Variational Cover Modification Steganography

Martin Beneš
University of Innsbruck
martin.benes@uibk.ac.at

Rainer Böhme
University of Innsbruck
rainer.boehme@uibk.ac.at

Abstract

Many current approaches to learning-based image steganography are, at best, imperceptible, but their security is vulnerable to state-of-the-art detector networks. Variational cover modification steganography tackles this problem by training a generator network to predict the parameters of the distribution of the least detectable stego-noise for a given cover image. It uses syndrome coding to embed the message within the distribution constraint and retrieve it exactly when a secret key is provided. Adversarial learning ensures that the generator evolves against a continuously refined detector. Experiments demonstrate significant security improvements, and several ablations support the generator architecture and the training protocol.

1 Introduction

Steganography hides secret messages inside casual cover objects, such as digital images, sent over public communication channels. The steganographer (she/her, by convention) is a taker — she must mimic the empirical distribution of the channel by making the stego objects with hidden messages *indistinguishable* from pristine covers. A practical approach is to sample a cover object, and modify it slightly such that only the recipient can extract the desired message using a pre-shared mechanism [1]. The rationale behind this is that digital images contain noise, making small changes indistinguishable.

The communication channel is observed by a passive adversary (he/him), who attempts to detect objects that contain a message. Called steganalysis, this task may involve manual (visual) inspection [2] or automated scanning using a trained classifier [3, 4, 5, 6]. Even slight steganographic modifications can frequently be discerned from noise, as illustrated in Fig. 1. As the security of steganography is defined by how difficult it is for an adversary to classify covers and stego objects, it can be measured using decision error rates in ROC curves and metrics derived from them, such as the area under the ROC curve (ROC AUC).

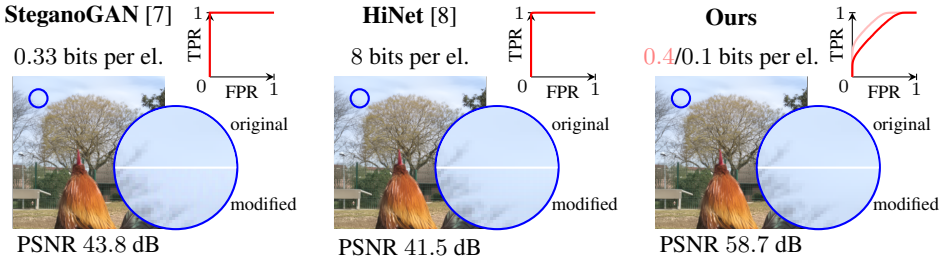


Figure 1: Visual and statistical security of selected steganographic methods. All achieve good visual quality (PSNR $\gtrsim$ 40 dB), but previous methods are perfectly detected by the state-of-the-art steganalysis detector, EfficientNet B0s [6] (ROC curves, top right). SteganoGAN and HiNet embed a fixed number of bits per element. Our method can hide at custom embedding rates.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

Secure steganography should withstand any form of detection. This requirement puts the steganographer in an unenviable position. Her embedding strategy must adapt to a non-trivial channel distribution that is not under her control [9]. The adversary, in turn, can move *after* she has committed to her strategy, and can thus optimize detection to the channel and her method [10]. The steganographer's only advantage over the adversary is a secret bit string, called key $\boldsymbol{k}$, agreed upon in advance with the recipient [11]. The key must not depend on the message or the cover, and is best chosen at random.

Steganography and learning have an intricate relationship. On the one hand, learning can help the steganographer to understand the cover distribution, especially its noise, and allow her to modify sampled objects in the least conspicuous spots. On the other hand, learning-based steganography struggles to achieve the desired property that the exact knowledge of the key should reveal the message completely, while a single bit flip should remove all traces of its existence. This is reminiscent of Shannon's notion of confusion in cryptography [12, p. 709]. Intuitively, any applicable loss function must be a non-differentiable impulse, since any indication of a gradient could allow the adversary to train a surrogate model and gain sufficient knowledge of the key [13]. Many learning-based methods abandon the idea of a key [14, 15, 16, 7, 8, 17], and thus compromise on security.

Variational cover modification steganography, the main contribution of this work, solves this dilemma by restricting the learning part to understand *where* the cover can be modified, and delegating the actual embedding to coding techniques established in the signal processing community. Specifically, we use adversarial training to evolve a generator that *predicts the joint distribution* of secure modifications. Unlike in typical GANs [18], the discriminator continuously improves its capability to *do steganalysis*. Following this approach, we obtain a learning-based steganographic method for color images that can transmit significantly longer messages than many learning-based methods in the literature, with zero bit errors. At the same time, it is significantly more secure than *all* the learning-based methods in our benchmark, and outperforms signal processing-based steganography for all but the shortest messages.

Roadmap Section 2 reviews existing learning-based approaches to steganography, positioning variational cover modification within this design space. Section 3 describes our experimental setup, mentions potential limitations, presents the main results, and documents our ablations. Section 4 discusses the insights gained and reflects on the potential impact of the design. Section 5 concludes. Our code is available at `https://github.com/uibk-uncover/vcms`.

2 Machine learning in steganography

In this section, we introduce notation and terminology (Sec. 2.1), then we summarize previous approaches that apply machine learning to steganography (Sec. 2.2), and finally we describe our variational approach (Sec. 2.3).

2.1 Notation and terminology

We denote vectors and tensors in boldface and their elements in plain, $\boldsymbol{x} = (x_1, \dots, x_n)$, with the length, $n = |\boldsymbol{x}|$, often implicit. To annotate whether an object, $\boldsymbol{x}$, contains a steganographic message, we write $\boldsymbol{x}^{(0)}$ for covers, $\boldsymbol{x}^{(1)}$ for stego objects with any message embedded, and $\boldsymbol{x}^{(\boldsymbol{m})}$ for stego objects with the specific message $\boldsymbol{m}$. Messages are assumed to have zero redundancy, meaning that the size of the message, $|\boldsymbol{m}|$, is a precise metric for the capacity. We shall use $h(\boldsymbol{p})$ for the binary entropy (in bits) of a categorical distribution with parameter vector $\boldsymbol{p}$. To relate the message size to the cover dimension, we define the embedding rate $\alpha = {|\boldsymbol{m}|}/{|\boldsymbol{x}|}$ in bits per element (bpe). This work uses RGB color images, which have three elements per pixel.

Regarding terminology, steganography is related to, but distinct from digital watermarking [19, Sec. 1]. Both techniques embed information into media objects. The primary goal of robust watermarking is making the mark hard to remove without impairing the utility of the object. The mark does not need to be undetectable. The primary goal of secure steganography is undetectability in a given channel.

2.2 Existing approaches

The literature documents numerous proposals to use machine learning for steganography. In Table 1, we organize them into three approaches: generative steganography, end-to-end methods, and evasion attacks. We contrast them with variational methods (cf. Sec. 2.3) and signal processing methods that

Table 1: Different approaches to using learning for steganography.

Approach	Setup phase	Embedding phase
Generative steganography	Training of image generator	Image generation from latent message
End-to-end methods	Adversarial training of detector and message encoder ("hiding network")	Forward pass
Evasion attacks	Training of detector against existing methods	Optimization against fixed detector, (coding)
Our approach:		
Variational methods	Adversarial training of detector and stego-noise distribution generator	Forward pass, coding
For comparison:		
Without learning	Manual (heuristic)	Signal processing, coding

do not involve learning. The differences lie in the setup, i. e., before secret communication takes place, and in how messages are embedded. The result of the setup phase is shared knowledge of all potential communication partners, akin to a public standard. This must be distinguished from the key exchange, which also takes place before communication and depends on a confidential channel between the sender and recipient. The embedding phase produces the stego object once the setup is complete and the message and concrete cover are known. The setup is reused for many exchanges.

Generative methods Unlike in cover modification steganography, generative steganography synthesizes a stego object containing the desired message without requiring a cover. A common pattern is to encode the message into the latent, and decode using an invertible diffusion process to ensure that it can be recovered [20, 21, 22, 23]. Another option is to condition the diffusion on an external constraint, such as a (secret) extraction network [24, 25]. With access to the image generator, generative methods can be secure in channels where synthetic images from the exact same model are plausible [26, 27]. However, such channels may not exist. Current methods fall short of mimicking natural images. Their outputs inevitably contain traces of image synthesis [28]. The same limitations also apply to recent works that propose to guide the diffusion process by a cover image [29, 30], which blends over to cover modification.

End-to-end methods End-to-end methods embed by cover modification using a "hiding" network [15, 14, 16, 7, 31, 32]. As illustrated in Figure 2a, the network takes a cover image, $\boldsymbol{x}^{(0)}$, a message, $\boldsymbol{m}$, an optional key, $\boldsymbol{k}$, and produces a stego image, $\boldsymbol{x}^{(\boldsymbol{m})}$. The message is extracted from $\boldsymbol{x}^{(\boldsymbol{m})}$ by a "revealing" network. Systems can be trained adversarially against a discriminator, which should make them undetectable.

The training loss of end-to-end methods typically combines three objectives: an *adversarial* term maximizes the detection error, either by fooling an actual discriminator, e. g., cross-entropy [14], the Wasserstein loss [7], or through discriminator-free regularization [8]; a *perceptual* term reduces the embedding impact, e. g., L2 norm [14], LPIPS [31], or SSIM [32] between $\boldsymbol{x}^{(0)}$ and $\boldsymbol{x}^{(1)}$; finally, a *decoding* term reduces bit errors in the extracted message, but may not fully avoid them.

Evasion attacks For steganography by evasion, the setup involves training a strong detector against existing steganographic methods. The message is embedded by an optimization that searches for changes which encode the message and fool this fixed detector [33]. The process resembles the search for adversarial examples [34] with additional constraints from the message. Coding has been proposed to efficiently meet this constraint [35]. While promising, a shortcoming of this approach is the high computational effort required *for each message*. This approach has not yet been successfully demonstrated for images in the spatial (pixel) domain, let alone color images.

2.3 Variational methods

Our work embarks on a different way of using machine learning for cover modification steganography. Variational methods learn to predict the distribution of undetectable stego-noise for a given realization, $\boldsymbol{x}^{(0)}$, of the cover source. Stego-noise, $\boldsymbol{d}$, is defined as the difference between stego and cover objects. Its distribution can be interpreted as a map indicating *where* it is more secure to modify the cover. As the number of modifications required to embed a message depends on its size, one should not predict the change probabilities, $\boldsymbol{p}$, directly. Rather, the generator, G, predicts a cost map, $\boldsymbol{\varrho}$, that assigns higher costs to elements that are less secure to modify (Figure 2b+c).

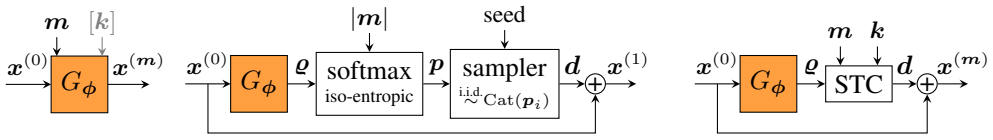
This approach becomes feasible because syndrome-trellis codes (STC) exist [36]. They are special kinds of covering codes that embed a message by modifying symbols inversely proportional to a cost vector, while allowing the recipient to extract the message without the knowledge of where the embedding took place. The latter principle was first introduced by Anderson [11], and STCs can be considered a generalization of this principle. Mathematically, the sender finds a solution to an underspecified system of equations that minimizes the sum of the costs associated with each symbol transition. This solution must satisfy the constraint that the recipient can recover the message by multiplying the received sequence with a key-dependent parity-check matrix using finite field arithmetics (see Appendix A.2 for a technical briefing). Algorithmically, the sender uses a variant of the Viterbi algorithm [37]. This makes STCs efficient, and the realizations of $\boldsymbol{d}$ closely follow the stego-noise distribution induced by the cost vector [38].

To learn where stego-noise is least detectable, the generator, G, is trained adversarially, using feedback from a discriminator, D, that performs steganalysis. To enable gradient flow from D to G despite the discrete nature of STCs, we replace the coding with a differentiable simulator, as shown in Figure 2b. The iso-entropic softmax approximates the coding by constraining the binary entropy of $\boldsymbol{p}$ to match the desired message size $|\boldsymbol{m}|$ (in bits),

$$h(\boldsymbol{p}) = h(\mathrm{softmax}(-\lambda\boldsymbol{\varrho})) = |\boldsymbol{m}|. \tag{1}$$

Observe that the costs $\boldsymbol{\varrho}$ are inverse logits of $\boldsymbol{p}$, i. e., $\boldsymbol{p} = \mathrm{softmax}(-\lambda\boldsymbol{\varrho})$, where the scalar λ^{-1} is the softmax temperature. The *entropy constraint* is expressed as an implicit function of λ in Eq. 1, hence, the solution must be found numerically. Finally, the stego-noise is sampled as $\boldsymbol{d} \sim \mathrm{Categorical}(\boldsymbol{p})$.

This approach is inspired by signal processing methods that perform steganography without learning [39], and hence without having to worry about differentiability [35]. Based on the observation that detectors err in textured or noisy areas, these methods hand-craft cost functions using linear high-pass filters [40, 41] or wavelets [42, 43]. However, there is no mechanism in place to ensure that the heuristics capture the security-related properties of the channel. Moreover, and perhaps consequently, all of these methods simplify the task to grayscale channels, which are uncommon in practice [44]. The feedback loop that is built into adversarial training was slow and involved humans (researchers) in the 2010s.



(a) **End-to-end.** (b) **Variational methods** during training. (c) **Variational methods** during inference.

Figure 2: Role of the generator in cover modification steganography. Our variational method generates the stego-noise distribution indirectly using predicted costs and message size $|\boldsymbol{m}|$. During training (b), the noise is simulated according to the probabilities $\boldsymbol{p}$. During inference (c), STC codes produce flip vectors very close to the distribution. End-to-end methods (a) try to combine this in one network.

3 Experiments

3.1 Setup

Data We use $80\,005$ cover images from the ALASKA dataset [44], the de-facto standard benchmark for color image steganography, and crop them to the dimension 3×296^2. The images are randomly assigned to the training, validation, and test sets in a ratio of $70{:}15{:}15$.

Steganography We compare our method with selected end-to-end learning-based steganographic methods (using their public implementations) that have been published in the last ten years at the leading venues in machine learning and computer vision: HiDDeN [16], AdvSteg [14], DeepSteg [15], SteganoGAN [7], HiNet [8], CRoSS [45], and DiffStega [29]. The methods were trained on our dataset, using each method's training protocol, to minimize out-of-distribution effects. For the CRoSS and DiffStega methods (with $> 10^9$ parameters), we use the pretrained weights. From the signal processing methods, we compare against common benchmarks, HILL [40] and WOW [42], and extend them to color using the CCM strategy by Cogranne et al. [46].

As steganography is easier to detect in larger covers, we harmonize the cover dimensions for a fair comparison. This was possible for all methods except AdvSteg [14], whose dense layers cause the number of parameters to grow with the input dimension. While we acknowledge this as an advantage for AdvSteg, the results give a very clear message anyway.

Adversarial system The adversarial system is at the core of variational cover modification steganography. We adopt the generator, discriminator, and training protocol of [47], which presented this method to the security community (see Appendix A.5 for the relation between the two papers). Our generator network, G, uses a single depthwise separable block with 7×7 kernels, 192 latent channels, and leaky ReLU activation between the depthwise and the pointwise convolutions. The discriminator, D, is based on UcNet [48], a specialized detector for color image steganography. The generator trains with a non-saturating adversarial loss, (using the notation introduced in Section 2.1)

$$\mathcal{L}_G = D(\boldsymbol{x}^{(0)} + \boldsymbol{d}) + \mathrm{ReLU}\left(\frac{\text{\# elements modified}}{\text{\# pixels}} - c\right). \tag{2}$$

The stego-noise $\boldsymbol{d}$ is drawn from the categorical distribution, $\boldsymbol{d} \sim \mathrm{Categorical}(\boldsymbol{p})$, where $\boldsymbol{p} = \mathrm{softmax}(-\lambda \cdot G(\boldsymbol{x}^{(0)}))$. The latter term is a soft constraint that leans the generator towards more conservative behavior of changing fewer elements. As long as the change rate remains within the budget $[0, c]$, the generator only minimizes detectability. As the change rate is bounded to $[0, 3]$ in RGB images, we set $c = 0.5$.

The discriminator trains with a hinge loss using margin ± 1,

$$\mathcal{L}_D = \mathrm{ReLU}(1 + D(\boldsymbol{x}^{(0)})) + \mathrm{ReLU}(1 - D(\boldsymbol{x}^{(0)} + \boldsymbol{d})). \tag{3}$$

This training setup allows the generator to keep learning even if the discriminator becomes confident. Later, in Table 4, we support our design by an ablation study that evaluates selected generator and discriminator architectural choices, and the change rate budget term of Equation 2.

To ensure that our setup (in Fig. 2b) is differentiable, we adopt tricks from evasion attacks [35]. This means calculating the gradient of the iso-entropic softmax manually by differentiating the (implicit) constraint using the implicit function theorem, $\frac{\partial \lambda}{\partial \varrho_i} = -\frac{\partial h}{\partial \varrho_i}\left(\frac{\partial h}{\partial \lambda}\right)^{-1}$. We use bracket search combined with binary search to solve for λ in the forward pass. Our sampler from the categorical distribution uses a discretized Gumbel-softmax combined with a straight-through estimator. See Appendix A.6 for comparisons to other approaches.

For more details on our model, see Appendix A.1. To facilitate reproducibility, we also provide the source code, trained weights, dataset splits, and decision vectors.

Detectors We benchmark against three state-of-the-art deep-learning detectors: B0s, SRNet, and UcNet. B0s is a modified variant of EfficientNet B0 [49] using known tricks to boost its performance for steganalysis: removing the stem layer stride [6], and pretraining on an ImageNet classification task [50]. SRNet [5] is an established architecture for steganalysis, adapted to color by increasing the number of input channels. UcNet [48] is a specific architecture for color steganalysis, which is also used as the discriminator.

Training the adversarial system The adversarial system is trained for 15 epochs at a fixed learning rate 10^{-4}, using the RMSprop optimizer, and a batch size of 32. To ensure invariance across message lengths, we draw the embedding rate for each cover randomly from $\alpha \sim \mathrm{Uniform}(0.1, 1)$. The training uses a 1:5 update ratio, i. e., five update steps of the discriminator are followed by one update step of the generator. At every update step, we apply a mild gradient norm constraint (norm 1). Once trained, we use the generator network to simulate the stego images for the covers.

Training the detectors The steganalysis detectors are trained on our dataset for each combination of steganographic method (e. g., the exact generator weights) and embedding rate to ensure that the security evaluation is as conservative as possible. For their training, we use the Adam optimizer with cross-entropy loss and a one-cycle policy at a peak learning rate 10^{-3} for 30 training epochs. Detectability is measured by the area under the ROC curve (ROC AUC). As the classes are balanced, ROC AUC ranges between 0.5 (random guess; worst) and 1.0 (perfect detection; best).

Computational resources The adversarial training was conducted on an internal workstation, equipped with 90GB GPU and 1000GB of RAM, requiring approximately five hours per run. We used the same system to train the end-to-end methods that we are comparing to, and the runtimes varied from three to 24 hours. Steganographic embedding—performed for approximately one hour per method and message size—yields 80 005 simulated stego images that occupy around 1.5GB of disk space. The detector training was distributed across two shared clusters of nodes equipped with consumer-grade GPUs (30GB VRAM, 16GB RAM), which took an average of eight hours per model. This implies that, for example, the results presented in Table 2 represent a cumulative computational cost of 23 GPU days and 300 GB of storage.

3.2 Limitations

Three factors may impair the generalizability of the experimental results below. First, the main results rely only on the ALASKA dataset, the de-facto standard in color image steganalysis, which contains images from many different cameras and randomized, well-documented processing steps. We confirm the improvement over signal processing-based methods on RAISE, a popular image-forensic dataset, in Appendix A.7. However, neither may be fully representative of real-world images, which undergo a plethora of processing pipelines, including heavy compression and recompression. Second, the main results use an embedding simulator, which draws realizations from the ideal distribution and may thus overestimate the practical security of the method. We confirm that embedding with actual STCs increases the detectability only slightly (cf. Appendix A.3). Third, the signal processing-based methods have been designed for grayscale images and had to be extended to color using one of several strategies [46]. It is unknown to what extent this strategy influences the detectors in our study. We argue that none of these limitations poses a substantial threat to the validity of the main point behind variational cover modification steganography or its performance compared to end-to-end methods.

Table 2: (In)security of learning-based cover modification steganography against state-of-the-art detectors. The end-to-end approaches embed very short messages or bitmaps of matching size. Our method is harder to detect (AUC $\lll 1$) and, thanks to coding, can adjust the message size flexibly.

ROC AUC (↓)	Steganographic method and embedding rate [bits per element]												
	End-to-end									*Ours*			
	HiDDeN	AdvSteg	DeepSteg	SteganoGAN			HiNet	CRoSS	DiffStega				
Detector	10^{-4}	10^{-3}	8.0	1	$\frac{2}{3}$	$\frac{1}{3}$	8.0	$< 10^{-3}$	8.0	.4	.2	.1	.05
B0s	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	.868	.850	.804	.769
(std. error)										(.015)	(.022)	(.044)	(.046)
SRNet	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	.869	.842	.797	.771
(std. error)										(.006)	(.036)	(.056)	(.059)
UcNet	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	.918	.864	.810	.766
(std. error)										(.013)	(.019)	(.025)	(.032)

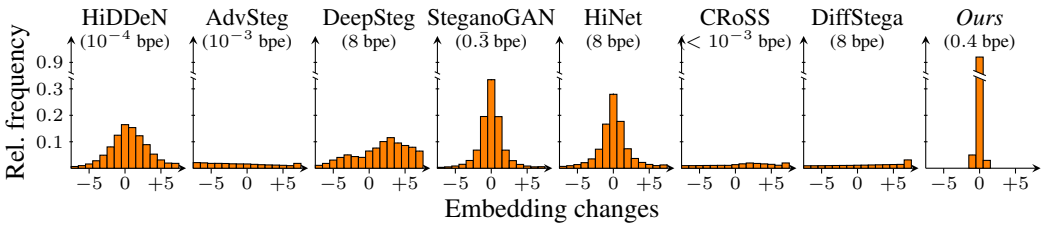


Figure 3: Distributions of the embedding changes made by the steganographic methods. The error bars are barely visible. They denote ± 1 standard error. Embedding rates are in parentheses.

3.3 Results

We compare the security and the shape of the stego-noise of our method to seven end-to-end methods, before we also compare to two signal processing-based methods.

Table 2 presents the comparison to the learning-based approaches. Our variational method is the only one that resists detection to some extent, while all end-to-end methods are perfectly detectable, irrespective of the message size. This is consistent with the evidence in [8, Fig. 5] and [51, Tab. 2]. To put the numbers into perspective, a ROC AUC of 0.80 approximately translates to 40% correct detection at a fixed false positive rate of 5%. The table also shows that the detector architecture is of minor impact, as the ROC AUC estimates mostly lie within one standard error. Long messages are the exception. They are best detected by UcNet, the newest architecture and the one specifically designed for color steganalysis. The standard errors are calculated over the performance of three distinct detector trainings that start from differently initialized weights, against three different generators extracted at checkpoints that are 1000 training steps apart. We use green color to annotate standard errors calculated this way. We refrain from calculating standard errors for the end-to-end methods. As the initial detectors achieved perfect separation in all cases, we deemed the additional ≈ 16 GPU days required to estimate (likely negligible) variances to be a waste of resources.

To understand the differences between the approaches, it is instructive to explore the marginal distribution of the stego-noise, depicted in Figure 3. The fact that our method limits changes to ± 1 likely contributes to its improved security. The impact of learning will become evident when we compare to the signal processing methods, which restrict the stego-noise to the same domain.

Table 3 compares our variational approach to methods based on signal processing. Significant improvement is evident for longer messages, but not for very short ones. This is because almost every image has some cheap spots that can be found regardless of learning. Learning becomes relevant for longer messages when the obvious spots are taken. This is illustrated in Figure 4. The standard errors in Table 3 span three different generator checkpoints for our method. For the signal processing-based methods, the three detectors are trained against a fixed method. We indicate this difference by printing the standard errors for a fixed method in gray.

Table 3: Security of learning-based costs compared to common signal processing-based benchmarks, using state-of-the-art detectors trained specifically against every method.

ROC AUC (↓)	Steganographic method and embedding rate [bits per element]											
	Signal processing-based								**Learnt**			
	HILL				WOW				*Ours*			
Detector	.4	.2	.1	.05	.4	.2	.1	.05	.4	.2	.1	.05
B0s	.944	.890	.821	.746	.948	.905	.834	.731	.868	.850	.804	.769
(std. error)	(.004)	(.020)	(.011)	(.011)	(.007)	(.002)	(.010)	(.009)	(.015)	(.022)	(.044)	(.046)
SRNet	.943	.899	.821	.716	.954	.899	.824	.731	.869	.842	.797	.771
(std. error)	(.007)	(.004)	(.007)	(.023)	(.002)	(.010)	(.005)	(.011)	(.006)	(.036)	(.056)	(.059)
UcNet	.951	.907	.840	.741	.949	.897	.817	.721	.918	.864	.810	.766
(std. error)	(.001)	(.002)	(.002)	(.003)	(.002)	(.003)	(.004)	(.006)	(.013)	(.019)	(.025)	(.032)

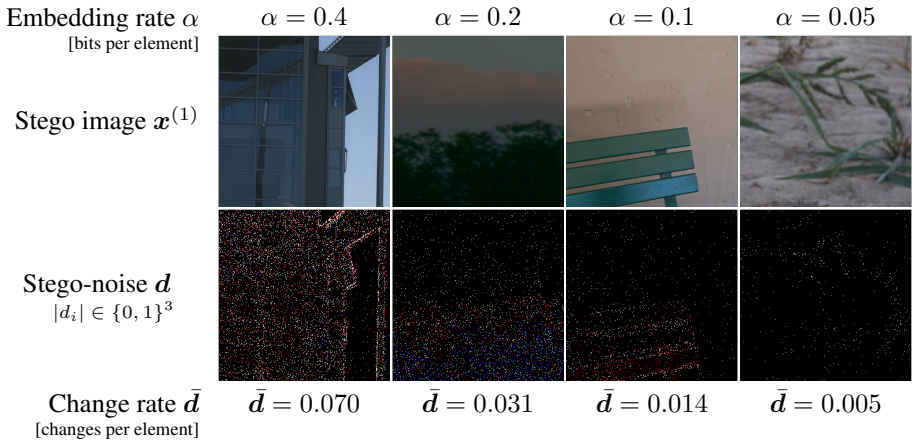


Figure 4: Sample stego images of our method, with the corresponding stego-noise realizations $\boldsymbol{d}$.

3.4 Ablation

We carry out ablation on three design elements related to the generator and discriminator networks as well as the training protocol. Table 4 shows the results. Regarding the kernel size of the generator, 7×7 performs much better than 5×5. Interestingly, it also outperforms 9×9. When we replace the UcNet with the much larger B0s as discriminator, we observe a substantial decrease of security. It can be explained in different ways, e. g., too large difference in the capacities of G and D, or suboptimal architectural components, such as the batch normalization in B0s. We leave further optimization of the design components to future work. Finally, removing the change rate budget from the loss function in Eq. 2 also reduces security. Without this regularizer, the generator becomes reckless and changes most of the pixels to push the sample towards the detection boundary. By doing so, it is easily caught by a freshly trained detector. Amongst the tested configurations, our baseline setup turns out to be the best.

We are aware that oscillating performance during the adversarial training could potentially distort the conclusion of the ablation. To show that this is not the case, we report the performance over a grid of checkpoints in Appendix A.4.

4 Discussion

We started out by observing that existing end-to-end learning-based steganography is insecure, and developed a method that combines the benefits of learning with established coding techniques. Table 5

Table 4: Results of the ablation study: 7×7 is a sweet spot, UcNet is a good choice for the discriminator, and without the change rate term in the loss function, the network underperforms.

ROC AUC (↓)	**EfficientNet B0s**			**# parameters**	
Condition Emb. rate (bpe):	0.4	0.2	0.1	G	D
Baseline as presented	.868	.850	.804	14.6k	675k
(std. error)	(.015)	(.022)	(.044)		
G: 9×9 convolution	.932	.851	.826	20.7k	
(std. error)	(.012)	(.031)	(.047)		
G: 5×5 convolution	.955	.947	.949	10k	
(std. error)	(.015)	(.024)	(.027)		
D: EfficientNet B0s	.987	.978	.967		4M
(std. error)	(.005)	(.002)	(.003)		
Training: no change rate budget	.977	.980	.969		
(std. error)	(.005)	(.004)	(.008)		

Table 5: Properties of existing end-to-end methods compared to our variational method.

Reference	Cover dim.		Secret message	Parameters	Bit error rate	Key	Code avail.
	Ref.	Here					
HiDDeN [16]	512^2	296^2	fixed to 30 bits	488k	$2.0 \cdot 10^{-4}$	no	yes‡
AdvSteg [14]	108^2	32^{2*}	fixed to 10 bits	6.4M	$1.3 \cdot 10^{-4}$	no	yes
DeepSteg [15]	64^2	296^2	RGB image†	2.6M	.494	no	no‡
SteganoGAN [7]	?	296^2	bitmap†	83k	.020	no	yes
HiNet [8]	256^2	296^2	RGB image†	4M	.183	no	yes
CRoSS [45]	512^2	296^2	secret prompt	1.1G	?	no!	yes
DiffStega [29]	512^2	296^2	RGB image†	1.1G	.361	no!	yes
Ours		296^2	any up to 1.5 bpe	15k	0.0	yes	yes

†: of cover dimension *: changing it requires retraining ?: not explicitly stated in the paper ‡ third-party implementation
!: The paper's notion of "key" is designed to protect the content of the message, but not its existence.

shows that our variational method occupies a favorable position of the design space. Among the methods considered, it is the only one that supports a secret key to hide the existence of the message. Unlike all others, it extracts the exact message without incurring bit errors. (For a fair evaluation, the message size would have to account for the overhead of error correcting codes [7, 31].) The adjustable message size means that the steganographer can select a point on the capacity–security tradeoff. Our method can also adapt to channel conventions, as it accepts covers of any dimension. Finally, its generator has the fewest parameters by a margin. This is presumably because the variational approach only has to learn the least detectable stego-noise, whereas the end-to-end methods require capacity to model the source as well as the embedding operation.

The bit error rate in Table 5 applies to lossless transmission. If the image is processed in transit, for example by recompression on social media platforms, coding-based extraction will most likely fail. Some of the other methods may recover part of the message from its redundancy, e. g., when an image is embedded in an image [15, 8]. But perceptual notions of robustness are fuzzy and hard to evaluate. A principled approach would begin with a precise channel model and select tailored error-correcting codes, which would reduce the effective message size. For example, robustness to cropping would require embedding shorter messages in tiles of the image; robustness against downscaling would mean embedding into low frequencies of a transformed image, and so on. We deem these problems orthogonal to learning how to make changes that are undetectable.

Although the idea behind variational cover modification is intuitive, it only works if a number of factors align. Replacing STC with a differentiable simulator is the first step. Then it is tempting to predict the probabilities, $\boldsymbol{p}$, directly, which has been pursued, for grayscale images, and with varying success [52, 53, 54, 55, 56, 57]. The main problem is that predicting $\boldsymbol{p}$ couples the learning of the source with the global entropy constraint. Only networks with a global scalar bottleneck can reasonably achieve this. But they are unnecessarily complex for adversarial training, and do not support arbitrary cover dimensions [58]. Our idea of predicting costs, $\boldsymbol{\varrho}$, in color images with a simple, fully convolutional generator, unlocks steganography by variational cover modification.

Turning to future work, the variational approach itself could unlock innovation in steganography. The problem of color image steganography is not fully solved. Complete undetectability in any cover, i. e., ROC AUC at 0.5, is still out of reach (cf. Table 2). What constrains our method is its implied distribution model for the stego-noise. While we do model color channels jointly, we must assume that modifications to cover elements are independent. This constraint is given by the coding, which is only efficient under additive costs (see Appendix A.2). Fortunately, innovation does not need to wait for better coding. The variational approach allows us to experiment with different classes of probability distributions that imply non-additive costs. As long as efficient and differentiable simulators exist, generators can be trained and benchmarked. Once a good distribution model for hard-to-detect stego-noise is found, one can turn to the coding layer and search for efficient constructions, or, for the sake of security, accept less efficient search.

The ideas presented here can also serve as blueprints for steganography in images compressed with JPEG or processed by social media platforms, as well as in other modalities, such as compressed

video, which appears more challenging because it nests elements of different type [59]. Applying variational cover modification to AI-generated sources is another possible avenue. It may avoid the need for invertible diffusion in generative steganography (cf. Sec. 2.2).

Finally, secure steganography has broader societal implications by empowering individuals against systematic oppression. Steganography lays the foundation for unobservable communications, censorship-resistant protocols, deniable encryption, and many privacy-enhancing technologies. Attempts to restrict the use of cryptography by law become unenforceable if secure steganography exists. It is in the interest of the community to avoid a false sense of security that may arise from inadequate benchmarks, like reducing security to imperceptibility, in highly visible research papers.

5 Conclusion

Steganography must be secure and practical. Security means undetectability by any choice of detector. Practical means that the method can adapt to a channel that is not under the steganographer's control.

Variational cover modification steganography tackles this by combining learning with coding in a novel way. While previous learning-based methods have focused on modeling the image, our approach is to model the least detectable stego-noise for a given image. Evaluations show that our adversarially learned method is much more secure than all end-to-end alternatives, and less detectable than signal processing-based methods for all but very short messages.

Acknowledgments and disclosure of funding

We thank Tomáš Pevný for his help with the differentiable simulator and during the rebuttal. Benedikt Lorch and Chiara Musso have provided valuable feedback. This research was funded in part by the Austrian Science Fund (FWF) 10.55776/PIN2146324 (DETERMINE).

References

[1] Gustavus Simmons. "The prisoners' problem and the subliminal channel". In: *Advances in Cryptology (CRYPTO)*. Ed. by David Chaum. Plenum Press, New York, 1984, pp. 51–67.

[2] Andreas Westfeld and Andreas Pfitzmann. "Attacks on steganographic systems". In: *Information Hiding (IH)*. Ed. by Andreas Pfitzmann. LNCS. Springer, 1999, pp. 61–76.

[3] Jessica Fridrich and Jan Kodovský. "Rich models for steganalysis of digital images". In: *IEEE Transactions on Information Forensics and Security (TIFS)* 7.3 (2012), pp. 868–882.

[4] Jan Kodovský, Jessica Fridrich, and Vojtěch Holub. "Ensemble classifiers for steganalysis of digital media". In: *IEEE Transactions on Information Forensics and Security (TIFS)* 7.2 (2012), pp. 432–444.

[5] Mehdi Boroumand, Mo Chen, and Jessica Fridrich. "Deep residual network for steganalysis of digital images". In: *IEEE Transactions on Information Forensics and Security (TIFS)* 14.5 (2019), pp. 1181–1193.

[6] Yassine Yousfi, Jan Butora, Jessica Fridrich, and Clément Fuji Tsang. "Improving EfficientNet for JPEG steganalysis". In: *Workshop on Information Hiding and Multimedia Security (IH&MMSec)*. Ed. by Dirk Borghys, Patrick Bas, Luisa Verdoliva, Tomáš Pevný, Bin Li, and Jennifer Newman. ACM, 2021, pp. 149–157.

[7] Kevin Zhang, Alfredo Cuesta-Infante, Lei Xu, and Kalyan Veeramachaneni. "SteganoGAN: High capacity image steganography with GANs". In: *CoRR* abs/1901.03892 (2019). arXiv: 1901.03892.

[8] Junpeng Jing, Xin Deng, Mai Xu, Jianyi Wang, and Zhenyu Guan. "HiNet: Deep image hiding by invertible network". In: *International Conference on Computer Vision (ICCV)*. IEEE, 2021, pp. 4733–4742.

[9] Rainer Böhme. "An epistemological approach to steganography". In: *Information Hiding (IH)*. Ed. by Stefan Katzenbeisser and Ahmad-Reza Sadeghi. LNCS. Springer, 2009, pp. 15–30.

[10] Pascal Schöttle and Rainer Böhme. "Game theory and adaptive steganography". In: *IEEE Transactions on Information Forensics and Security (TIFS)* 11.4 (2016), pp. 760–773.

[11] Ross Anderson. “Stretching the limits of steganography”. In: *Information Hiding (IH)*. Ed. by Ross Anderson. LNCS. Springer, 1996, pp. 39–48.

[12] Claude Shannon. “Communication theory of secrecy systems”. In: *The Bell System Technical Journal* 28.4 (1949), pp. 656–715.

[13] Tobias Kupek, Cecilia Pasquini, and Rainer Böhme. “On the difficulty of hiding keys in neural networks”. In: *Workshop on Information Hiding and Multimedia Security (IH&MMSec)*. Ed. by Christian Riess, Franziska Schirrmacher, Irene Amerini, Paolo Bestagini, and Tomáš Pevný. ACM, 2020, pp. 73–78.

[14] Jamie Hayes and George Danezis. “Generating steganographic images via adversarial training”. In: *Advances in Neural Information Processing Systems (NIPS)*. Ed. by Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna Wallach, Rob Fergus, S. Vishwanathan, and Roman Garnett. Vol. 30. 2017, pp. 1954–1963.

[15] Shumeet Baluja. “Hiding images in plain sight: Deep steganography”. In: *Advances in Neural Information Processing Systems (NIPS)*. Ed. by Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna Wallach, Rob Fergus, S. Vishwanathan, and Roman Garnett. Vol. 30. 2017, pp. 2069–2079.

[16] Jiren Zhu, Russell Kaplan, Justin Johnson, and Li Fei-Fei. “HiDDeN: Hiding data with deep networks”. In: *European Conference on Computer Vision (ECCV)*. Ed. by Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss. LNCS. Springer, 2018, pp. 657–672.

[17] Shao-Ping Lu, Rong Wang, Tao Zhong, and Paul Rosin. “Large-capacity image steganography based on invertible neural networks”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. CVF/IEEE, 2021, pp. 10816–10825.

[18] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. “Generative adversarial networks”. In: *Communications of the ACM* 63.11 (2020), pp. 139–144.

[19] Ingemar Cox, Matthew Miller, Jeffrey Bloom, Jessica Fridrich, and Ton Kalker. *Digital watermarking and steganography*. Elsevier, 2008. ISBN: 978-0-12-372585-1.

[20] Ping Wei, Ge Luo, Qi Song, Xinpeng Zhang, Zhenxing Qian, and Sheng Li. “Generative steganographic flow”. In: *International Conference on Multimedia and Expo (ICME)*. IEEE, 2022, pp. 1–6.

[21] Qing Zhou, Ping Wei, Zhenxing Qian, Xinpeng Zhang, and Sheng Li. “Improved generative steganography based on diffusion model”. In: *IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)* 35.7 (2025), pp. 6494–6507.

[22] Zihao Xu, Dawei Xu, Zihan Li, and Chuan Zhang. “MDDM: Practical message-driven generative image steganography based on diffusion models”. In: *International Conference on Machine Learning (ICML)*. Ed. by Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu. PMLR 267. OpenReview.net, 2025, pp. 69832–69848.

[23] Aurélien Noirault, Tomáš Pevný, Jan Butora, Vincent Itier, and Patrick Bas. “Better inversion of diffusion models for generative steganography”. In: *Workshop on Information Hiding and Multimedia Security (IH&MMSec)*. Ed. by Alessandro Piva, Mauro Barni, Benedetta Tondi, Dasara Shullani, and Daniele Baracchi. ACM, 2026, pp. 127–137.

[24] Xiyao Liu, Ziping Ma, Junxing Ma, Jian Zhang, Gerald Schaefer, and Hui Fang. “Image disentanglement autoencoder for steganography without embedding”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE/CVF, 2022, pp. 2293–2302.

[25] Wenkang Su, Jiangqun Ni, and Yiyan Sun. “StegaStyleGAN: Towards generic and practical generative image steganography”. In: *AAAI Conference on Artificial Intelligence (AAAI)*. Ed. by Michael Wooldridge, Jennifer Dy, and Sriraam Natarajan. Vol. 38. 1. AAAI Press, 2024, pp. 240–248.

[26] Chengsheng Yuan, Zhaonan Ji, Qi Cui, Zhili Zhou, Xinting Li, and Zhihua Xia. “Towards provably secure and highly robust generative image steganography leveraging latent diffusion model”. In: *AAAI Conference on Artificial Intelligence (AAAI)*. Ed. by Sven Koenig, Chad Jenkins, and Matthew Taylor. AAAI Press, 2026, pp. 1498–1506.

[27] Etienne Levecque, Aurélien Noirault, Tomáš Pevný, Jan Butora, Patrick Bas, and Rémi Cogranne. “Targeted pooled latent-space steganalysis applied to generative steganography, with a fix”. In: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2026, pp. 13452–13456.

[28] Francesco Marra, Diego Gragnaniello, Luisa Verdoliva, and Giovanni Poggi. "Do GANs leave artificial fingerprints?" In: *Conference on Multimedia Information Processing and Retrieval (MIPR)*. IEEE, 2019, pp. 506–511.

[29] Yiwei Yang, Zheyuan Liu, Jun Jia, Zhongpai Gao, Yunhao Li, Wei Sun, Xiaohong Liu, and Guangtao Zhai. "DiffStega: Towards universal training-free coverless image steganography with diffusion models". In: *International Joint Conference on Artificial Intelligence (IJCAI)*. ijcai.org, 2024, pp. 1579–1587.

[30] Jiannian Wang, Yao Lu, and Guangming Lu. "SSHR: More secure generative steganography with high-quality revealed secret images". In: *International Conference on Machine Learning (ICML)*. Ed. by Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu. PMLR 2533. OpenReview.net, 2025, pp. 63824–63839.

[31] Jingxuan Tan, Xin Liao, Jiate Liu, Yun Cao, and Hongbo Jiang. "Channel attention image steganography with generative adversarial networks". In: *IEEE Transactions on Network Science and Engineering (TNSE)* 9.2 (2022), pp. 888–903.

[32] Jianhua Yang, Fei Shang, Yi Liao, and Yifang Chen. "Toward high capacity and robust JPEG steganography based on adversarial training". In: *Security and Communication Networks* 2023.1 (2023). Ed. by Hanzhou Wu, p. 3813977.

[33] Varsha Kishore, Xiangyu Chen, Yan Wang, Boyi Li, and Kilian Weinberger. "Fixed neural network steganography: Train the images, not the network". In: *International Conference on Learning Representations (ICLR)*. OpenReview.net, 2022.

[34] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples". In: *International Conference on Learning Representations (ICLR)*. Ed. by Yoshua Bengio and Yann LeCun. 2015.

[35] Solène Bernard, Patrick Bas, John Klein, and Tomáš Pevný. "Backpack: A backpropagable adversarial embedding scheme". In: *IEEE Transactions on Information Forensics and Security (TIFS)* 17 (2022), pp. 3539–3554.

[36] Tomáš Filler and Jessica Fridrich. "Minimizing additive distortion functions with non-binary embedding operation in steganography". In: *International Workshop on Information Forensics and Security (WIFS)*. IEEE. 2010, pp. 1–6.

[37] Andrew Viterbi. "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm". In: *IEEE Transactions on Information Theory (TIT)* 13.2 (1967), pp. 260–269.

[38] Olaf Köhler, Cecilia Pasquini, and Rainer Böhme. "On the statistical properties of syndrome trellis coding". In: *International Workshop on Digital Watermarking (IWDW)*. Ed. by Christian Kraetzer, Yun-Qing Shi, Jana Dittmann, and Hyoung Kim. LNCS. Springer. 2017, pp. 331–346.

[39] Tomáš Filler and Jessica Fridrich. "Gibbs construction in steganography". In: *IEEE Transactions on Information Forensics and Security (TIFS)* 5.4 (2010), pp. 705–720.

[40] Bin Li, Ming Wang, Jiwu Huang, and Xiaolong Li. "A new cost function for spatial image steganography". In: *International Conference on Image Processing (ICIP)*. IEEE. 2014, pp. 4206–4210.

[41] Vahid Sedighi, Jessica Fridrich, and Rémi Cogranne. "Content-adaptive pentary steganography using the multivariate generalized Gaussian cover model". In: *Media Watermarking, Security, and Forensics (MWSF)*. Ed. by Adnan Alattar, Nasir Memon, and Chad Heitzenrater. Vol. 9409. SPIE Proceedings. SPIE. 2015, 94090H.

[42] Vojtěch Holub and Jessica Fridrich. "Designing steganographic distortion using directional filters". In: *International Workshop on Information Forensics and Security (WIFS)*. IEEE. 2012, pp. 234–239.

[43] Vojtěch Holub, Jessica Fridrich, and Tomáš Denemark. "Universal distortion function for steganography in an arbitrary domain". In: *EURASIP Journal on Information Security (JIS)* 2014.1 (2014), p. 1.

[44] Rémi Cogranne, Quentin Giboulot, and Patrick Bas. "The ALASKA steganalysis challenge: A first step towards steganalysis". In: *Workshop on Information Hiding and Multimedia Security (IH&MMSec)*. Ed. by Rémi Cogranne, Luisa Verdoliva, Siwei Lyu, Juan Troncoso-Pastoriza, and Xinpeng Zhang. ACM, 2019, pp. 125–137.

[45] Jiwen Yu, Xuanyu Zhang, Youmin Xu, and Jian Zhang. "CRoSS: Diffusion model makes controllable, robust and secure image steganography". In: *Advances in Neural Information Processing Systems (NeurIPS)*. Ed. by Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine. Vol. 36. 2023, pp. 80730–80743.

[46] Rémi Cogranne, Quentin Giboulot, and Patrick Bas. "Steganography by minimizing statistical detectability: The cases of JPEG and color images". In: *Workshop on Information Hiding and Multimedia Security (IH&MMSec)*. Ed. by Christian Riess, Franziska Schirrmacher, Irene Amerini, Paolo Bestagini, and Tomáš Pevný. ACM, 2020, pp. 161–167.

[47] Martin Beneš and Rainer Böhme. "Automating color image steganography research from first principles". In: *European Symposium on Research in Computer Security (ESORICS)*. Vol. 16970. LNCS. Springer, 2026, pp. 385–405.

[48] Kangkang Wei, Weiqi Luo, Shunquan Tan, and Jiwu Huang. "Universal deep network for steganalysis of color image based on channel representation". In: *IEEE Transactions on Information Forensics and Security (TIFS)* 17 (2022), pp. 3022–3036.

[49] Mingxing Tan and Quoc Le. "EfficientNet: Rethinking model scaling for convolutional neural networks". In: *International Conference on Machine Learning (ICML)*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. PMLR. PMLR, 2019, pp. 6105–6114.

[50] Yassine Yousfi, Jan Butora, Eugene Khvedchenya, and Jessica Fridrich. "ImageNet pre-trained CNNs for JPEG steganalysis". In: *International Workshop on Information Forensics and Security (WIFS)*. IEEE. 2020, pp. 1–6.

[51] Antoine Mallet and Patrick Bas. "On the possible detectability of image-in-image steganography". In: *International Conference on Image Processing (ICIP)*. IEEE. 2026, pp. 1–6.

[52] Jianhua Yang, Danyang Ruan, Jiwu Huang, Xiangui Kang, and Yun-Qing Shi. "An embedding cost learning framework using GAN". In: *IEEE Transactions on Information Forensics and Security (TIFS)* 15 (2020), pp. 839–851.

[53] Weixuan Tang, Bin Li, Mauro Barni, Jin Li, and Jiwu Huang. "An automatic cost learning framework for image steganography using deep reinforcement learning". In: *IEEE Transactions on Information Forensics and Security (TIFS)* 16 (2021), pp. 952–967.

[54] Dongxia Huang, Weiqi Luo, Peijia Zheng, and Jiwu Huang. "Automatic asymmetric embedding cost learning via generative adversarial networks". In: *International Conference on Multimedia (MM)*. Ed. by Abdulmotaleb Saddik, Tao Mei, Rita Cucchiara, Marco Bertini, Diana Tobon Vallejo, Pradeep Atrey, and M. Hossain. ACM, 2023, pp. 8316–8326.

[55] Dongxia Huang, Weiqi Luo, Minglin Liu, Weixuan Tang, and Jiwu Huang. "Steganography embedding cost learning with generative multi-adversarial network". In: *IEEE Transactions on Information Forensics and Security (TIFS)* 19 (2024), pp. 15–29.

[56] Qi Cui, Zhili Zhou, Ruohan Meng, Shaowei Wang, Hongyang Yan, and Jonathan Wu. "ARES: On adversarial robustness enhancement for image steganographic cost learning". In: *IEEE Transactions on Multimedia (TMM)* 26 (2024), pp. 6542–6553.

[57] Miaoxin Ye, Saixing Zhou, Weiqi Luo, Shunquan Tan, and Jiwu Huang. "GAN-based symmetric embedding costs adjustment for enhancing image steganographic security". In: *International Conference on Multimedia (MM)*. Ed. by Jianfei Cai, Mohan Kankanhalli, Balakrishnan Prabhakaran, Susanne Boll, Ramanathan Subramanian, Liang Zheng, Vivek Singh, Pablo César, Lexing Xie, and Dong Xu. ACM, 2024, pp. 7046–7054.

[58] Saixing Zhou, Miaoxin Ye, Weiqi Luo, Xin Liao, and Kangkang Wei. "Color image steganography using generative adversarial networks with a phased training strategy". In: *Workshop on Information Hiding and Multimedia Security (IH&MMSec)*. Ed. by Shruti Agarwal, Scott Craver, Shan Jia, Chau-Wai Wong, and Benedetta Tondi. ACM, 2025, pp. 142–152.

[59] Verena Lachner and Rainer Böhme. "A critical survey on video steganography: Trends, pitfalls, and recommendations". In: *ACM Computing Surveys (CSUR)* 58.281 (2026), pp. 1–35.

[60] Eric Jang, Shixiang Gu, and Ben Poole. "Categorical reparameterization with Gumbel-softmax". In: *International Conference on Learning Representations (ICLR)*. OpenReview.net, 2017.

[61] Jan Butora, Yassine Yousfi, and Jessica Fridrich. "How to pretrain for steganalysis". In: *Workshop on Information Hiding and Multimedia Security (IH&MMSec)*. Ed. by Dirk Borghys, Patrick Bas, Luisa Verdoliva, Tomáš Pevný, Bin Li, and Jennifer Newman. ACM, 2021, pp. 143–148.

[62] Ron Crandall. *Some notes on steganography*. Posted on steganography mailing list. 1998.

[63] Jessica Fridrich, Miroslav Goljan, Petr Lisoněk, and David Soukal. "Writing on wet paper". In: *IEEE Transactions on Signal Processing (TSP)* 53.10 (2005), pp. 3923–3935.

[64] Jessica Fridrich. *Steganography in digital media: Principles, algorithms, and applications*. Cambridge University Press, 2009. ISBN: 978-0-521-19019-0.

[65] Andreas Westfeld. "Multimedia steganography". In: *Information Hiding*. Ed. by Stefan Katzenbeisser and Fabien Petitcolas. Artech House, 2016, pp. 21–43.

[66] Patrick Bas, Rémi Cogranne, and Marc Chaumont. "Steganography: Embedding data into multimedia content". In: *Multimedia Security 1: Authentication and Data Hiding*. Ed. by William Puech. Wiley-ISTE, 2022. Chap. 5, pp. 161–188. ISBN: 978-1-78945-026-2.

[67] Claude Shannon. "A mathematical theory of communication". In: *The Bell System Technical Journal* 27.3 (1948), pp. 379–423.

[68] Tomáš Pevný, Tomáš Filler, and Patrick Bas. "Using high-dimensional image models to perform highly undetectable steganography". In: *Information Hiding (IH)*. Ed. by Rainer Böhme, Philip Fong, and Reihaneh Safavi-Naini. LNCS. Springer, 2010, pp. 161–177.

[69] Ishaan Gulrajani, Faruk Ahmed, Martín Arjovsky, Vincent Dumoulin, and Aaron Courville. "Improved training of Wasserstein GANs". In: *Advances in Neural Information Processing Systems (NIPS)*. Ed. by Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna Wallach, Rob Fergus, S. Vishwanathan, and Roman Garnett. Vol. 30. 2017, pp. 5767–5777.

[70] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. "Which training methods for GANs do actually converge?" In: *International Conference on Machine Learning (ICML)*. Ed. by Jennifer Dy and Andreas Krause. PMLR. PMLR, 2018, pp. 3478–3487.

[71] Weixuan Tang, Shunquan Tan, Bin Li, and Jiwu Huang. "Automatic steganographic distortion learning using a generative adversarial network". In: *IEEE Signal Processing Letters (SPL)* 24.10 (2017), pp. 1547–1551.

[72] Jianhua Yang, Danyang Ruan, Xiangui Kang, and Yun-Qing Shi. "Towards automatic embedding cost learning for JPEG steganography". In: *Workshop on Information Hiding and Multimedia Security (IH&MMSec)*. Ed. by Rémi Cogranne, Luisa Verdoliva, Siwei Lyu, Juan Troncoso-Pastoriza, and Xinpeng Zhang. ACM, 2019, pp. 37–46.

[73] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional networks for biomedical image segmentation". In: *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. Ed. by Nassir Navab, Joachim Hornegger, William Wells, and Alejandro Frangi. LNCS. Springer. 2015, pp. 234–241.

[74] Duc-Tien Dang-Nguyen, Cecilia Pasquini, Valentina Conotter, and Giulia Boato. "RAISE: A raw images dataset for digital image forensics". In: *Multimedia Systems Conference (MMSys)*. Ed. by Wei Ooi, Wu-chi Feng, and Feng Liu. ACM, 2015, pp. 219–224.

[75] Clement Tsang and Jessica Fridrich. "Steganalyzing images of arbitrary size with CNNs". In: *Media Watermarking, Security, and Forensics (MWSF)*. Ed. by Adnan Alattar, Nasir Memon, and Gaurav Sharma. IS&T, 2018, pp. 121–1–121–8.

A Technical appendices

A.1 Detailed method and setup

Our method Figure 5 shows our system model, complementing Figure 2b with further details. In orange are the input and output of the system—the cover and the stego image, each with 3 channels for the RGB color space. The generator consists of a depthwise-separable block: a depthwise convolution with 7×7 kernels, no bias, and 3×64 output channels; a leaky ReLU that introduces non-linearity; and a pointwise convolution with 1×1 kernels, bias term, and 27 output channels, that correspond to all the combinations of ± 1 changes in an RGB color pixel:

0 +1 -1 0 +1 -1 0 +1 -1 0 +1 -1 0 +1 -1 0 +1 -1 0 +1 -1 0 +1 -1
0 0 0 +1 +1 +1 -1 -1 -1 0 0 0 +1 +1 +1 -1 -1 -1 0 0 0 +1 +1 +1 -1 -1 -1
0 0 0 0 0 0 0 0 0 +1 +1 +1 +1 +1 +1 +1 +1 +1 -1 -1 -1 -1 -1 -1 -1 -1 -1

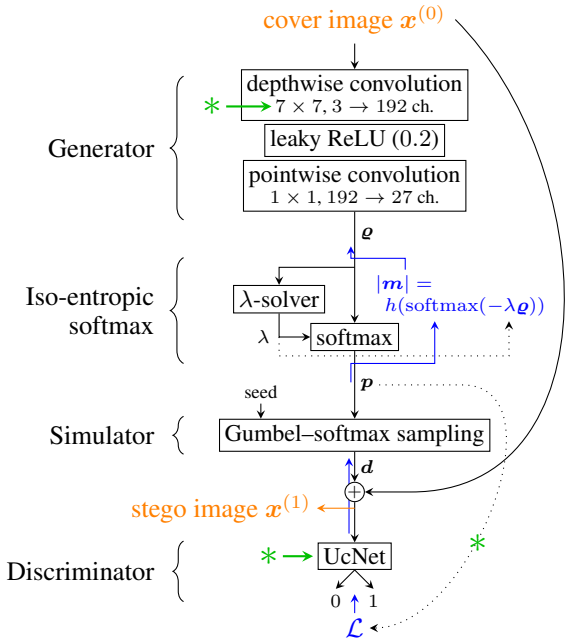


Figure 5: Training setup of our generator. Blue arrows are the gradient paths during back-propagation. (∗) mark the components considered in the ablation.

Continuing with Figure 5, in blue are the gradient paths from the discriminator that illustrate the differentiability tricks. In the iso-entropic softmax, we replace the automatic differentiation of the solver with the gradient of the constraint. The value of λ is carried over from the forward pass.

The green markers denote design aspects that are covered by the ablation study in Section 3.4: the filter size in the depthwise convolution of the generator; the architecture of the discriminator; and the change rate budget in the loss function, which features the probability map $\boldsymbol{p}$.

The seed in the Gumbel-softmax sampler is an indirect placeholder for the choice of message $\boldsymbol{m}$ and key $\boldsymbol{k}$, enabling reproducibility. The Gumbel-softmax block conspicuously lacks the input for the temperature τ, which typically controls the simulated annealing, i. e., the exploration-exploitation tradeoff [60]. This is because in our method, this tradeoff is already controlled by the value of λ through the choice of the message size $|\boldsymbol{m}|$. As a result, τ is not a free parameter in our setup.

Training of detectors: α-curriculum Simulated annealing is a popular technique in steganalysis, where it is known as α-curriculum [61]. Usually, a detector for lower α is initialized with weights from a detector trained for higher α, which tends to improve its performance. The detector for the highest α is initialized with the weights acquired from an ImageNet-pretrained model [50]. We adopt this technique for detectors of the signal processing-based method and our variational approach, using the curriculum $\alpha = 1 \to 0.4 \to 0.2 \to 0.1 \to 0.05$ bits per element. Each curriculum step follows the setup described in Section 3.1 of the main text.

A.2 Coding for steganography

In this appendix, we summarize the key principles of coding for steganography that may not be widely recognized in machine learning, specifically the results in [62], [63], and [36]. Recommended chapters in textbooks on this topic are [64, Ch. 9], [65, Ch. 2], and [66, Sec. 5.2].

Preliminaries The steganographic method must define a mapping from cover elements to symbols of the code alphabet. Without loss of generality, we assume binary code symbols $\{0, 1\}$, and a mapping that simply takes the least significant bits (LSBs) of the intensity values x_i. This way, a sequence of bits, $\boldsymbol{b}^{(0)}$, can be extracted from the cover, $\boldsymbol{x}^{(0)}$. In order to embed a message given as a

maximum-entropy bit string, the sender has to find a sequence $\boldsymbol{b}^{(\boldsymbol{m})}$, which reproduces the message $\boldsymbol{m}$ if the mapping is applied to the stego object, $\boldsymbol{x}^{(\boldsymbol{m})}$, followed by decoding using the correct key, $\boldsymbol{k}$.

The *embedding operation* performs the actual cover modification by replacing each element, $x_i^{(0)}$, of the cover, where $b_i^{(0)} \neq b_i^{(\boldsymbol{m})}$, with the nearest value, $x_i^{(1)}$, that maps to the target symbol, $b_i^{(\boldsymbol{m})}$. "Nearest" is often defined by the absolute difference, but other distance metrics are possible, too. For the binary symbols mapped from LSBs in the example above, the embedding operation simply increments or decrements mismatching values by one to ensure that the LSBs match.

Generalized selection channel with random block codes Sender and recipient use a cryptographically secure pseudo-random number generator, seeded with $\boldsymbol{k}$, to produce a shared sparse matrix, $\boldsymbol{H}$, over the code alphabet. $\boldsymbol{H}$ has as many columns as there are cover elements, and as many rows as there are message bits.

By convention, the recipient extracts the message by computing the matrix product,

$$\boldsymbol{m} = \boldsymbol{H}\boldsymbol{b}^{(\boldsymbol{m})}, \tag{4}$$

over the field GF(2) in the case of the binary code alphabet. This corresponds to applying a parity check matrix in channel decoding, yielding the *syndrome* (error pattern); hence the name "syndrome coding" in steganography.

To ensure that this decoding rule works, the sender must solve,

$$\boldsymbol{H}\boldsymbol{v} = \boldsymbol{m} - \boldsymbol{H}\boldsymbol{b}^{(0)}, \tag{5}$$

for the change vector, $\boldsymbol{v}$, indicating which symbols in $\boldsymbol{b}^{(0)}$ must be flipped to obtain $\boldsymbol{b}^{(\boldsymbol{m})}$. The solution can be found, for example, with Gaussian elimination. If $\boldsymbol{H}$ is sparse in the right way, solutions exist with very high probability [63].

Crucially, if $\boldsymbol{H}$ has more columns than rows, the sender can exclude cover elements from being modified by eliminating the corresponding columns from $\boldsymbol{H}$ before solving for $\boldsymbol{v}$, and inserting zeros ("do not modify") at the respective positions in $\boldsymbol{v}$. This means that the sender can avoid selecting risky pixels, and the recipient does not need to know which pixels were selected or excluded in order to correctly reconstruct the message. This property is known as "selection channel" [11].

In expectation, there is no coding loss with random block codes. However, since solvers for $\boldsymbol{v}$ tend to have cubic complexity in the number of equations (rows), it is common to split covers with thousands of elements into smaller blocks. (Compare the left panel to the center panel in Fig. 6.) This incurs some coding loss, meaning that every block needs a few more columns than rows to keep the probability of finding solutions for all blocks high.

Considering costs with convolutional codes If the dimensions of $\boldsymbol{H}$ yield an underspecified system, multiple solutions are feasible. Syndrome trellis codes (STCs) take advantage of this by making it possible to efficiently find a solution that minimizes the sum of costs associated with each symbol transition, i. e.,

$$\min_{\boldsymbol{v}} \sum_i v_i \cdot \varrho_i, \qquad \text{s. t.} \quad \boldsymbol{m} = \boldsymbol{H}\boldsymbol{b}^{(\boldsymbol{m})}. \tag{6}$$

Observe that each transition at every position in the vector can have its own cost. In variational cover modification steganography, it is this cost vector, $\boldsymbol{\varrho}$, that is predicted by the generator G.

To efficiently find solutions where the probability of changing an element, p_i, is inversely proportional to its cost ϱ_i, STCs take inspiration from convolutional decoding using the Viterbi algorithm [37]. The sparse matrix, $\boldsymbol{H}$, implied by the algorithm's trellis, takes the form of the right panel in Fig. 6. Overlapping submatrices $\hat{\boldsymbol{H}}$ of small dimensions—typical "constraint heights" are in the range between 8 and 15 for binary alphabets—ensure flexibility in the choice of positions to change, while keeping the search space tractable.

Costs are scaled to probabilities by the message size $|\boldsymbol{m}|$ through the binary entropy,

$$|\boldsymbol{m}| = h(\boldsymbol{p}) = h(\text{softmax}(-\lambda\boldsymbol{\varrho})). \tag{7}$$

This connects to Eq. 1 in the main text. The probabilities $\boldsymbol{p}$ are, for a fixed cover image and its cost vector $\boldsymbol{\varrho}$, obtained by iterating over possible keys $\boldsymbol{k}$ and messages $\boldsymbol{m}$. The entropy is a result of

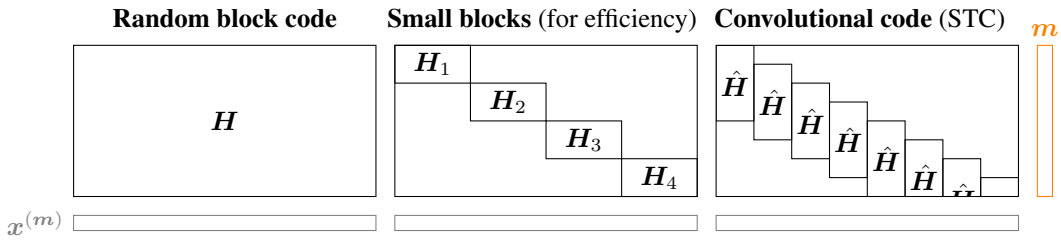


Figure 6: Illustration of the parity check matrices used in syndrome coding. See text for details.

the Shannon's source coding theorem [67, Sec. 9]. The Boltzmann distribution, embodied by the $\mathrm{softmax}$ function, is the solution to minimal sum of costs at a fixed level of entropy.

For short messages in large covers, the number of embedding changes can be much lower than the message size. This effect of increasing *embedding efficiency* can be seen in Fig. 4 (in the main text), where the wedge between change rates, $\overline{\boldsymbol{d}}$, and embedding rates, α, widens for shorter messages. Embedding efficiency, e, is defined as the number of message bits per changed element. To get an intuition for this effect, consider that there are $n+1$ ways to modify at most one of n positions. Therefore, we can communicate $\log_2(n+1) \cdot {}^{(n+1)}\!/_{n}$ bits per change. (The first factor uses the definition of entropy. The second factor accounts for the fact that we don't need to make any change with probability ${}^{1}\!/_{(n+1)}$.) Empirically, change vectors $\boldsymbol{v}$ produced by STCs are realizations of a categorical distribution with parameter vector $\boldsymbol{p}$ [38]. The embedding simulator, used for the training of the generator and in the evaluation, parametrizes the categorical sampler in exactly this way.

Since the matrix $\boldsymbol{H}$ of STCs has a particular structure, a naive use could reveal information to the adversary. To maintain security, a pseudo-random permutation dependent on $\boldsymbol{k}$ is applied to $\boldsymbol{b}^{(0)}$ before coding, and inverted for $\boldsymbol{b}^{(m)}$ before applying the embedding operation. With knowledge of $\boldsymbol{k}$, the recipient can recover this permutation and apply it prior to extraction.

Finally, binary code alphabets are often suboptimal. Mappings to ternary symbols support individual costs for incrementing and decrementing [68]. As our method works on images with three color channels, it uses 27-ary symbols to learn the costs of each direction in the color cube.

A.3 Security with coding

Converting the costs to probabilities for the simulator using the entropy constraint (Eq. 1) assumes that there is zero coding loss. Practical codes, however, incur a strictly positive coding loss, i. e., they make slightly more changes per message bit than asymptotically necessary. These additional changes may increase the detectability in practical use. Hence, the simulated results tend to overestimate the practical security, which we acknowledge in the limitations (Sec. 3.2).

To quantify this gap, we embed with actual STCs into 200 covers, randomly selected from the ALASKA test set, and measure both the coding loss and its effect on detectability. The STCs operate on 27-ary symbols, one per pixel, matching the costs our generator predicts for all 27 possible changes of a color pixel, $\{-1, 0, +1\}^3$. For each cover, we draw a uniformly random message and a random key, which determines the permutation of the pixels and their costs before coding, as well as the full-rank submatrix $\hat{\boldsymbol{H}}$ over $\mathrm{GF}(3^3)$. The width of $\hat{\boldsymbol{H}}$ follows from the embedding rate, rounded down so that the whole message fits. Its height, the constraint height h, trades coding loss for complexity. As the trellis has 27^h states, we use $h \in \{2, 3\}$.[1]

Figure 7 shows the distribution of the coding loss over the covers, i. e., the relative decrease of the embedding efficiency compared to the simulator, ${}^{(e_{\text{sim}} - e_{\text{stc}})}\!/_{e_{\text{sim}}}$. The embedding efficiency, e, is defined as in Appendix A.2. The loss decreases with the constraint height and depends only weakly on the embedding rate. Occasional negative values occur because e_{sim} is an expectation, whereas e_{stc} is computed from a single realization per image.

[1] The Viterbi algorithm stores one back-pointer per state and trellis column, i. e., about Nq^h bytes, neglecting the columns for the message symbols. For $q = 27$ and $N = 296^2$ px, this amounts to 1.7 GB for $h = 3$ and 47 GB for $h = 4$. Embedding with $h = 3$ takes about 10 s per image with our implementation.

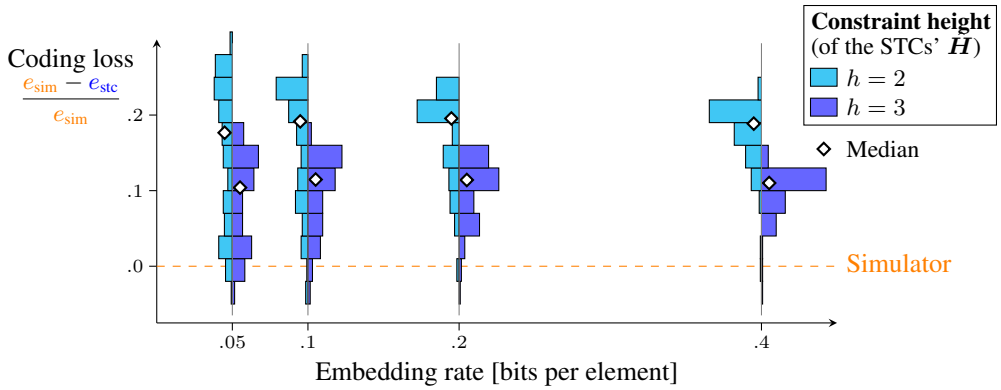


Figure 7: Coding loss of 27-ary STCs relative to the simulator, per embedding rate. Mirrored histograms over 200 covers show the constraint heights $h = 2$ (left) and $h = 3$ (right). Increasing h shrinks the loss from about 19 % to 11 %. The embedding rate affects the spread of the loss, but barely its median.

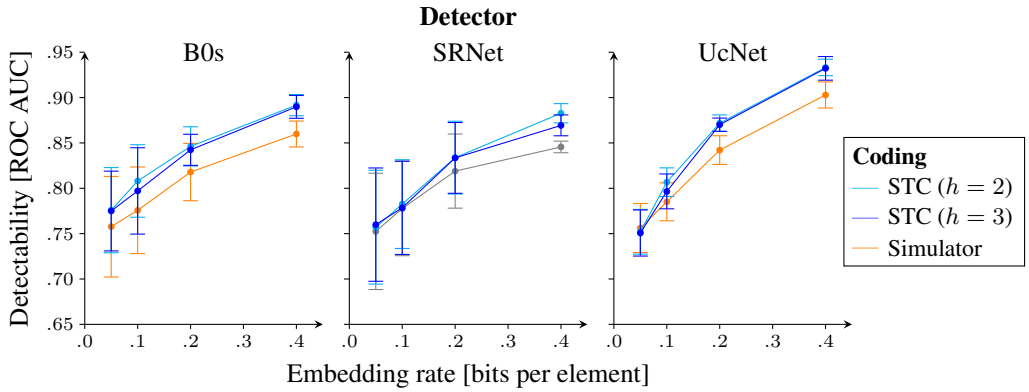


Figure 8: Security of embedding with actual 27-ary STCs compared to the simulator on the same 200 covers, using the detectors trained against the simulator. STC embedding is slightly more detectable, and the gap grows with the message length. The error bars denote ± 1 standard error taken over three generator checkpoints and the matched detectors.

Figure 8 compares the STC embedding to the simulator in terms of detectability. We apply the detectors trained against the simulator in the main results to the STC and simulator stego images of the same 200 covers, i. e., without retraining. STC embedding is slightly more detectable throughout, and the gap grows with the embedding rate, from about $+0.01$ ΔROC AUC at 0.05 bpe to about $+0.03$ at 0.4 bpe. This is in line with the coding loss in Figure 7: at a fixed loss, longer messages require more additional changes. Accordingly, the larger constraint height $h = 3$ tends to be less detectable than $h = 2$. The slight overestimation of security due to the simulator does not affect our key findings: the end-to-end methods remain perfectly detectable, and the comparison with signal processing-based methods is fair since they are also evaluated using the simulator.

A.4 Selecting the generator state

GANs are notorious for instabilities during training [69, 70]. In our adversarial system, this would manifest as large oscillations in detectability of generators from different update steps.

We explore this concern in Figure 9, by training separate detectors for different generator states, that were acquired with a single adversarial training run. The error bars were obtained with bootstrapping, hence representing variation within the dataset. The least and the most detectable generator states are 0.06 units apart (in terms of ROC AUC from 0.5 to 1.0). While the selection of the generator weights

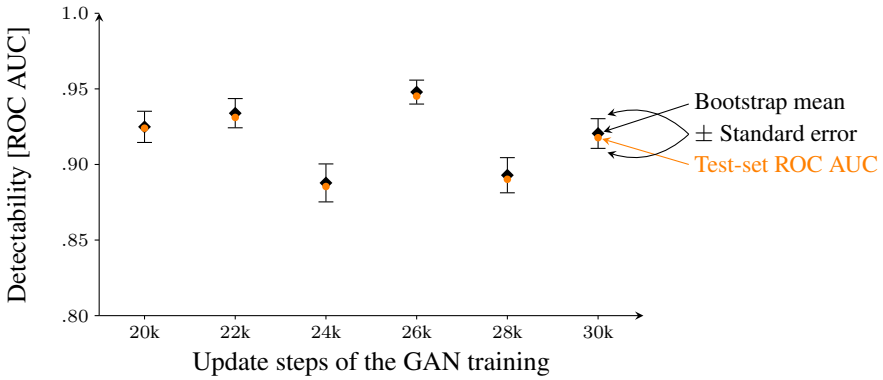


Figure 9: Sensitivity of the performance to the choice of the GAN checkpoint. We train a separate B0s for each training step at $\alpha = 0.4$ bpe. Our baseline uses the weights after 24k update steps. The error bars are acquired by bootstrapping (250 random cover-stego pairs, drawn $100\times$ with replacement).



(a) Training setup.

(b) Inference setup.

Figure 10: Compared to our variational method (Fig. 2), existing methods use the generator to predict probability maps. Therefore, their architectures must have a scalar bottleneck to satisfy the entropy constraint. This means fixed (or inflexible) input dimensions, and retraining for each embedding rate.

plays some role, the training remains fairly stable. We observe no perfectly detectable states or large oscillations.

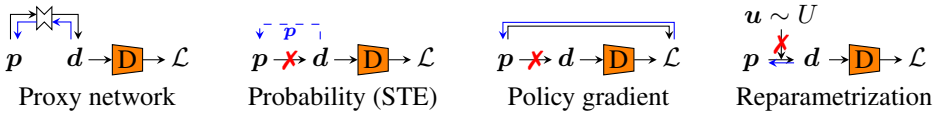
A.5 Predecessors to variational cover modification steganography

Methods that learn the probabilities p As discussed in Sec. 4, there exist steganographic methods that attempt to model the change distribution, and hence could be classified as "variational" in a broad sense. All of these approaches use generator networks that directly infer the probability map, $\boldsymbol{p}$, for fixed entropy. We illustrate this idea in Figure 10, to be compared with Figure 2 in the main text.

The probability map is used in a differentiable sampler. Fig. 11 illustrates several ways that have been explored in the literature: by employing a proxy network [71]; by straight-through replacing the gradient of the noise with the probability [72]; by using a specific loss that features the probability $\boldsymbol{p}$, which allows the gradient to pass back [53]; or by reparametrization by double-tanh construction [52], which separates the randomness, and establishes an arithmetic (i. e., differentiable) relationship of $\boldsymbol{p}$ and $\boldsymbol{d}$. We compare the double-tanh and the Gumbel-softmax constructions below in Sec. A.6.

Predicting the probability moves the entropy constraint from an external iso-entropic softmax into the generator itself. This approach has three major limitations, which make it impractical.

1. *Large and constrained generator*. Only networks with a scalar bottleneck can yield output probabilities of fixed entropy, because the entropy is calculated as a sum over terms from each element. All the methods thus employ variants of the U-Net architecture [73].



Proxy network | Probability (STE) | Policy gradient | Reparametrization

Figure 11: Existing approaches to differentiable categorical sampling. Forward paths are black, gradient paths are blue, (✗) marks non-differentiable paths.

Table 6: Security of the PTS method compared to our variational design.

ROC AUC (↓)	Steganographic method and embedding rate [bits per element]							
	PTS* (39.1M param.)				*Ours* (15K param.)			
Detector	.4	.2	.1	.05	.4	.2	.1	.05
B0s	.941	.873	.767	.660	.868	.850	.804	.769
(std. error)	(.004)	(.000)	(.005)	(.008)	(.015)	(.022)	(.044)	(.046)
SRNet	.926	.797	.705	.565	.869	.842	.797	.771
(std. error)	(.001)	(.012)	(.004)	(.036)	(.006)	(.036)	(.056)	(.059)
UcNet	.897	.783	.671	.587	.918	.864	.810	.766
(std. error)	(.004)	(.004)	(.005)	(.010)	(.013)	(.019)	(.025)	(.032)

*For PTS, the dataset was top-left cropped to 3×256^2 to meet the method's constraints.

2. *Fixed cover image dimension*. The U-Net transforms the image to scalar by repeated convolutions with stride 2. The decoder mirrors this using transposed convolutions with stride 2. This effectively restricts the input image (cover) to fixed dimensions, typically 256^2 or multiples thereof.
3. *Training with a fixed message length*. The message length, i. e., the output entropy, is not an explicit input, but an implicit constraint in the loss function. During training, the output of the generator must learn for a single message size over all the covers. This may hinder generalization to other message sizes.

There is also a fourth limitation that applies to nearly all the approaches:

4. *Single channel*. The majority of the existing methods only apply to grayscale cover images, which are very uncommon in real-world communication channels. The only method for color is the phased-training strategy (PTS) [58]. For completeness, we compare our method to PTS in Table 6. Our method is less detectable than PTS only for longer messages, and lags behind for shorter ones. Error bars for PTS are computed from a single generator state, because we use pretrained weights provided by the authors. Our generator remains competitive for longer messages with a fraction of the parameters and for arbitrary input dimensions.

Relation to Beneš and Böhme [47] This paper and [47] report selected results from the same project to two relevant research communities: machine learning and security. Both communities differ in their conventions for evaluation and presentation. This paper and [47] use the same generator, discriminator, and adversarial training protocol, but differ in the framing and analysis methods.

The only small overlap is in Table 3, which reports the configurations covered in [47, Tab. 3], as this paper would be incomplete without this comparison. Compared to [47, Tab. 3], Table 3 shares two of the seven steganographic methods and three of the four detection methods, adds two more detector runs per configuration to estimate standard errors, and reports a different performance metric (ROC AUC versus P_E in [47]).

Apart from this small overlap, all experiments and results reported in this paper are novel, meaning they do not appear in [47]. In particular, the emphasis on learning the least detectable stego-noise from the cover distribution, the notion of "variational" cover modification, and the relation to other learning-based approaches to steganography are unique to this paper. By contrast, [47] emphasizes the automation of the research process, reports and compares runtime behavior, and spends attention on the interpretation of the learned cost function in color images.

A.6 Selection of solvers and simulators

The steganographic simulator by Filler et al. [39] requires solving Equation 1 for λ. The authors propose a simple solver consisting of an exponential bracket search, followed by a binary search within the bracket. Their solution, including the hyperparameters for initialization and stopping criteria, became the de-facto standard in research code. Bernard et al. [35], who proposed the differentiable

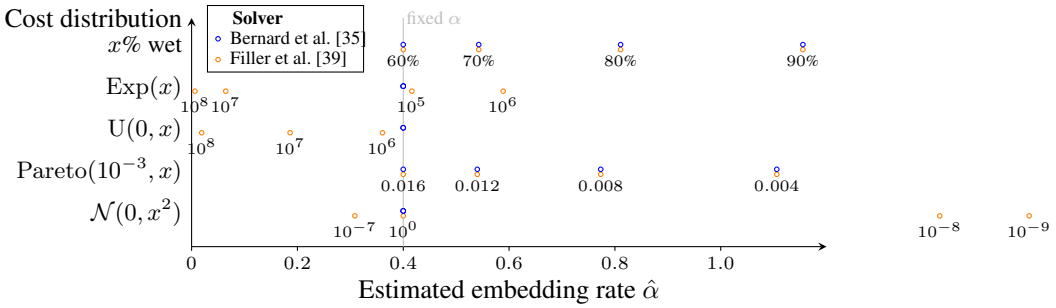


Figure 12: Accuracy of solvers for the iso-entropic softmax under extreme cost distributions. Marks show the attained rate, $\hat{\alpha}$, for i. i. d. costs with parameter x (below each mark) against the target $\alpha = 0.4$ (gray line). The solver by Bernard et al. [35], which we use, is more stable. The standard error bars are smaller than the marks and thus invisible.

simulator, use a similar construction, but with changes to the configuration. In Figure 12, we compare the two solvers in terms of their stability under extreme cost distributions for the target embedding rate, $\alpha = 0.4$. Inaccuracies arise from numerical instabilities or from reaching the maximal number of iterations during the search. For well-behaved distributions, both solvers yield accurate λ estimates. For extreme exponential, uniform, and normal cost distributions, the solver from [39] misses the target, whereas the solver from [35] remains accurate. For the Pareto and wet cost distributions, both solvers miss the target alike. As the solver from [35] is more stable, we use it in our adversarial training.

While the differentiable simulator from [35] uses the Gumbel-softmax reparametrization, some literature employs a double-tanh construction [52], designed for binary (and, in principle, ternary) sampling. We compare how well these two approaches converge to the target (categorical) distribution $\boldsymbol{p}$. We draw a number of samples ($\boldsymbol{d}$'s), average them, and calculate the Kullback–Leibler divergence (D_{KL}). As shown in Figure 13, both the Gumbel-softmax and double-tanh estimators converge to the target distribution at the same rate as the true categorical sampler. The choice is thus decided by the alphabet: our generator models the 27-ary joint changes of a color pixel, which the double-tanh construction cannot represent. It is restricted to binary and ternary alphabets.

A.7 Generalization to a different dataset

To test generalization beyond ALASKA (cf. limitations in Sec. 3.2), we repeat the main experiments on RAISE [74], a popular image-forensic dataset of 8 156 images from 3 cameras. We fetch the developed, uncompressed images ("TIFF"), published by the authors, and convert them to the dimension 3×296^2 with the following two-step procedure.

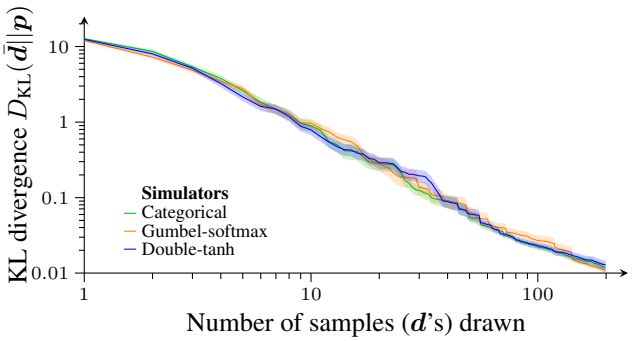


Figure 13: Convergence of the estimated categorical distribution $\bar{\boldsymbol{d}}$ with 3 outcomes to the true distribution $\boldsymbol{p}$, across different simulators. All simulators converge at the same rate. The error bars are standard errors over 100 seeds.

Table 7: Security of our method compared to common signal processing-based benchmarks on the RAISE dataset, analogously to Table 3 on ALASKA. Everything trainable is retrained on this dataset.

ROC AUC (↓)	Steganographic method and embedding rate [bits per element]											
	Signal processing-based								**Learnt**			
	HILL				WOW				*Ours*			
Detector	.4	.2	.1	.05	.4	.2	.1	.05	.4	.2	.1	.05
B0s	.997	.987	.952	.877	.994	.979	.928	.860	.837	.757	.715	.664
(std. error)	(.000)	(.001)	(.001)	(.009)	(.001)	(.002)	(.008)	(.003)	(.009)	(.013)	(.020)	(.022)
SRNet	.995	.976	.931	.841	.996	.975	.927	.845	.899	.855	.832	.815
(std. error)	(.002)	(.001)	(.003)	(.004)	(.001)	(.003)	(.005)	(.008)	(.007)	(.016)	(.022)	(.031)
UcNet	.993	.952	.858	.766	.990	.946	.855	.751	.882	.819	.791	.770
(std. error)	(.001)	(.002)	(.004)	(.003)	(.001)	(.003)	(.004)	(.006)	(.011)	(.012)	(.017)	(.016)

- Crop the image to the resolution s^2 using the *smart crop* [75, 44], where the size s is uniformly drawn from $\{296, \dots, \min(\text{height}, \text{width})\}$.
- Resize the image to 296^2 with `pillow`'s default settings (e. g., bicubic interpolation).

The random resampling factor suppresses demosaicing traces, which would otherwise reduce the options for secure steganographic embedding. Still, RAISE is substantially more homogeneous than ALASKA, which randomly varies the demosaicing method as well as whether and in which order cropping and resizing are applied. We split RAISE 70:15:15 into training, validation, and test sets, retrain our generator on it, and also train all detectors anew.

Table 7 compares our method to the signal processing-based methods on RAISE, to be compared with Table 3. Our method is more secure than the signal processing-based methods across nearly all embedding rates and detectors. A notable gap in favor of our method is at 0.4 bpe. We attribute this to learning: a more homogeneous source, such as RAISE, has more consistent statistics, which the learned costs can fit, whereas hand-crafted costs are the same for any source.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification:

   Claim Current learning-based image steganography is vulnerable.
   Contribution Evidence in Figure 1 and Table 2.

   Claim Variational cover modification steganography tackles this problem ...
   Contribution The entire Section 2.3.

   Claim Syndrome coding embeds the message within the distribution constraint and can retrieve it exactly.
   Contribution Section 2.3 with background on coding in Appendix A.2.

   Claim Experiments demonstrate significant security improvements.
   Contribution Section 3.3.

   Claim Ablations support the generator architecture and the training protocol.
   Contribution Section 3.4.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Two of the six paragraphs in Section 4 focus on limitations of the approach. Limitations that may impair the generalizability of our experimental results are collected in the dedicated Section 3.2.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.

- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper does not include theoretical results

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section 3.1 describes the complete setup. Appendix A.1 provides even more detail on every aspect. Code, access to data, splits, trained models, prediction vectors (intermediate results), and example images for visual inspection are included in the supplemental material. No proprietary data or access to third-party cloud resources is necessary to reproduce the results.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often

one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.

- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We release the code and data preprocessing scripts, and derived data. We took note of the submission guidelines. The ALASKA dataset, our primary data source, is publicly available and free of charge [44].

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 3.1 describes the setting to a level of detail that is necessary to appreciate the results. Appendix A.1 reports more details. The supplementary material also contains code for calculating the metrics reported in the tables.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We report standard errors for all relevant metrics, including for the ablations, and explain how they are computed in the text. When comparing quantitative differences, we interpret them in relation to the standard errors (and implied standard deviations, where appropriate). Note that calculating the standard errors involves retraining detector models. Due to resource limitations before the submission deadline, some standard errors are still missing in Table 3. We will supply them in time for the camera-ready deadline. Some of the estimates in Table 2 do not have standard errors. In Section 3.3, we explain why.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We document the computational resources in a dedicated paragraph of Section 3.1, broken down by adversarial training, training of the method we compare to (to reduce out-of-distribution effects), and detector training. Inference does not contribute significantly. For example, the results presented in Table 2 represent a cumulative computational cost of 23 GPU days. The replication on RAISE, as requested by the reviewers, costed another 5 GPU days.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.

- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We do not process any data relating to natural persons or interact with human subjects in any way.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss the societal impact at the end of Section 4. This part is brief because our research does not result in any new or unexpected positive or negative societal impacts.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [Yes]

Justification: All models were trained using public datasets, thereby eliminating the risk of any unintended information leakage.

Steganography may be considered a dual-use good. Strong steganographic methods may be subject to import, export, or usage controls in certain jurisdictions. In order to avoid any potential misuse of our methods and models by malicious parties, we follow the conventions of the security community and do not release operational embedding functions.

Initially, all of the experiments presented in this paper were conducted with an embedding simulator that embeds a random bit string rather than user-provided payload messages. This functionality is, in principle, sufficient to reproduce all of our results. On request of the reviewers, we created a proof-of-concept of the 27-ary STCs in order to reconfirm that the key findings are valid despite the use of the simulator (see Apx. A.3). We follow the conventions in the community and hold back this proof-of-concept code.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All original authors of code and data are cited. We will release our code and models under a CC-BY license.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: Our code is documented to the extent necessary for replication and further experimentation/exploration. Apart from the handful of example images in the supplementary material, we do not create new dataset or derive assets from existing datasets.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: For the submitted version, LLMs were only used to check individual sentences for typographical or grammatical errors. Occasionally, we would allow them to replace a word or find a synonym. (For example, the last sentence is an LLM output generated from

our input: “LLMs were only used for checking individual sentences for typographical or grammatical mistakes. Occassionally we let them replace a word or find synonyms.”)

The camera-ready version was passed through Claude Opus 5.5 for a consistency check. This revealed one confusion of terms in one paragraph of Apx. A.2, which has been corrected.

The proof-of-concept used to produce the results in Apx. A.3 requested by the reviewers was created with the assistance of Claude Opus.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.