Supplementary Document: Large-capacity Image Steganography Based on Invertible Neural Networks

Shao-Ping Lu[1*] Rong Wang[1*] Tao Zhong[1] Paul L. Rosin[2]
[1]TKLNDST, CS, Nankai University, Tianjin, China
[2]School of Computer Science & Informatics, Cardiff University, UK
slu@nankai.edu.cn; nkwangrong@163.com; zei.t@qq.com; RosinPL@cardiff.ac.uk

We provide more experimental results in this supplementary document. Sec. 1 presents the ablation experiments on different numbers of invertible blocks or different structures of our sub-modules; Sec. 2 shows how losses's weights affect the construction quality of the container and revealed hidden images; Sec. 3 shows the exact hyper-parameters of our models; Sec. 4 shows more visual results on comparing our method against [1]'s method or the ground truths.

1. Submodule Selection

For the network structure, our experiments are carried out on different numbers of invertible blocks and models, where the Dense Block or Residual Block are employed as invertible block sub-modules.

Table 1. Ablation experiments for the network architecture.

Inv Blocks	sub-module	Container Image	Revealed Image
16	Dense Block	**39.48/.968**	37.26/.964
8	Dense Block	38.81/.965	38.65/.976
4	Dense Block	35.27/.965	**40.28/.980**
16	Residual Block	36.62/**.968**	38.46/.972
8	Residual Block	**37.44**/.953	39.17/.973
4	Residual Block	36.50/.956	**40.23/.977**

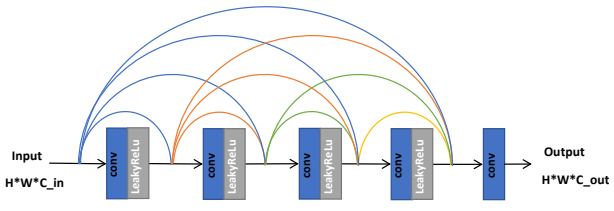


Figure 1. The architecture of our Dense Block sub-model.

*indicates equal contribution.

Table 2. Sub-module architectures.

Conv (input_channel, output_channel, size, stride, activation)
Conv2D (C_{in}, 32, 3, 1, LeakyReLU)
Conv2D (C_{in} + 32, 32, 3, 1, LeakyReLU)
Conv2D ($C_{in} + 32 \times 2$, 32, 3, 1, LeakyReLU)
Conv2D ($C_{in} + 32 \times 3$, 32, 3, 1, LeakyReLU)
Conv2D ($C_{in} + 32 \times 4$, C_{out}, 3, 1, None)

Let's take hiding an image as an example to discuss the network architecture by setting all the loss weights to 1. In Tab. 1, as the number of invertible blocks increases, the PSNR of the generated container image increases, but the PSNR of the revealed image decreases gradually. Easy to follow that increasing the loss weight of the container image, i.e. α_{co}, could improve the quality of the container image. Therefore, we choose to use 4 invertible blocks when hiding an image. It can also be observed from Tab. 1 that choosing Dense Block or Residual Block as sub-modules has slight influence on the results. We then adopt a 5-layer Dense Block (Tab. 2, Fig. 1) as our sub-module. For the sub-module $\phi(\cdot)$, C_{in} is the number of feature channels in the hidden branch b_2, and C_{out} is the number of feature channels in the host branch b_1. Here C_{out} is always set to 3. For sub-modules $\rho(\cdot)$ and $\eta(\cdot)$, the values of C_{in} and C_{out} are equal to C_{out} and C_{in} of the sub-module $\phi(\cdot)$, respectively.

2. Loss Function Adjustment

Table 3. Ablation experiments for loss weights.

$(\alpha_{co}, \alpha_{hi})$	Container Image	Revealed Image
(1, 1)	38.81/.965	38.65/.976
(2, 1)	**40.11/.974**	35.49/.950
(1, 2)	31.78/.949	**40.76/.980**
(2, 2)	36.44/.963	36.13/.965

Ablation experiments are also designed for different loss function weights. In addition to improving the container

image quality by adjusting the loss weights with 4 invertible blocks (as described in the paper), we also perform the experiments when using 8 invertible blocks. However, as shown in Tab. 3, increasing the loss weight α_{co} can indeed improve the PSNR of the container image, but the quality of revealed hidden image will significantly decline, making it difficult to make a good trade-off. Therefore, it is reasonable for us to use 4 invertible blocks when hiding an image.

3. Parameters Setting

The detailed parameters setting of ours models are shown in Tab. 4.

Table 4. The detailed parameters setting.

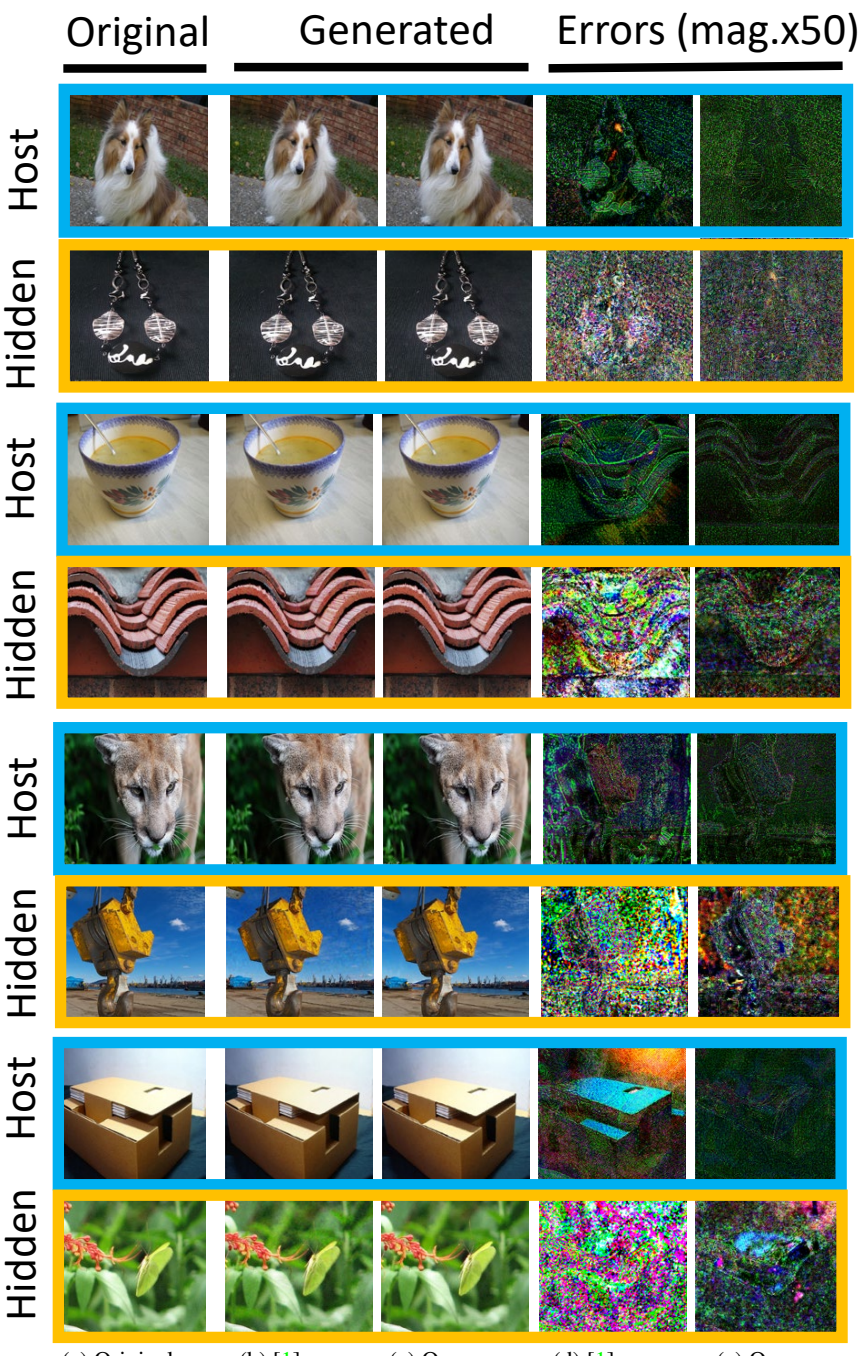
The number of hidden images	1	2	3	4	5
Channels of b_1	3	3	3	3	3
Channels of b_2	3	6	9	12	15
Inv Blocks	4	8	16	16	16
α_{co}	32	64	64	64	64

4. Comparisons

Here we add some visual comparisons. Fig. 2 shows that the container images and the revealed hidden images created by our method and [1]'s method are visually plausible. However, our results are with less reconstruction errors. What's more, as shown in Fig. 3, the method in [1] is prone to color bias in some cases when hiding two images, while both the container image and revealed hidden images generated from our method are with higher quality. Fig. 4, Fig. 5 and Fig. 6 show some examples where we hide $3 \sim 5$ images, respectively.

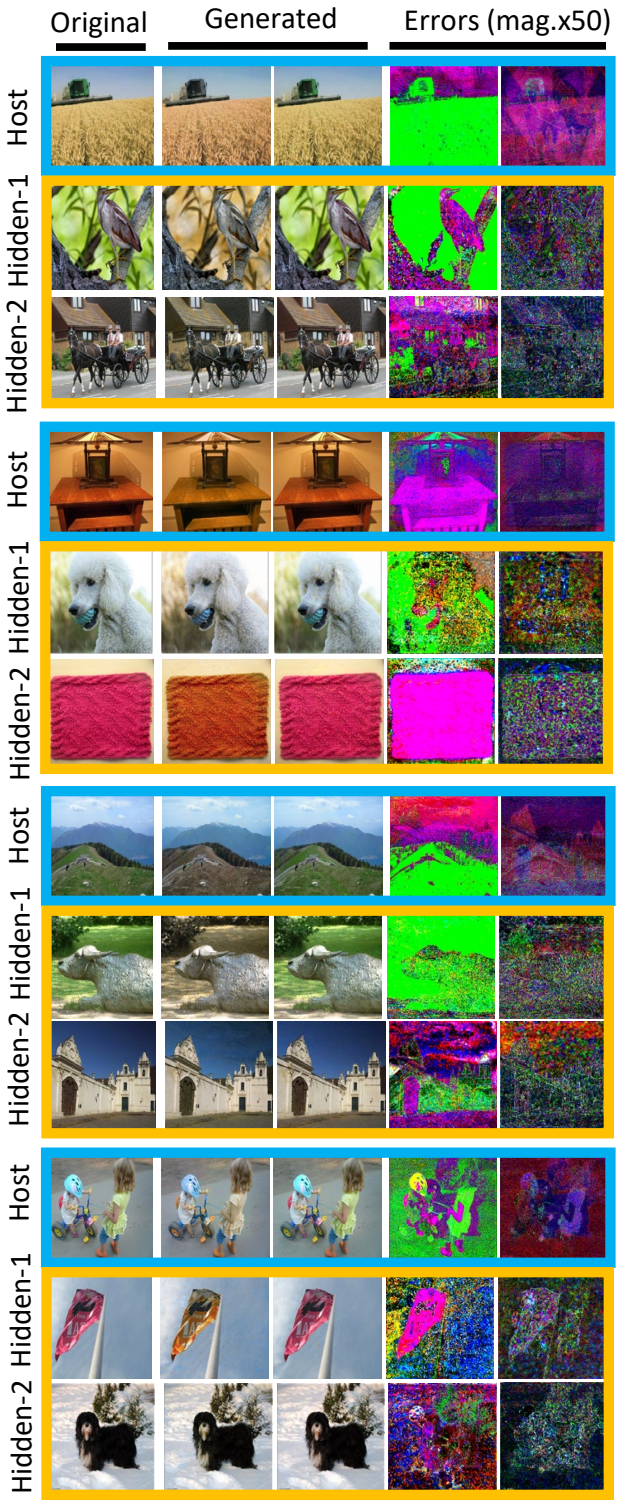
References

[1] Shumeet Baluja. Hiding images within images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2019. 1, 2, 3, 4



(a) Original (b) [1] (c) Ours (d) [1] (e) Ours

Figure 2. Visual comparison for hiding and revealing an image.



(a) Original (b) [1] (c) Ours (d) [1] (e) Ours

Figure 3. Visual comparison for hiding and revealing two images.

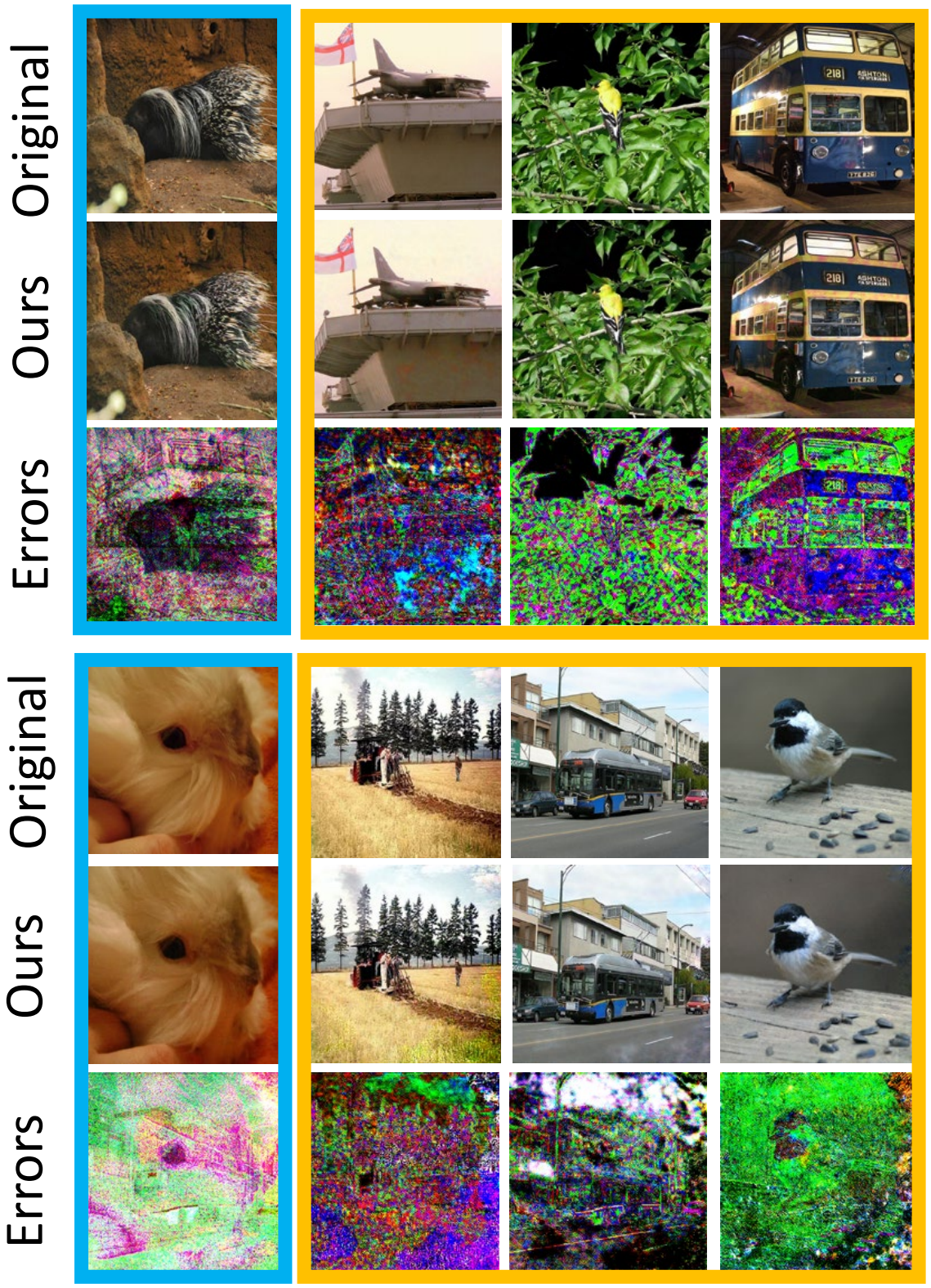


Figure 4. Two examples for hiding and revealing three images, with a blue border on the host images and an orange border on the hidden images. In each example, the top row is the original images and the middle row is our generated results, while the third row is the $\times 50$ magnified errors between them.

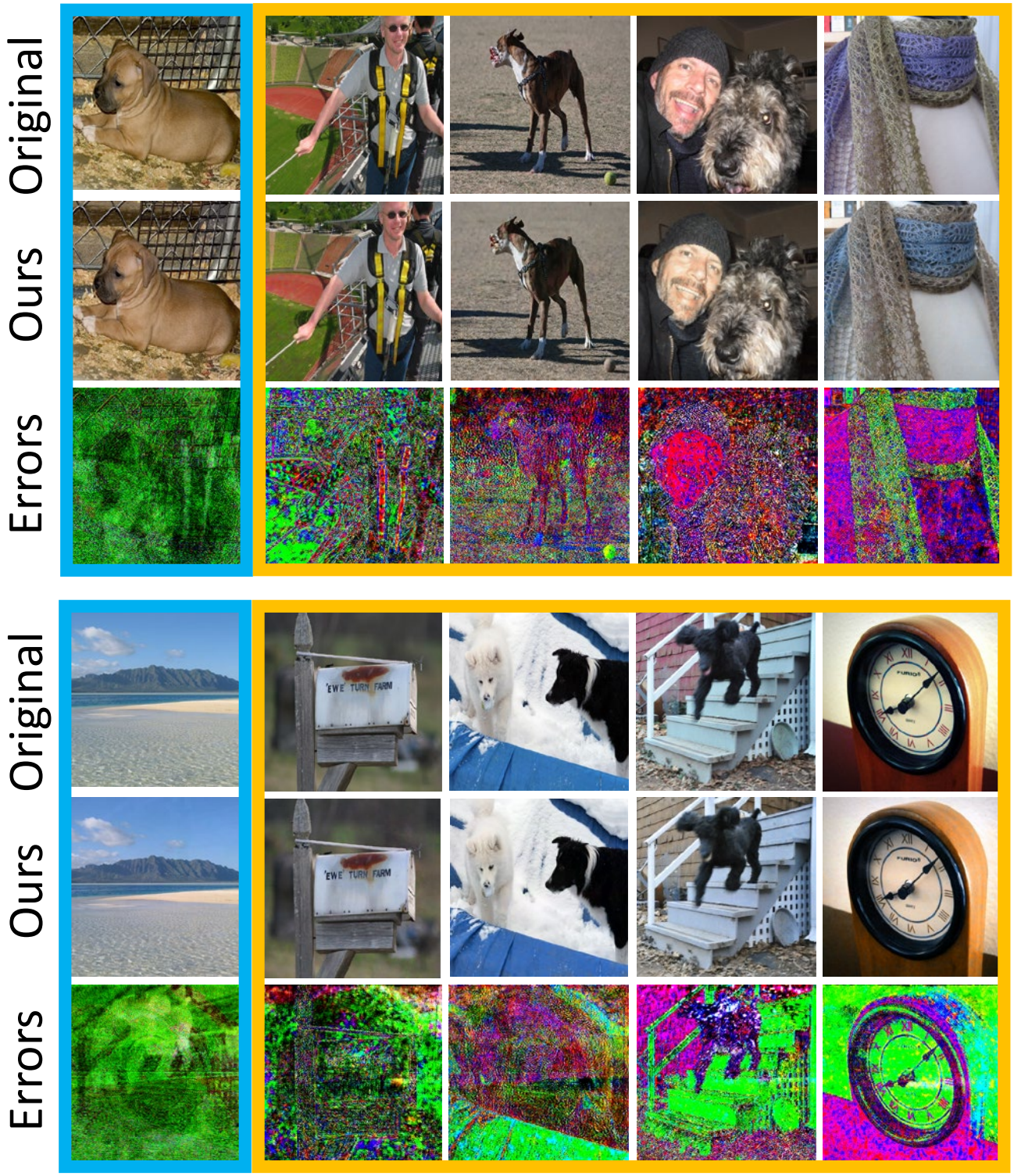


Figure 5. Two examples for hiding and revealing four images, with a blue border on the host images and an orange border on the hidden images. In each example, the top row is the original images and the middle row is our generated results, while the third row is the $\times 50$ magnified errors between them.

Figure 6. Two examples for hiding and revealing five images, with a blue border on the host images and an orange border on the hidden images. In each example, the top row is the original images and the middle row is our generated results, while the third row is the $\times 50$ magnified errors between them.