Pixel-Stega: Generative Image Steganography Based on Autoregressive Models

Siyu Zhang, Zhongliang Yang, Haoqin Tu, Jinshuai Yang, and Yongfeng Huang

Abstract—In this letter, we explored generative image steganography based on autoregressive models. We proposed Pixel-Stega, which implements pixel-level information hiding with autoregressive models and arithmetic coding algorithm. Firstly, one of the autoregressive models, PixelCNN++, is utilized to produce explicit conditional probability distribution of each pixel. Secondly, secret messages are encoded to the selection of pixels through steganographic sampling (stegosampling) based on arithmetic coding. We carried out quantitative and qualitative assessment on gray-scale and colour image datasets. Experimental results show that Pixel-Stega is able to embed secret messages adaptively according to the entropy of the pixels to achieve both high embedding capacity (up to 4.3 bpp) and nearly perfect imperceptibility (about 50% detection accuracy).

Index Terms—steganography, generative steganography, image steganography, arithmetic coding, autoregressive model.

I. INTRODUCTION

With the rapid development of surveillance technology and the increasingly severe threat to information security [1], [2], the protection of private communication has aroused more and more attention in recent years [3]. Steganography is one of the most predominant techniques against eavesdropping [4], which refers to the problem of sending secret messages embedded in seemingly innocuous carriers over a public channel so that an eavesdropper cannot even detect the presence of the hidden messages.

Digital image is the most widely investigated carrier for steganography due to its large redundancy and frequent use in cyberspace [5], [6]. Traditional image steganographic methods mainly embed secret messages by slightly modifying some insensitive features of the image, either in spatial domain [7]–[10] or in transform domain [11]–[16], while minimizing an heuristically defined distortion function. There are also some methods incorporating with automatic distortion functions based on deep learning approaches like Generative Adversarial Networks (GANs) [17]–[21].

Recent advances in the field of image generation have promoted the development of another paradigm, generative image steganography, which directly generates realistic-looking steganographic images (stego-images) according to the secret messages [22]. It is more flexible but also challenging. In preliminary studies, researchers tried unnatural image synthesis for information hiding, such as texture [23]–[29] and fingerprint [30], [31]. However, such kind of images has limited scope of application.

To generate more generic stego-images, some researchers realized generative image steganography by encoding secret messages into the inputs of generative models [32]–[39]. For example, Hu et al. [32] embedded secret messages into a latent vector $z$, and transformed it into realistic image with the generator of GAN. They trained an additional extractor network for information recovery. Li et al. [33] improved the method by jointly training the generator and the extractor. Chen et al. [34] used a similar strategy with the encoder of a variational autoencoder (VAE). Liu et al. [37] encoded secret messages to the selection of class labels of an Auxiliary Classifier Generative Adversarial Network (ACGAN). However, these methods generally suffer from low embedding capacity, as well as inaccurate information extraction.

To circumvent these difficulties, another line of thought tried to embed secret messages via the outputs of generative models instead of inputs. On condition that the output probability distribution of an image is known, the embedding and extraction of generative image steganography is exactly the equivalent problem of source decoding and encoding, respectively. Specifically, in steganographic embedding or source decoding, secret messages/codes are transformed into images, while in steganographic extraction or source encoding, images are transformed into secret messages/codes. Therefore, a possible way to achieve generative image steganography is to deploy steganographic sampling (stegosampling) strategy based on lossless source coding algorithm on explicit generative models, such as autoregressive models. On the one hand, lossless source coding enables the absolutely accurate extraction. On the other hand, the secret message is able to be encoded into 256 sampling values of all pixels in an image, which enables a considerable embedding capacity as long as the conditional distribution has a large entropy. There have been attempts in generative image steganography based on autoregressive models. Yang et al. [40] proposed a rejection-sampling-based strategy to embed secret messages into the Least Significant Bit (LSB) of each pixel during generation. However, this method is not adaptive to sharp conditional distribution so that it may fail to generate meaningful stego-images, as demonstrated in Figure 2.

In this letter, we further explored generative image steganography based on autoregressive models. We are committed to improving the stegosampling algorithm to pursue both high embedding capacity and high imperceptibility. We proposed Pixel-Stega, which employed the same autoregressive model as Yang et al. [40], but adopted arithmetic coding algorithm.

Our code is available at https://github.com/Mhzzzzz/Pixel-Stega.
At Bob’s end, the secret message \( m \) is extracted by taking the pixel value \( v \) as the input of arithmetic encoding algorithm.

### B. Steg sampling Based on Arithmetic Coding

Arithmetic coding is a kind of entropy coding used in lossless data compression, which encodes a symbol flow by creating a code which represents a fractional value on a real number between 0 and 1 \([42]\). On each symbol recursion, an interval \([l_i, u_i]\) (which is initially set to \([0, 1]\)) is successively partitioned according to the cumulative distribution of the symbol, and narrowed down by taking the sub-interval containing the symbol value as the new interval \([l_{i+1}, u_{i+1}]\). Following Zigler et al. \([43]\), we adopted a fixed-precision implementation of arithmetic coding \([44]\).

1) Steganographic Embedding: In practice, the required precision of digits grows as encoding continues. To mitigate this issue, fixed-precision implementation of arithmetic coding uses \(prc\)-bit registers to store the mantissa of the code and the boundaries of the interval. On this basis, Pixel-Stega generates each pixel \( x_i \) by three steps. Firstly, the conditional probability distribution \( p(x_i|x_{<i}) \) is sorted and the current interval \([l_i, u_i]\) is partitioned into sub-intervals proportional to its cumulative distribution \( c(x_i|x_{<i}) \). This procedure can be seen as a quantization of the distributions and we obtain the quantized probability distribution \( q(x_i|x_{<i}) \). The quantization interval is the current minimum decimal that the registers can represent. Secondly, we determine pixel \( x_i \) to be the value that its sub-interval contains the real number represented by the secret message. We can locate the sub-interval referring to the register storing the secret message. Finally, the same prefix of all numbers in the new interval is the newly embedded secret message. No matter how the embedding procedure continues later, this prefix will not be changed anymore. Thus, registers storing boundaries shift left to overflow the prefix, and the register storing the secret message (more like a sliding
window) also shifts left (the sliding window moves right) by the same number of digits. An example is illustrated in Figure 4. At the first embedding step ($i = 208$), the interval still remains initialized as $[0.00000, 1.00000)$. The interval is partitioned in proportion to $q(x_{208} | x_{<208})$. According to the first $prc$-bit secret message 01111, the sub-interval $[0.01110, 0.10000)$ with a pixel value of 4 is selected as the next interval. Since the open boundary 0.10000 is not available, all numbers in the sub-interval start with the prefix 0.0111. As a result, it is considered that the first 4 bit secret message is encoded to the 208-th pixel. The registers storing boundaries and the secret message shift left by 4 bits to overflow the prefix. This process is repeated until the stego-image is completed.

2) Steganographic Extraction: The extraction procedure is quite an inverse process of the embedding. Firstly, the same operation is conducted to obtain the sorted distributions and partition the interval. Secondly, the current pixel $x_i$ locates the next interval. Finally, we extract the same prefix of numbers in the next interval as the secret message. Registers shift left by the number of digits.

C. Imperceptibility
As is often assumed, the secret message $m$ consists of uniformly distributed random bits, hence the pixel is equivalent to be sampled from the quantized distribution $q(x_i | x_{<i})$, which reveals the advantage of Pixel-Stega in two respects. On the one hand, it is able to make the most of the information provided by the output distribution $p(x_i | x_{<i})$ thus provides a strong guarantee for embedding capacity. The average amount of information that can be embedded at each time step of Pixel-Stega is given by the entropy of the quantized distribution $H(q(x_i | x_{<i}))$, and affected by $H(p(x_i | x_{<i}))$. This corresponds to the characteristic of high compression rate (close to entropy) of arithmetic coding [45]. On the other hand, it ensures high imperceptibility as each pixel is determined adaptively on the basis of $q(x_i | x_{<i})$. Distortion between the equivalent distribution $q(x_i | x_{<i})$ and the original distribution $p(x_i | x_{<i})$ is slight enough. If the impact of truncation, quantization and blocking of secret message is ignored ($prc$ tends to infinity), the stego-message $m$ has infinite length), $q(x_i | x_{<i})$ is actually identical to $p(x_i | x_{<i})$ and it is able to achieve the optimal imperceptibility. Arithmetic coding is widely applied in generative linguistic and audio steganography due to its advantage in embedding capacity and imperceptibility [34], [43], [46]. But as far as we know, there hasn’t been any research about generative image steganography with arithmetic-coding-based stegosampling.

III. EXPERIMENTS

A. Setups
We conducted experiments on MNIST [47], Frey Faces and CIFAR-10 [48]. Details of the datasets are shown in Table I. We followed the original settings of PixelCNN++ in [49]. For CIFAR-10, we utilized a public Pytorch version pre-trained model. For MNIST and Frey Faces, we trained the model from scratch and took the parameters with best test loss during 200-epoch training with batch size 16 and learning rate

2http://www.cs.nyu.edu/~roweis/data/frey_rawface.mat

3https://github.com/pclucas14/pixel-cnn-pp
B. Embedding Capacity

In Table II, we list the result of embedding capacity measured in embedding rate (ER). As we can see, autoregressive-model-based steganographic methods have much higher embedding capacity than other generative methods. Compared with method in [40], the result of Pixel-Stega on MNIST does not seem to show too much superiority. As discussed above, the amount of information that can be embedded at each time step for Pixel-Stega is actually affected by the entropy of pixels $H(p(x_i|x_{<i}))$. Images in MNIST have similar pattern of black background and white handwritten digits in the middle, which results in low entropy for most pixels. We visualize the entropy $H(p(x_i|x_{<i}))$ as well as the amount of embedded information of each pixel in Figure 5. We found that the uncertainty is mainly provided by pixels at the edge of the strokes. For most pixels in the background or inside the stroke, they have zero entropy and cannot contribute to any space to conceal secret messages.

On the contrary, results on Frey Faces and CIFAR-10 illustrate that Pixel-Stega has a prominent advantage in embedding capacity than method in [40]. Images in these datasets are more diverse, which means their pixels have large entropy. The results reveal that Pixel-Stega is able to make the full use of $H(p(x_i|x_{<i}))$ to its potential. However, method in [40] only encodes secret messages to the LSB of pixels, which is distinctly a waste of the amount of information that $p(x_i|x_{<i})$ supplies (only 1 bit information is used).

C. Imperceptibility

We measured the distortion between the equivalent distribution $q(x_i|x_{<i})$ and the original conditional distribution $p(x_i|x_{<i})$ by Kullback-Leibler divergence (KLD) and Jensen–Shannon divergence (JSD).

On the contrary, results on Frey Faces and CIFAR-10 illustrate that Pixel-Stega has a prominent advantage in embedding capacity than method in [40]. Images in these datasets are more diverse, which means their pixels have large entropy. The results reveal that Pixel-Stega is able to make the full use of $H(p(x_i|x_{<i}))$ to its potential. However, method in [40] only encodes secret messages to the LSB of pixels, which is distinctly a waste of the amount of information that $p(x_i|x_{<i})$ supplies (only 1 bit information is used).

C. Imperceptibility

We measured the distortion between the equivalent distribution $q(x_i|x_{<i})$ and the original conditional distribution $p(x_i|x_{<i})$ by Kullback-Leibler divergence (KLD) and Jensen–Shannon divergence (JSD).

On the contrary, results on Frey Faces and CIFAR-10 illustrate that Pixel-Stega has a prominent advantage in embedding capacity than method in [40]. Images in these datasets are more diverse, which means their pixels have large entropy. The results reveal that Pixel-Stega is able to make the full use of $H(p(x_i|x_{<i}))$ to its potential. However, method in [40] only encodes secret messages to the LSB of pixels, which is distinctly a waste of the amount of information that $p(x_i|x_{<i})$ supplies (only 1 bit information is used).

C. Imperceptibility

We measured the distortion between the equivalent distribution $q(x_i|x_{<i})$ and the original conditional distribution $p(x_i|x_{<i})$ by Kullback-Leibler divergence (KLD) and Jensen–Shannon divergence (JSD).
According to the results listed in Table IV, the distortion between $p$ and $q$ of method in [40] can be large, especially on MNIST. As discussed, due to the simplex pattern of images in MNIST, the probability mass of most pixels almost concentrates on a single value thus leading to an extremely sharp distribution. Under such circumstances, using rejection sampling strategy to embed secret messages into the LSB of pixels is not reasonable. As a result, it fails to generate realistic stego-images (Figure 2). In contrast, Pixel-Stega has more negligible distortion, which provides evidence that Pixel-Stega is adaptive to all kinds of distribution $p(x_i|\mathbf{x}^c_i)$, no matter sharp or flat.

| DATASET | MODEL | KLD ↓ | JSD ↓ |
|---------|-------|-------|-------|
| MNIST   | [40]  | 1.2480E+01 | 1.6928E-01 |
|         | Pixel-Stega | 6.2003E-06 | 2.0988E-06 |
| Frey Faces | [40]  | 2.0547E-02 | 3.0439E-03 |
|         | Pixel-Stega | 3.3624E-06 | 9.6097E-07 |
| CIFAR-10 | [40]  | 1.2132E-02 | 2.3720E-03 |
|         | Pixel-Stega | 1.2151E-05 | 3.9509E-06 |

D. Comprehensive Assessment

To measure the performance of Pixel-Stega comprehensively, we calculated effective embedding rate (EER) [54] with the corresponding accuracy result of steganalysis method in Table III. The result reveals that Pixel-Stega achieves higher comprehensive performance than method in [40]. Besides, since EER is always less than or equal to ER, Pixel-Stega also surpasses other previous generative image steganographic methods. According to Figure 5, Pixel-Stega embeds more information to pixels with larger entropy and vice versa, thereby both ensuring high embedding capacity and nearly perfect imperceptibility, so as to achieve satisfying comprehensive performance.

| DATASET | MODEL | EER$_1$ ↑ | EER$_2$ ↑ | EER$_3$ ↑ |
|---------|-------|-----------|-----------|-----------|
| MNIST   | [40]  | 0.0000    | 0.0000    | 0.0000    |
|         | Pixel-Stega | 0.5515    | 0.5589    | 0.5813    |
| Frey Faces | [40]  | 0.9684    | 0.9740    | 0.9740    |
|         | Pixel-Stega | 3.9824    | 3.8399    | 3.6019    |
| CIFAR-10 | [40]  | 0.9344    | 0.8856    | 0.9354    |
|         | Pixel-Stega | 4.0500    | 3.8196    | 3.7524    |

To further evaluate the imperceptibility, we tested the anti-steganalysis ability by three state-of-the-art steganalysis approaches, XuNet [51], Yedroudj-Net [52] and SRNet [53]. Small adjustments were made to let the networks fit the shape of stego-images. We used 10-fold cross-validation to distinguish 5,000 cover-images and 5,000 stego-images. Each model was trained for 50 epochs with batchsize 64 and learning rate 0.01.

According to the results in Table III, we found Pixel-Stega achieves nearly perfect imperceptibility (closer to 0.5 is better), which further indicates its adaptability to all kinds of output distribution $p(x_i|\mathbf{x}^c_i)$. Despite on MNIST, method in [40] seems also qualified to resist steganalysis. However, it exposes secret messages directly to the monitoring party (the LSB of pixels is actually the secret message). On the contrary, Pixel-Stega maintains a higher level of security, as secret messages are really and truly hidden inside the stego-images. No meaningful information can be extracted even if some abnormality is noticed.

IV. Conclusion

In this letter, we proposed Pixel-Stega, a generative image steganographic method based on autoregressive models and arithmetic coding. We found Pixel-Stega is able to embed secret messages adaptively on the basis of the entropy $H(p(x_i|\mathbf{x}^c_i))$ of pixels. More information is embedded into high entropy region and less into low entropy region, which means (1) the explicit distribution $p(x_i|\mathbf{x}^c_i)$ is used to its full potential; (2) the stegosampling strategy is adaptive to either sharp or flat distributions. As a result, Pixel-Stega achieved both high embedding capacity (up to 4.3 bpp) and nearly perfect imperceptibility (about 50% detection accuracy) at the same time. It is hoped that this letter can facilitate the development of generative image steganography.

REFERENCES

[1] G. Greenwald, "Xkeyscore: Nsa tool collects’ nearly everything a user does on the internet.,” The Guardian, vol. 31, p. 2013, 2013. 1

[2] S. Landau, “Making sense from snowden: What’s significant in the nsa surveillance revelations,” IEEE Security & Privacy, vol. 11, no. 4, pp. 54–63, 2013. 1
[53] M. Boroumand, M. Chen, and J. Fridrich, “Deep residual network for steganalysis of digital images,” *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 5, pp. 1181–1193, 2018.

[54] S. Zhang, Z. Yang, J. Yang, and Y. Huang, “Provably secure generative linguistic steganography,” *arXiv preprint arXiv:2106.02011*, 2021.