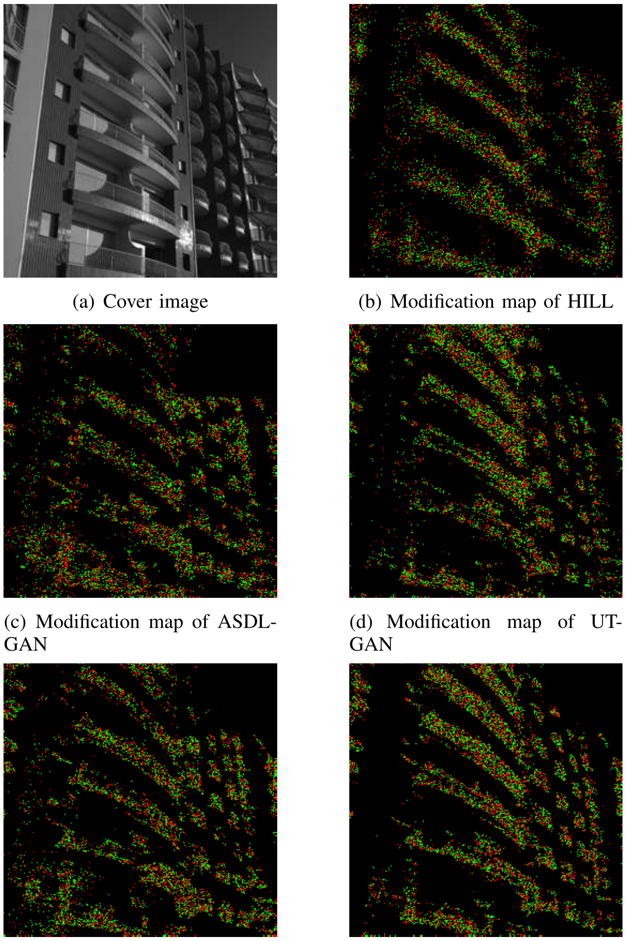
We performed experiments on 10,000 256×256 BOSSBase images and got the averaged results for the payload rate of 0.4 bpp. The results are given in Table VI. It can be observed that compared to ASDL-GAN/UT-GAN, SPAR-RL-v1/SPAR-RL-v2 have larger ratio on $Q(0)$, $Q(3)$, $Q(4)$, $Q(5)$, $Q(6)$, $Q(7)$, $Q(8)$, $Q(9)$, while have smaller ratio on $Q(1)$ and $Q(2)$, indicating that the modifications of SPAR-RL are more concentrated.

Note that concentrating the modifications within high-frequency areas is the result of the interaction between steganography and steganalysis. On the steganalysis side, existing steganalyzers utilize or learn high-pass filters to extract embedding traces from high-frequency residuals. On the steganography side, it is expected that the modifications are concentrated in high-frequency areas which are hard to detect by steganalyzers. In the proposed SPAR-RL, the pixels in high-frequency areas obtain larger rewards assigned by the

TABLE VI

FREQUENCIES OF OCCURRENCES OF THE AMOUNT OF MODIFIED PIXELS IN 3 × 3 PATCHES (0.4BPP).THE STATISTICS OF HILL ARE SERVED AS THE BASELINE WITH YELLOW COLOR. IN THE SAME ROW, THE VALUE WITH GREEN COLOR IS SMALLER THAN THE BASELINE, WHILE THE VALUE WITH RED COLOR IS LARGER THAN THE BASELINE

	Q(0)	Q(1)	Q(2)	Q(3)	Q(4)	Q(5)	Q(6)	Q(7)	Q(8)	Q(9)
HILL	0.6055	0.1698	0.0975	0.0592	0.0352	0.0193	0.0091	0.0034	0.0009	0.0001
ASDL-GAN	0.5886	0.1811	0.1003	0.0612	0.0371	0.0199	0.0086	0.0027	0.0005	0.0000
SPAR-RL-v1	0.6110	0.1447	0.0880	0.0624	0.0447	0.0284	0.0143	0.0052	0.0012	0.0001
UT-GAN	0.6219	0.1411	0.0812	0.0587	0.0443	0.0298	0.0157	0.0058	0.0013	0.0001
SPAR-RL-v2	0.6220	0.1307	0.0783	0.0588	0.0471	0.0340	0.0193	0.0078	0.0019	0.0002


(a) Cover image

(b) Modification map of HILL

(c) Modification map of ASDL-GAN

(d) Modification map of UT-GAN

(e) Modification map of SPAR-RL-v1

(f) Modification map of SPAR-RL-v2

Fig. 8. Illustration of the cover image "08178.pgm" from 256 × 256 *BOSSBase* and its modification map of different steganographic methods on 0.4 bpp, where red points indicate modifications of +1, and green points indicate modifications of −1.

TABLE VII

THE CHANGE RATE AND FFC FOR DIFFERENT METHODS ON 0.4 BPP

Steganography	**Change rate**	**FFC**		
		2nd order	**3rd order**	**4th order**
HILL	9.60%	0.64%	0.11%	0.02%
HILL-CMD	12.39%	2.01%	0.53%	0.12%
ASDL-GAN	9.98%	0.66%	0.11%	0.02%
SPAR-RL-v1	10.78%	0.86%	0.16%	0.03%
UT-GAN	10.64%	0.84%	0.15%	0.03%
SPAR-RL-v2	11.30%	1.02%	0.21%	0.05%

high-pass filter based environment network, and have more chances to be modified.

Finally, considering that synchronizing modification directions may bring benefit to security performance [13], [14], we investigated whether the modification directions were synchronized in SPAR-RL. In Fig. 8, we show an example image and its corresponding modification maps obtained by different steganographic methods. It can be observed that the synchronizing effect is not obvious. We conducted experiments to further investigate the effect as follows. Following the settings in [13], an evaluation metric, called *FCC (frequency of consecutive changes)*, was used to compute the average frequency of occurrences in the row/column direction for consecutive positive/negative changes. The n-th order FCC, denoted by $F(n)$, is obtained by

$$F(n) = \frac{1}{4}(H(n,1) + H(n,-1) + V(n,1) + V(n,-1)) \tag{36}$$

where

$$H(n,k) = \frac{\sum_{i=1}^{n_1} \sum_{j=1}^{n_2-n+1} \delta(m_{i,j}-k)\ldots\delta(m_{i,j+n}-k)}{n_1(n_2-n+1)}, \tag{37}$$

$$V(n,k) = \frac{\sum_{i=1}^{n_1-n+1} \sum_{j=1}^{n_2} \delta(m_{i,j}-k)\ldots\delta(m_{i+n,j}-k)}{(n_1-n+1)n_2}, \tag{38}$$

Experimental results for the payload rate of 0.4 bpp on $10,000$ 256×256 BOSSBase images are shown in Table VII. It can be observed that the FFC value for SPAR-RL-v1 and SPAR-RL-v2 are slightly larger than that for ASDL-GAN and UT-GAN, respectively. It should be noted that in SPAR-RL-v1 and SPAR-RL-v2, the cost for positive and negative modifications are set to be the same, and the pixels are sampled independently. The reason for higher FFC value for SPAR-RL-v1 and SPAR-RL-v2 is that they have larger change rate and stronger effect of modification concentration. Please note that the FFC value for HILL-CMD [13] is even larger than that for SPAR-RL-v1 and SPAR-RL-v2. The reason is that in HILL-CMD, the embedding costs from different directions are manually adjusted for the purpose of synchronizing modification directions, while SPAR-RL is not specifically designed for such a purpose and the cost from different modification directions are set to be the same.

VI. Conclusion

In this article, we have proposed a brand new cost learning framework, called SPAR-RL. The proposed framework works in a RL setting wherein the agent utilizes a policy network to learn the optimal embedding policy by maximizing the rewards from the environment. The environment network is employed to assign pixel-wise rewards to modifications according to their capability to deceive the steganalyzer. By means of the interaction between the agent and the environment, SPAR-RL can automatically learn embedding costs with satisfied security and stability performance. Extensive experiments have been conducted to assess the performance of SPAR-RL, and the following conclusions can be drawn.

1) When the same network architecture is used, steganographic methods based on SPAR-RL obtain better security performance than those build under the ASDL-GAN and UT-GAN frameworks.
2) The training process of SPAR-RL is stable and there is no requirement of extra curriculum learning for different payload rates.
3) The design of the reward function is critical to obtain good performance. Our proposed DG function is effective to guide the agent to learn a secure embedding policy.

The proposed framework represents a promising approach for automatic cost learning with reinforcement learning. Further investigation may include the following aspects. Firstly, other forms of reward functions and the way of returning rewards, such as those used in traditional machine learning methods, may be investigated under the SPAR-RL framework. Secondly, the security performance against selection channel steganalyzers should be investigated, and might possibly be improved by feeding the information of embedding change probabilities to the environment network. Thirdly, to consider the correlation among neighboring pixels, we can decompose the image-level action into patch-level actions, and the policy network may learn the embedding pattern of synchronizing modification directions. Fourthly, to capture the interactions of pixel modifications during embedding, we may consider to explore multi-step MDP in RL and adapt it for steganography.

Acknowledgment

The authors would like to thank Dr. Yang for sharing the code of UT-GAN [38], and for the useful suggestion regarding its training process. They also would like to thank appreciate anonymous reviewers for their valuable comments.

References

[1] T. Filler, J. Judas, and J. Fridrich, “Minimizing additive distortion in steganography using syndrome-trellis codes,” *IEEE Trans. Inf. Forensics Security*, vol. 6, no. 3, pp. 920–935, Sep. 2011.

[2] T. Pevný, T. Filler, and P. Bas, “Using high-dimensional image models to perform highly undetectable steganography,” in *Proc. 12th Int. Conf. Inf. Hiding*, Jun. 2010, pp. 161–177.

[3] V. Holub and J. Fridrich, “Designing steganographic distortion using directional filters,” in *Proc. IEEE Int. Workshop Inf. Forensics Secur.*, Dec. 2012, pp. 234–239.

[4] V. Holub, J. Fridrich, and T. Denemark, “Universal distortion function for steganography in an arbitrary domain,” *EURASIP J. Inf. Secur.*, vol. 2014, no. 1, pp. 1–13, Jan. 2014.

[5] B. Li, M. Wang, J. Huang, and X. Li, “A new cost function for spatial image steganography,” in *Proc. IEEE Int. Conf. Image Process.*, Oct. 2014, pp. 4026–4210.

[6] L. Guo, J. Ni, W. Su, C. Tang, and Y.-Q. Shi, “Using statistical image model for JPEG steganography: Uniform embedding revisited,” *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 12, pp. 2669–2680, Dec. 2015.

[7] W. Zhou, W. Zhang, and N. Yu, “A new rule for cost reassignment in adaptive steganography,” *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 11, pp. 2654–2667, Nov. 2017.

[8] S. Kouider, M. Chaumont, and W. Puech, “Adaptive steganography by oracle (ASO),” in *Proc. IEEE Int. Conf. Multimedia Expo*, Jul. 2013, pp. 1–6.

[9] J. Fridrich and J. Kodovsky, “Multivariate Gaussian model for designing additive distortion for steganography,” in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, May 2013, pp. 2949–2953.

[10] V. Sedighi, R. Cogranne, and J. Fridrich, “Content-adaptive steganography by minimizing statistical detectability,” *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 2, pp. 221–234, Feb. 2016.

[11] X. Qin, B. Li, and J. Huang, “A new spatial steganographic scheme by modeling image residuals with multivariate Gaussian model,” in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2019, pp. 2617–2621.

[12] T. Pevný and A. D. Ker, “Exploring non-additive distortion in steganography,” in *Proc. 6th ACM Workshop Inf. Hiding Multimedia Secur.*, Jun. 2018, pp. 109–114.

[13] B. Li, M. Wang, X. Li, S. Tan, and J. Huang, “A strategy of clustering modification directions in spatial image steganography,” *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 9, pp. 1905–1917, Sep. 2015.

[14] T. Denemark and J. Fridrich, “Improving steganographic security by synchronizing the selection channel,” in *Proc. 3rd ACM Workshop Inf. Hiding Multimedia Secur. (IH&MMSec)*, Jun. 2015, pp. 5–14.

[15] W. Li, W. Zhang, K. Chen, W. Zhou, and N. Yu, “Defining joint distortion for JPEG steganography,” in *Proc. 6th ACM Workshop Inf. Hiding Multimedia Secur.*, Jun. 2018, pp. 5–16.

[16] J. Fridrich and J. Kodovský, “Rich models for steganalysis of digital images,” *IEEE Trans. Inf. Forensics Security*, vol. 7, no. 3, pp. 868–882, Jun. 2012.

[17] B. Li, Z. Li, S. Zhou, S. Tan, and X. Zhang, “New steganalytic features for spatial image steganography based on derivative filters and threshold LBP operator,” *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 5, pp. 1242–1257, May 2018.

[18] X. Song, F. Liu, C. Yang, X. Luo, and Y. Zhang, “Steganalysis of adaptive JPEG steganography using 2D Gabor filters,” in *Proc. 3rd ACM Workshop Inf. Hiding Multimedia Secur. (IH&MMSec)*, 2015, pp. 15–23.

[19] V. Holub and J. Fridrich, “Low-complexity features for JPEG steganalysis using undecimated DCT,” *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 2, pp. 219–228, Feb. 2015.

[20] J. Kodovský, J. Fridrich, and V. Holub, “Ensemble classifiers for steganalysis of digital media,” *IEEE Trans. Inf. Forensics Security*, vol. 7, no. 2, pp. 432–444, Apr. 2012.

[21] R. Cogranne, V. Sedighi, J. Fridrich, and T. Pevný, “Is ensemble classifier needed for steganalysis in high-dimensional feature spaces?” in *Proc. IEEE Int. Workshop Inf. Forensics Secur.*, Nov. 2015, pp. 1–6.

[22] W. Tang, H. Li, W. Luo, and J. Huang, “Adaptive steganalysis against WOW embedding algorithm,” in *Proc. 2nd ACM Workshop Inf. Hiding Multimedia Secur. (IH&MMSec)*, 2014, pp. 91–96.

[23] T. Denemark, V. Sedighi, V. Holub, R. Cogranne, and J. Fridrich, “Selection-channel-aware rich model for steganalysis of digital images,” in *Proc. IEEE Int. Workshop Inf. Forensics Secur.*, Dec. 2014, pp. 48–53.

[24] W. Tang, H. Li, W. Luo, and J. Huang, “Adaptive steganalysis based on embedding probabilities of pixels,” *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 4, pp. 734–745, Apr. 2016.

[25] S. Zhou, W. Tang, S. Tan, and B. Li, “Content-adaptive steganalysis via augmented utilization of selection-channel information,” in *Proc. Int. Workshop Digital Watermarking*, Oct. 2018, pp. 261–274.

[26] Y. Qian, J. Dong, W. Wang, and T. Tan, “Deep learning for steganalysis via convolutional neural networks,” *Proc. SPIE*, vol. 9409, Mar. 2015, Art. no. 94090J.

[27] G. Xu, H.-Z. Wu, and Y.-Q. Shi, “Structural design of convolutional neural networks for steganalysis,” *IEEE Signal Process. Lett.*, vol. 23, no. 5, pp. 708–712, May 2016.

[28] M. Yedroudj, F. Comby, and M. Chaumont, “Yedroudj-net: An efficient CNN for spatial steganalysis,” in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Apr. 2018, pp. 15–20.

[29] G. Xu, “Deep convolutional neural network to detect J-UNIWARD,” in *Proc. 5th ACM Workshop Inf. Hiding Multimedia Secur.*, Jun. 2017, pp. 67–73.

[30] J. Ye, J. Ni, and Y. Yi, “Deep learning hierarchical representations for image steganalysis,” *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 11, pp. 2545–2557, Nov. 2017.

[31] J. Zeng, S. Tan, B. Li, and J. Huang, "Large-scale JPEG steganalysis using hybrid deep-learning framework," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 5, pp. 1200–1214, May 2017.

[32] M. Boroumand, M. Chen, and J. Fridrich, "Deep residual network for steganalysis of digital images," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 5, pp. 1181–1193, May 2019.

[33] I. J. Goodfellow *et al.*, "Generative adversarial nets," in *Proc. 27th Int. Conf. Neural Inf. Process. Syst.*, Dec. 2014, pp. 2672–2680.

[34] S. Baluja, "Hiding images in plain sight: Deep steganography," in *Proc. 31th Int. Conf. Neural Inf. Process. Syst.*, Dec. 2017, pp. 2069–2079.

[35] D. Volkhonskiy, I. Nazarov, and E. Burnaev, "Steganographic generative adversarial networks," 2017, *arXiv:1703.05502*. [Online]. Available: http://arxiv.org/abs/1703.05502

[36] J. Zhu, R. Kaplan, J. Johnson, and L. Fei-Fei, "Hidden: Hiding data with deep networks," in *Proc. Eur. Conf. Comput. Vis.*, Sep. 2018, pp. 657–672.

[37] W. Tang, S. Tan, B. Li, and J. Huang, "Automatic steganographic distortion learning using a generative adversarial network," *IEEE Signal Process. Lett.*, vol. 24, no. 10, pp. 1547–1551, Oct. 2017.

[38] J. Yang, D. Ruan, J. Huang, X. Kang, and Y.-Q. Shi, "An embedding cost learning framework using GAN," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 839–851, 2020.

[39] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Proc. Med. Image Comput. Comput.-Assisted Intervent*, Oct. 2015.

[40] V. François-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau, "An introduction to deep reinforcement learning," *Found. Trends Mach. Learn.*, vol. 11, nos. 3–4, pp. 219–354, 2018.

[41] D. Silver *et al.*, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, Oct. 2017.

[42] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Proc. 12th Int. Conf. Adv. Neural Inf. Process. Syst.*, Nov. 1999, pp. 1057–1063.

[43] K. Janocha and W. Marian Czarnecki, "On loss functions for deep neural networks in classification," 2017, *arXiv:1702.05659*. [Online]. Available: http://arxiv.org/abs/1702.05659

[44] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *Proc. Int. Conf. Learn. Represent.*, 2015, pp. 1–11.

[45] W. Tang, B. Li, S. Tan, M. Barni, and J. Huang, "CNN-based adversarial embedding for image steganography," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 8, pp. 2074–2087, Aug. 2019.

[46] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. Int. Conf. Artif. Intell. Statist.*, May 2010, pp. 249–256.

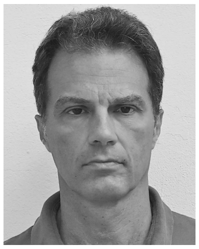
[47] Y. LeCun, L. Bottou, G. Orr, and K. Müller, "Efficient BackProp," in *Neural Networks: Tricks of the Trade* (Lecture Notes in Computer Science), vol. 7700, 2nd ed. Berlin, Germany: Springer, 2012, pp. 9–48.

[48] P. Bas, T. Filler, and T. Pevný, "Break our steganographic system: The ins and outs of organizing BOSS," in *Proc. Int. Workshop Inf. Hiding*, May 2011, pp. 59–70.

Weixuan Tang (Member, IEEE) received the B.E. and Ph.D. degrees from the School of Electronics and Information Technology, Sun Yat-sen University, Guangzhou, China, in 2014 and 2019, respectively. He is currently an Associate Professor with Guangzhou University, Guangzhou. His research interests include digital image steganography, steganalysis, and deep learning.

Bin Li (Senior Member, IEEE) received the B.E. degree in communication engineering and the Ph.D. degree in communication and information system from Sun Yat-sen University, Guangzhou, China, in 2004 and 2009, respectively.

He was a Visiting Scholar with the New Jersey Institute of Technology, Newark, NJ, USA, from 2007 to 2008. He is currently a Professor with Shenzhen University, Shenzhen, China, where he joined in 2009. He is also the Director with the Shenzhen Key Laboratory of Media Security and the Vice Director with the Guangdong Key Lab of Intelligent Information Processing. He has served as the IEEE Information Forensics and Security TC Member since 2019. His current research interests include multimedia forensics, image processing, and deep machine learning.



Mauro Barni (Fellow, IEEE) graduated in electronic engineering from the University of Florence in 1991. He received the Ph.D. degree in informatics and telecommunications in 1995.

During the last two decades, he has been studying the application of image processing techniques to copyright protection and authentication of multimedia and the possibility of processing signals that have been previously encrypted without decrypting them. Lately, he has been working on theoretical and practical aspects of adversarial signal processing. He has authored or coauthored about 300 articles published in international journals and conference proceedings and holds five patents in digital watermarking and image authentication. He has coauthored the book *Watermarking Systems Engineering: Enabling Digital Assets Security* and other Applications (Dekker, Inc., 2004). He has participated in several National and European research projects on diverse topics, including computer vision, multimedia signal processing, remote sensing, digital watermarking, and IPR protection.

Dr. Barni is a member of EURASIP. He was a recipient of the Individual Technical Achievement Award of EURASIP for 2016. He has been the Chairman of the IEEE Information Forensic and Security Technical Committee from 2010 to 2011. He was the Technical Program Chair of ICASSP 2014. He was appointed DL of the IEEE SPS for the years 2013 C2014. He was the Editor-in-Chief of the IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY from 2015 to 2017. He was the Funding Editor of the *EURASIP Journal on Information Security*. He has been serving as an Associate Editor of many journals, including several IEEE TRANSACTIONS.

Jin Li (Senior Member, IEEE) received the B.S. degree in mathematics from Southwest University in 2002 and the M.S. and Ph.D. degrees in information security from Sun Yat-sen University in 2004 and 2007, respectively. He is currently a Professor and the Executive Dean of the Institute of Artificial Intelligence and Blockchain, Guangzhou University, and the Vice Dean of the School of Computer Science and Cyber Engineering, Guangzhou University. He served as a Senior Research Associate with the Korea Advanced Institute of Technology, South Korea, and Illinois Institute of Technology, USA, from 2008 to 2010, respectively. He has published more than 100 articles in international conferences and journals, including the IEEE INFOCOM, the IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY (TIFS), the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS (TPDS), the IEEE TOC (16 articles in IEEE TRANSACTIONS series journals), ESORICS. His research interests include design of secure protocols in cloud computing (secure cloud storage and outsourcing computation) and cryptographic protocols. He also served as a Program Chairs and Committee for many international conferences such as the IEEE CSE 2017, the IEEE EUC 2017, ISICA 2015. He received three National Science Foundation of China (NSFC) Grants, including the NSFC Outstanding Youth Foundation.

Jiwu Huang (Fellow, IEEE) received the B.S. degree from Xidian University, Xi'an, China, in 1982, the M.S. degree from Tsinghua University, Beijing, China, in 1987, and the Ph.D. degree from the Institute of Automation, Chinese Academy of Science, Beijing, in 1998. He is currently a Professor with the College of Information Engineering, Shenzhen University, Shenzhen, China. Before joining Shenzhen University, he has been with the School of Information Science and Technology, Sun Yat-sen University, Guangzhou, China, since 2000. His current research interests include multimedia forensics and security. He serves as a member of the IEEE CASS Multimedia Systems and Applications Technical Committee and the IEEE SPS Information Forensics and Security Technical Committee. He is an Associate Editor of the IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY.