Reducing Blocking Artifacts in CNN-Based Image Steganography by Additional Loss Functions

Tuan Dung Pham
HMI lab
VNU University of Engineering and Technology
Hanoi, Vietnam

Viet Cuong Ta
HMI lab
VNU University of Engineering and Technology
Hanoi, Vietnam

Thi Thanh Thuy Pham
Faculty of Information Security
Academy of People Security
Hanoi, Vietnam

Thanh Ha Le
HMI lab
VNU University of Engineering and Technology
Hanoi, Vietnam

***Abstract*—Our work improves the encoded image quality from HiDDeN framework, an end-to-end image steganography based on deep convolution neural network. In the encoding phase of HiDDeN framework, to embed a message in a cover image, it is required to split the cover image into smaller image blocks and embed the message bits in each block in parallel. These embedded blocks are then combined to form an encoded image that has the same size as the cover image. This image reconstruction process causes artifacts that appear on the boundaries of the blocks. This can be explained by the fact that when message bits are embedded in the image blocks, the pixel-level information of each image block is unequally alternated. In order to reduce block artifacts, in this work we propose a blocking loss as an additional objective function in HiDDeN framework. This loss measures the difference between encoded images and modified versions of the cover images. The proposed method is evaluated on COCO 2014 and BOSS datasets and the experimental results show the effectiveness in reducing the block artifacts that appeared in the encoded images of HiDDeN framework. This has an important impact on increasing the invisibility or transparency of the steganography system. In addition, the experimental result on secrecy of the proposed method also indicates the same performance as the HiDDeN pipeline.**

***Index Terms*—image steganography, convolution network, blocking artifact**

I. Introduction

Image-based steganography is mainly used in such tasks as watermarking or information hiding. The main purpose is to embed a hidden message, which is present in the form of a sequence of bits, into a cover image. The result of this procedure is an encoded image or stego image, from which the original sequence of bits can be extracted by the decoding process. Normally, the encoded image should be viewed as similar as possible to the cover image by the human vision system. This is equivalent to a metric of invisibility of an image-based steganography system. The difference between the cover image and encoded image is measured by PSNR (peak signal to noise rate). In addition to invisibility, several metrics are used to evaluate the efficiency of image-based steganography. The capacity presents how many bits of message can be hidden in the cover image. The robustness shows how altered a hidden message is when the stego image is distorted. The secrecy presents the rate of hidden message can be recovered from the encoded image.

Some popular approaches for image steganography including Least Significant Bit (LSB) [1], HUGO [2], or S-UNIWARD [3]. These approaches evolve around the idea of replacing unnecessary bits in the cover image by the bits of the message needed to be hidden. This modification is done in such a way that it can not make the visual difference between cover image and encoded image. In the aspect of security, in order to detect one image contains hidden message or not, there are a number of ways. For example, the work in [4] uses image features to detect the encoded image with the assumption that it has access to the encoding model.

Recently, with the advance of deep networks and generative models, CNN-based image steganography can be applied and achieved prominent results. Neural networks have been used for both steganography and digital watermarking such as the work of C.Jin et al [5], Kandi et al [6] and Mun et al [7].

The HiDDeN framework [9] shows an end-to-end trainable deep network that can be focused on both secrecy and robustness. Its main idea is to split a binary sequence of the message unequally and conceal them to each pixel of the cover image. In order to do this, the proposed pipeline uses deep convolution networks as an encoder and decoder. The encoder receives an image as the cover image and a message and creates the encoded image, which contains the hidden message. The decoder would take the encoded image and the initial image to extract the hidden message. In the training phase, the HiDDeN aims to optimize several metrics. The first one is the differences between the encoded image and the input image. The second one is the difference between the decoded message and the input message. The final one is a GAN-based loss [8], for detecting real or fake images.

Compare to the previous works, such as HUGO [2] or S-UNIWARD [3], HiDDeN outperforms both of these in terms of robustness and secrecy. However, to reach a reasonable capacity, the HiDDeN pipeline requires dividing the cover image into small blocks and encode each block individually. When combining these blocks to the original size of the cover image, the blocking artifacts as the vertical and horizontal edges between each consecutive pair of blocks are created. The effects would be similar to blocking artifacts of JPEG as described in [15]. This shows a visible change of the encoded image in comparison with the cover image. Therefore, the invisibility or imperceptibility of image-based steganography would be violated.

In order to reduce the block artifacts caused by HiDDeN framework, in this paper we add another loss called blocking loss to this framework. The proposed loss attempts to reduce the differences between the encoded image and a modified version of the cover image. Moreover, an entropy weight is added to put more emphasis on the region where the network should be trained to prevent the mismatch of the boundaries of the encoded consecutive blocks. The proposed method is tested on the datasets of COCO 2014 [16] and BOSS [19]. Our work has shown better invisibility but retains the secrecy metric of HiDDeN framework.

The remains of our papers are organized as follows: in the section II, we introduce several related works to image steganography, the HiDDeN framework and its block artifacts issue are presented in section III, our proposed solution is introduced in section IV, we provide the experiment results and discussion in section V. The summary is given in section VI.

II. Related Works

The pipeline framework of an image steganography system may consist of two inputs: a cover image and a message. The message is concealed in the cover image by embedding algorithm to output encoded image. In a reverse process, the hidden message can be extracted from the encoded image. The difficulty or ease of recovering hidden message from encoded image by steganalyzers is evaluated in security criteria.

Least-Significant Bit (LSB) [1] starts with the idea of replacing the rightmost bit in a binary number string of a pixel in cover image by a bit of a message so that this replacement is transparent to human vision system. By design, LSB has an asymmetry embedding operation and altered statistics which is a potential vulnerability to the development of highly accurate targeted steganalyzers lead to reliable detection. More advanced techniques based on LSB such as LSB matching [10] was created to overcome this weakness by modulating the pixel value by 1 so that the LSBs of pixels match the secret message.

Besides LSB-based image steganography, recent algorithms attempt to optimize a distortion metric when a message is embedded into the cover image. Several notable works including HUGO [2], which relies on means of efficient coding algorithm and S-UNIWARD [3] with using the functions on the spatial domain from a bank of directional high-pass filters.

The recent advances of deep convolution neural network (CNN) allows the image encoder and decoder in the image steganography could be built easily. The encoder structures includes VGG-base structures [11] or ResNet structures with residual connection [12]. In the decoder part, generative neural networks such as Variational autoencoders (VAEs) [13] or generative adversarial nets (GANs) [8] achieve good performances in a wide range of image generation tasks.

Prior to HiDDeN [9], CNN network approaches for image steganography is used on several parts of a large pipeline. For example, [5] divide original image in to blocks then uses neural network to measure secrecy of each block. In the first step, the original image is divided into blocks, and then neural networks decide adaptive different embedding strengths for each block, depend on their textural features and luminaries. In the second step, the watermark detection is based on the correlation of different keys. [6] propose the use of an auto-encoder base on CNN and [7] to construct a CNN-decoder for decoding the message bit. The deep CNN network could also be used in detecting the encoded images. SRNet [14] proposes the usage of a deep residual architecture which removes the pooling step in the early step to preserve the stego signal in images.

III. HiDDeN Framework and Block Artifacts

A. *Overview of HiDDEN Framework*

The main idea of HiDDeN [9] is to use an encoder to hide a binary bits sequence of a secret message in spatial regions of the cover image. This results into a encoded image or stego image from which the hidden message could be retrieved by a parallel-reverted decoder. In the encoding phase, the encoder E receives a cover image I_{co} of shape C x H x W and a binary secret message $M_{in} \in \{0, 1\}$ of length m and produces an encoded image I_{en} with the same shape as I_{co}. In the decoding phase, The noise layer N receives I_{en} and I_{co} as inputs and distorts the encoded image to produce a noised image I_{no}. The decoder D recovers the message M_{out} from I_{no}. For training the encoder E and decoder D, the authors proposed to use a large number of images I_{co} and randomly pick a binary message M_{in} to feed through the network. The losses are then computed on the image reconstruction loss L_I, which measures how differs between the cover image I_{co} and the encode image I_{en}, the decoded message error L_M between M_{in} and M_{out} and the GAN loss L_G of A. As reported in [9], the detection rate of encoded images by HiDDeN is 50% with a capacity of 0.2 bit per pixel. The detection result is better than HUGO [2] and S-UNIWARD [3], which have the detection rate of 70% and 68%, respectively.

In the proposed approach, to incorporate the message M of length m into the cover image I_{co}, the network architecture contains a message volume as an intermediary representation. The message volume is a layer which has the spatial size H x W as the image and the depth channel is m. Given a fixed bit per pixel (BPP) λ, the size of the message volume would increase linearly to the image size. Specifically, to reach the capacity λ of a gray cover image I_{co} with size of H x W, it requires the input message to have the length m = λ x H x W. For example, m could be over the size of 3200 depth channels with $H = W = 128$ and BPP capacity $\lambda = 0.2$. Therefore, it makes the training procedure impractical with a large image size. Moreover, the number of depth channels in intermediary representation could affect the performance of the network in the message decoding phase. In other ways, it is more difficult to retrieve the hidden message from a large message volume. To overcome the issues, the authors of HiDDeN proposed an approach to split the cover image and the message into smaller blocks and train the network in each smaller block, one-by-one.

Without the loss of generality, we could assume the input cover image I_{co} has the dimension of H = W = N. By selecting a block size KxK, and let $n = N/K$, the image I_{co} would be divided into n^2 image blocks $I_b^{i,j}$ with size KxK ($0 \leq i,j < n$). As a result, given the same BPP λ, the encoding message length m is fixed by K, rather than N. Therefore, it also removes the linear relation between the size of the intermediary representation and the input cover image size. However, by splitting a cover image I_{co} into n^2 smaller blocks of $I_b^{i,j}$, encoding the message into each block and reassembling these encoded blocks to form an encoded image I_{en} create block artifacts in I_{en}. This phenomenon is similar to lossy image compression of JPEG. It is a noticeable distortion of the encoded images in the HiDDen steganography system. This means the invisibility metric of image-based steganography is not satisfied. Fig 1 indicates an example of block artifacts (horizontal and vertical lines or edges) that appeared on the encoded image (the middle one) compared to the normal display of the cover image (the left one). The standard encoder of HiDDeN works on (16x16) block size ($K = 16$) of 128x128 cover image. The embedding capacity $\lambda = 0.2$ results in a message size that can be hidden in each block is $m = 52$ bits . The peak signal to noise (PSNR) for image, computed by $|I_{co} - I_{en}|$, shows the differences emphasized by vertical and horizontal edges along with the blocks of size K x K.

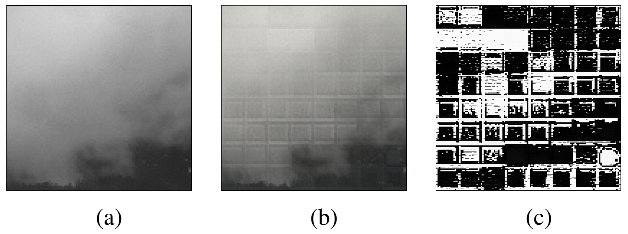

Fig. 1: From left to right, the cover image (a), the encoded image (b) and its peak signal to noise image. The white pixel in signal to noise image indicates large differences in pixel values between the encode image and the cover image.

In the following sections, we present the proposed methods to calculate block artifacts and reduce block artifacts in encoded images resulting from HiDDeN system.

B. Block Artifact Calculation

In order to measure the influence of block artifacts, we can compute the difference between the pixels of two neighboring blocks. The blocking values of each image could be computed similarly to the blocking effects of JPEG. The work in [15] defines an edge as a boundary between two regions with relatively distinct gray-level properties. We adapt this definition for computing the block artifacts of the full size NxN encoded image, which are reassembled from n^2 images with the size of KxK.

For an vertical edge at position $x = j * K$ with $0 \leq j < n$, it splits the cover image into two consecutive blocks. Fig 2 illustrates the edges which are constructed from image blocks of size KxK within the cover image of size NxN (N=$n * K$). The total number of vertical edges in an image is $n - 1$. Let w is the edge width of each side, we could forms a pair of image regions R_j^{left} and R_j^{right}. Each region R_j has a dimension of $1 \times w$ and each pixel i in this region has a gray scale value p_i with $0 \leq i < w$. The gray level P over a region R_j is calculated by the average value $P(R_j) = (\sum p_i)/w$. Therefore, the distinct gray level D_j between two average gray value of R_j^{left} and R_j^{right} is defined as the absolute value of the subtraction between the two: $D_j = \left|P(R_j^{left}) - P(R_j^{right})\right|$.

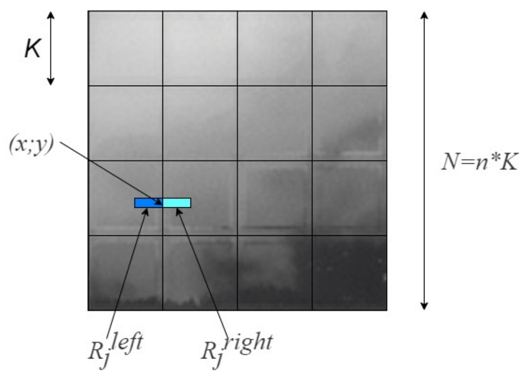

Fig. 2: Illustration of the region pair's coordinate in the spatial domain.

With a given constant G value, we count the number of distinct gray level values D_j that are greater than G in vertical direction V_{count}. Similarly, we could compute the number of distinct gray level values in horizontal direction H_{count}. The block artifact value B of one image is then calculated as the fraction of the sum over total number of region pairs:

$$B = \frac{H_{count} + V_{count}}{2 * n * (n-1)} \tag{1}$$

By using the above equation, the encoded image in Fig. 1 has a block value of 0.1093 with $G = 1/8$ of maximum gray level.

The mismatch issue between two consecutive image blocks is similar to JPEG [15] or other pipelines which require

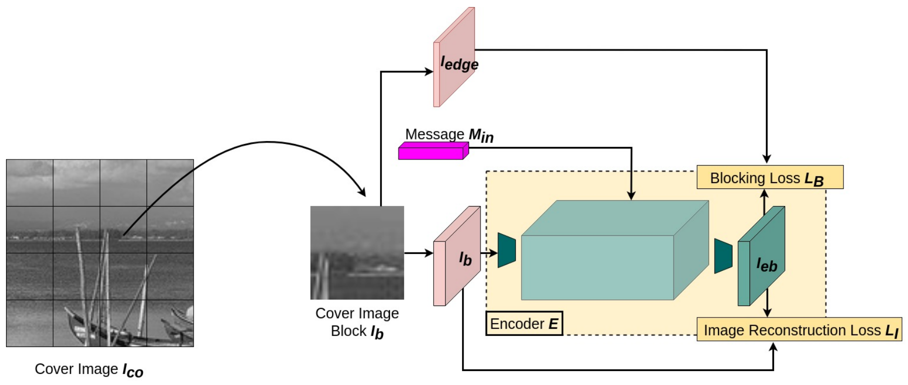

Fig. 3: Loss variables of original HiDDeN framework and our Loss variables. Original HiDDeN loss function L_I based on Cover image block I_b and Encoded Image block I_{eb}, while we calculate our loss L_B on Edge image block I_{edge} and Encoded Image block I_{eb}.

splitting the original image to smaller sizes. There are several methods that attempt to fix the artifacts. Novel methods such as adaptive filtering [15] or wavelet transform [17] are proposed to reduce the artifacts. These methods can be applied for any block-based image encoders. In [18], the authors proposed a BlockCNN architecture which reads the image blocks consecutively and fixes the mismatch boundaries.

IV. Training Additional Loss for Reducing Blocking Artifacts

Given the block artifacts value B from Equation 1, one could set up a training schedule between two sharing-edge blocks of the cover image and user the derivative of B to train. However, it requires to modify the encoder to work on two blocks of the cover image instead of one. This would add more complexity to the message encoding and decoding at the later phases. Instead, we create a modified version of the cover image named the edge image I_{edge}. The edge image I_{edge} is produced by replacing the pixels in the boundary regions of the original cover image with the pixels of the next image blocks. We modify the HiDDeN training procedure by adding a blocking loss L_B in HiDDeN framework (Fig 3). The main purpose of the new loss L_B is to ask the encoder to produce the output encoded image similar to the I_{edge}, which reduces the differences between the pixels of those boundaries. Therefore, it would reduce the effects of blocking artifacts in the later process of block reassembling.

A. The Modified Cover Image

The edge image I_{edge} can be created from the I_{co} by blending the pixels in the boundary region of one image block with the corresponding pixels in the boundary region of its consecutive block, given the block size K. We propose a way to construct the I_{edge} by keeping the pixels in the center the same as the initial I_{co}. In the edge boundary regions, there are two factors of α and β that are used to control the difference between the I_{edge} and the I_{co}. The α parameter is used to control the blending factor of the pixel's gray value and the β parameter defines the width of the edge boundary.

In general, the cover image I_{co} of size NxN is split into smaller blocks $I_b^{i,j}$ of size KxK. Fig 4 illustrates the building of the right boundary region of the $I_b^{i,j}$, using α and β parameters.

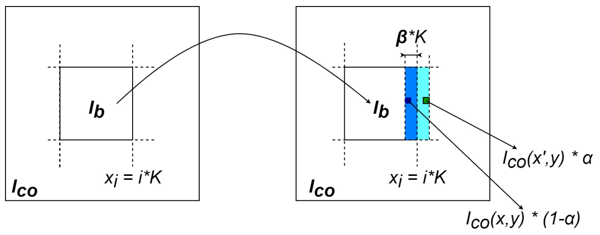

Fig. 4: Given a block image I_b with length K defined by vertical edge $x_i = iK$, the left boundary region of I_b has the width of βK and its gray level values adjusted by the boundary region of the consecutive block image, using the blending factor α.

The new value of a pixel in the block I_b of the edge image is calculated by linear interpolate respect to the α value.

In general, any block has a maximum number of 4 boundary regions depends on its position: right-vertical, left-vertical, top-horizontal and bottom-horizontal region. We could formulate the gray level value of $I_{edge}(x, y)$ for a right-vertical region with its corresponding edge $x_i = i*K$ as follow:

$$I_{edge}(x, y) = I_{co}(x, y) * (1 - \alpha) + I_{co}(x', y) * \alpha \tag{2}$$

with $x_i - \beta * K \leq x \leq x_i$, $x' = 2x_i - x$. The pixel (x', y) is the pixel in the left-vertical boundary region of the consecutive block $I_b^{i+1,j}$ of $I_b^{i,j}$. The equation for the right-vertical edge could be applied to create other left-vertical, top-horizontal, bottom-horizontal edges. The I_{edge} will then have the boundary of each K x K block images modified by the above procedure. Therefore, the difference at the boundary regions of block size K could be learned by adding a new loss.

Given the edge image, we create a new loss, L_B to measure how I_{en} is different from I_{edge}. More specifically, the new loss L_B asks the network to produce the image with smaller differences in the edge boundary regions for each vertical edge $x_i = iK$ and $y_j = jK$. In our work, we select to optimize the mean square error (MSE) loss of the encoded image I_{en} and I_{edge}. The L_B would be optimized together with the other three losses, L_I, L_M and L_G of HiDDeN pipeline. Because the I_{edge} contains the information of the edge boundary regions, by optimizing the L_B, it would reduce the block artifacts. In general, there is a correlation between the I_{edge} and I_{co}, which affects the L_I and L_B in the training. The factor α and β would control the variations between the two losses. Our later experiments show that the adding loss L_B would not affect the convergence of the training procedure.

B. Adding Weight to MSE Loss based on Image Entropy

In the standard HiDDeN network, the entropy of an image could affect the encoding and decoding hidden messages. It is easier to embed/extract the hidden message into/from a low entropy region than a high entropy region. In other words, the low entropy region leaves more way to alternate the sub-pixel level structures for hiding the message. However, in the low entropy region, the encoded image is easier to discover. Given the two image blocks which share a boundary $x_i = iK$ or $y_j = jK$, the entropy of each block could affect the way its pixels be modified. It would lead to the block artifacts if the differences are larger than a threshold to create edge-structures. In our works, the threshold is defined by the constant G. The Fig. 5 illustrates our idea. There exists a correlation between the entropy of the input cover image and the output block values of the corresponding output encoded image. When the input cover image I_{co} has high entropy, the output or the encoded image I_{en} suffers more block effects.

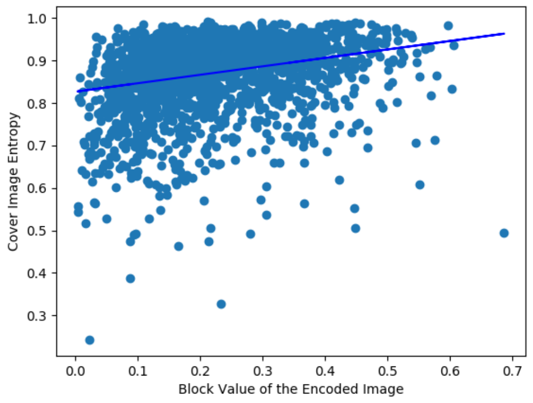

Fig. 5: Correlation between the block values of the encoded images and the entropy of the input cover images of size 128x128, calculating on 1000 gray-scale images.

Based on this observation, we add the entropy of the input cover image as a weight of the standard MSE loss into the computation of L_B. The purpose of this weight is to emphasize the loss between the encoded image I_{en} and the edge image I_{edge} when the image entropy value is high. Therefore, it would reduce the block artifacts.

TABLE I: The mean blocking values in the test images for three approaches on COCO and BOSS dataset

Dataset	Standard	MSE	Entropy-MSE
COCO 2014	0.0949	0.0902	**0.0851**
BOSS	0.0307	0.0254	**0.0233**

V. Experiments and Results

For testing our approach, we use images from COCO [16] and BOSS datasets [19]. In COCO dataset, we randomly chose 10000 images for training and 2000 images for testing. For BOSS dataset, we select 2000 images for training and 500 images for testing. Each image is preprocessed by cropping into 128x128 and converted into gray level. Image block size is selected at K = 16, resulting 64 blocks in each image. With each image block, the encoded message length is $m = 52$, which results in an embedding capacity of 0.203, closed to 0.2 bit per pixel. For a full-size 128x128 image, the length of message to be hidden in testing images is 52x64 bits in total.

The encoder and decoder are set up by using default configs of HiDDeN [9]. The optimizer is Adam with learning rate at 1e-5. The batch size is 32. We train the network with 50 epochs.

In the first experiment, the edge images for computing the loss L_B are created with the parameters of $\alpha = 1.$ and $\beta = 0.2$. Given the block size of $K = 16$, the config results in an edge width $w = 0.2 * 16 \sim 3$ pixels.

For computing block value, the distinct gray level G in Eq. 1 is set at $G = 1/8$ of maximum gray level. The block values are presented in Table I. We report the results on three configs: the **Standard** shows the mean block values calculated from encoded images by the HiDDeN pipeline in [9], the **MSE** and the **Entropy-MSE are two versions of our proposed method with the additional loss training to match the edge images**.

As we can see, both **MSE** and **Entropy-MSE** have lower block values than **Standard**. This results from the fact that two loss metrics are proposed to optimize block values. On the COCO dataset, the relative improvement of our approach is 5% and 10%, with **MSE** and **Entropy-MSE** respectively. On the BOSS dataset, the relative improvement is increased to 17% and 24%, with **MSE** and **Entropy-MSE** respectively. The **Entropy-MSE** have a minor improvement than the raw **MSE** loss.

An example of the output images from COCO dataset is illustrated in Fig 6. Both the proposed **MSE** and **Entropy-MSE** methods can reduce the artifacts in the encoded images to some degrees. In terms of visual effects, our proposed methods result in higher quality than the standard approach of HiDDeN.

We also evaluate our model in other metrics of secrecy. At epoch 50, **MSE** and **Entropy-MSE** has the bit decode errors

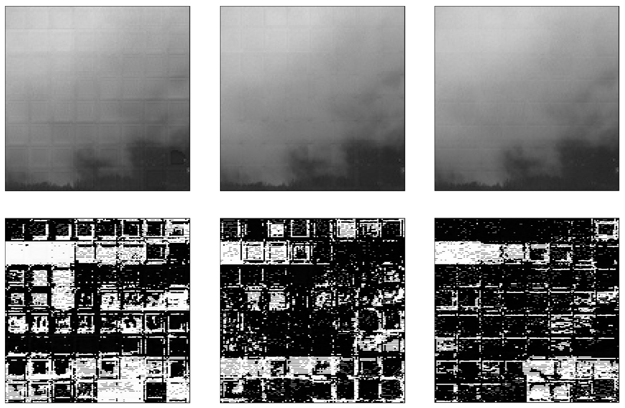
(a) Standard (b) MSE (c) Entropy-MSE

Fig. 6: The first row are the encoded images and the second row are the PSNR images of each configs, with the same input cover image and the same input hidden message.

TABLE II: The mean blocking values in the test images on COCO dataset on different values of α and β with **Entropy-MSE** loss

α	β	**Entropy-MSE**
$\alpha = 1.0$	$\beta = 0.1$	0.0837
$\alpha = 1.0$	$\beta = 0.2$	0.0851
$\alpha = 1.0$	$\beta = 0.3$	0.0853
$\alpha = 0.5$	$\beta = 0.1$	0.0828
$\alpha = 0.5$	$\beta = 0.2$	0.0841
$\alpha = 0.5$	$\beta = 0.3$	0.0856

under 1e-5, similar in the report of **Standard** approach [9]. For secrecy, by using the methods of [20] with embedding capacity 0.2 and Discrete Cosine Transform Residual as the feature extractor, it results in a detection rate of 50% for the three configs. This indicates that adding loss functions does not affect the secrecy of the pipeline.

In order to measure the effects of two parameters of α and β on the performance of **Entropy-MSE** methods, the results of block values with different choices of α and β are reported in Table II. From Table II, we can see that the values of α and β have effects on the block values. The lowest blocking value is at 0.0828, which can be reached by selecting $\alpha = 0.5$ and $\beta = 0.1$.

VI. Conclusion

In this work, we study the block artifacts which come from the process of encoding a long message into the cover image. We introduce a blocking loss for reducing block artifacts appeared on encoded grayscale images of HiDDeN method. The proposed loss functions are computed on the modified version of the cover images. An additional constraint is added to these loss functions to allow the networks to mimic the cover image itself on the boundary regions of block images. We tested our approach on two alternative versions, mean square loss and entropy-weighted mean square loss. Both versions are tested on the COCO dataset and BOSS dataset and compared with the baseline approach of HiDDeN. The visual quality of output encoded images are improved compared to the standard HiDDeN's output stego images.

Acknowledgements

This research has been supported by VNU University of Engineering and Technology under project number CN20.24.

References

[1] Gupta, S., Goyal, A., and Bhushan, B. (2012). Information hiding using least significant bit steganography and cryptography. International Journal of Modern Education and Computer Science, 4(6), 27.

[2] Pevný, T., Filler, T., and Bas, P. (2010, June). Using high-dimensional image models to perform highly undetectable steganography. In International Workshop on Information Hiding (pp. 161-177). Springer, Berlin, Heidelberg.

[3] Holub, V., Fridrich, J., Denemark, T.: Universal distortion function for steganography in an arbitrary domain. EURASIP Journal on Information Security 2014(1)

[4] Lerch-Hostalot, Daniel, and David Megías. "Unsupervised steganalysis based on artificial training sets." Engineering Applications of Artificial Intelligence 50 (2016): 45-59.

[5] Jin, C., Wang, S.: Applications of a neural network to estimate watermark embedding strength. In: Workshop on Image Analysis for Multimedia Interactive Services, IEEE (2007) 16 Zhu*, Kaplan*, Johnson and Fei-Fei

[6] Kandi, H., Mishra, D., Gorthi, S.R.S.: Exploring the learning capabilities of convolutional neural networks for robust image watermarking. Computers and Security (2017)

[7] Mun, S.M., Nam, S.H., Jang, H.U., Kim, D., Lee, H.K.: A robust blind watermarking using convolutional neural network. arXiv preprint arXiv:1704.03248 (2017)

[8] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., and Bengio, Y. (2014). Generative adversarial nets. In Advances in neural information processing systems (pp. 2672-2680)

[9] Zhu, J., Kaplan, R., Johnson, J., and Fei-Fei, L. (2018). Hidden: Hiding data with deep networks. In Proceedings of the European Conference on Computer Vision (ECCV) (pp. 657-672)

[10] Mielikainen, Jarno. "LSB matching revisited." IEEE signal processing letters 13.5 (2006): 285-287

[11] Karen Simonyan and Andrew Zisserman."Very Deep Convolutional Networks for Large-Scale Image Recognition". International Conference on Learning Representations, 2015.

[12] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[13] Pu, Yunchen, et al. "Variational autoencoder for deep learning of images, labels and captions." Advances in neural information processing systems. 2016.

[14] Boroumand, Mehdi, Mo Chen, and Jessica Fridrich. "Deep residual network for steganalysis of digital images." IEEE Transactions on Information Forensics and Security 14, no. 5 (2018): 1181-1193.

[15] Lee, Y. L., H. C. Kim, and HyunWook Park. "Blocking effect reduction of JPEG images by signal adaptive filtering." IEEE Transactions on Image Processing 7.2 (1998)

[16] Lin, Tsung-Yi, et al. "Microsoft coco: Common objects in context." European conference on computer vision. Springer, Cham, 2014.

[17] Kim, Nam Chul, et al. "Reduction of blocking artifact in block-coded images using wavelet transform." IEEE transactions on circuits and systems for video technology 8.3 (1998): 253-257.

[18] Maleki, D., Nadalian, S., Derakhshani, M. M., and Sadeghi, M. A. (2018, May). BlockCNN: A Deep Network for Artifact Removal and Image Compression. In CVPR Workshops (pp. 2555-2558).

[19] Bas, P., Filler, T., Pevn'y, T.: break our steganographic system: The ins and outs of organizing BOSS. In: International Workshop on Information Hiding, Springer (2011) 59–70

[20] Lerch-Hostalot, Daniel, and David Megías. "Unsupervised steganalysis based on artificial training sets." Engineering Applications of Artificial Intelligence 50 (2016): 45-59.