IICNet: A Generic Framework for Invertible Image Conversion Supplementary Materials

Ka Leong Cheng*, Yueqi Xie*, Qifeng Chen
The Hong Kong University of Science and Technology
{klchengad, yxieay}@connect.ust.hk, cqf@ust.hk

1. Experimental Details

Each experiment is trained for $50K$ mini-batch iterations using Adam [3] optimizer. We set the initial learning rate to be 2×10^{-4} and halve the learning rate for every $10K$ iterations. Generally, each experiment takes around $3 \sim 4$ days to train on a single RTX 2080 Ti GPU. In the remainder of this section, we show some detailed experimental setups for all the 5 RIC tasks discussed in our paper. Table 1 shows a summary of training hyper-parameters.

1.1. Spatial-Temporal Video Embedding

We use the high-quality DAVIS 2017 video dataset in this task, containing 150 video sequences of $480p$ resolution. The official TrainVal, Test-Dev, Test-Challenge set contains 90, 30, and 30 video samples. Our train set contains all the 90 samples in the official TrainVal set plus the 15 samples with odd indices (alphabetical order) in the official Test-Dev set; our validation set contains the 15 samples with even indices (alphabetical order) in the official Test-Dev set; our test set contains the 30 samples in the official Test-Challenge set.

For training, we subsample all the possible video subsamples with a time step of 5 between consecutive frames using the middle frame as the reference image. Here, we give a concrete example. The first subsample of a video with embedding range $N = 5$ contains frame 0, 5, 10, 15, 20 of the original video, and frame 10 is selected as the reference image. Each subsampled video sequence is randomly cropped with a resolution of 144×144 as input using a batch size of 2. For testing, without specifications, the statistics are reported by testing on all the possible video subsamples with a time step of 1 at full resolution.

Since we find it is likely that the PSNR and SSIM values reported in [10] are calculated using grayscale images, we also offer grayscale PSNR and SSIM comparison here for reference in Table 2.

*Joint first authors

1.2. Mononizing Binocular Images

The employed Flickr1024 dataset [7] contains $1,024$ binocular images of various categories. We follow the official train and test splits. For training, the input binocular images are randomly cropped with a resolution of 144×144 using a batch size of 2; for testing, we use the full resolution.

1.3. Embedding Dual-View Images

We train and test our method using the DIV2K dataset [1], which consists of 800 training samples and 100 validation samples, where each sample is composed of a $2K$ resolution image and corresponding $\times 2$, $\times 4$, $\times 8$ bicubic downsampled low-resolution images. We treat the first 750 official training samples as the train set, the remaining 50 samples as the validation set, and the 100 official validation samples as the test set. The batch size we use for training is 2. For each sample, we randomly crop the low-resolution image with a size of 144×144 as the input normal-view image and crop the high-resolution counter-party with the same size according to the center content of the input normal-view image as the input zoomed-view image, where the normal-view image is the reference image. For testing, we test using full resolution inputs.

1.4. Composition and Decomposition

The Adobe Matting dataset includes $49,300$ training samples and $1,000$ testing samples. Each sample comprises a foreground image, a background image, an alpha matting for the foreground image, and a composed image. The Real Matting dataset contains 57 videos with a corresponding background image. Each video is captured by a handheld or fixed camera with a person performing body motions in the front. We manually annotate all the 57 video samples and summarize that there are in total 17 different background scenes, 13 different people wearing 28 different kinds of clothes. Note that the manual annotation is available with our released codes. We select samples with background id 12 as the validation set, samples with background id 1, 3, 10 as the test set, and the remaining samples as the train set. Specifically, each sample is composed of a video frame and

Exp ID	#Inputs	Batch Size	Input Res.	Rel.	Down	#Blocks	$\mathcal{L}_{emb}(\lambda_1)$	$\mathcal{L}_{freq}(\lambda_2)$	$\mathcal{L}_{res}(\lambda_3)$	Remark
st-01	5	2	144	✓	✗	10	1	0.001	3	
st-02	5	1	144	✗	✗	30	1	0.001	3	
st-03	5	2	144	✓	✗	10	1	0	3	
st-04	7	2	144	✓	✗	10	1	0.001	3	
st-05	9	2	144	✓	✗	10	1	0.001	3	
st-06	3	2	144	✓	✓	10	1	0.001	0.4	
st-07	5	2	144	✓	✓	10	1	0.001	0.4	
mono-01	2	2	144	✓	✗	16	1	0.001	3	
mono-02	2	1	144	✗	✗	24	1	0.001	3	
mono-03	2	2	144	✓	✗	16	1	0	3	
dual-01	2	2	144	✓	✗	10	1	0.001	3	$\times$ 2
dual-02	2	2	144	✓	✗	10	1	0.001	3	$\times$ 4
dual-03	2	2	144	✓	✗	10	1	0.001	3	$\times$ 8
adobe-01	3	1	256	✓	✗	10	1	0.001	3	
real-01	2	1	256	✓	✗	10	1	0.001	3	
hide-01	2	2	144	✓	✗	10	1	0.001	3	
hide-02	3	2	144	✓	✗	10	1	0.001	3	
hide-03	4	2	144	✓	✗	10	1	0.001	3	
hide-04	5	2	144	✓	✗	10	1	0.001	3	
hide-05	5	1	144	✗	✗	30	1	0.001	3	
hide-06	5	2	144	✓	✗	10	1	0	3	

Table 1: A summary of training hyper-parameters in different experiments.

Step	**Embedding**				**Restored**			
	Zhu et al. [10]		Ours		Zhu et al. [10]		Ours	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
1	32.657	0.8652	**41.554**	**0.9754**	35.877	0.9486	**38.196**	**0.9606**
3	31.452	0.8274	**40.764**	**0.9689**	34.828	0.9380	**37.872**	**0.9583**
5	30.978	0.8115	**40.428**	**0.9662**	34.405	0.9334	**37.667**	**0.9570**

Table 2: Comparison on Temporal Video Embedding test set with embeding range of 9 and time step of 1 in grayscale.

its corresponding background image. For both datasets, we resize the input images as 256×256 during training using. Considering the memory issue, all the testing samples are first downsampled by 2 before testing.

1.5. Hiding Images in an Image

We use the Flicker 2W dataset [4], which selects $20,745$ high-quality and general images from Flickr.com. Since the dataset images have different resolutions, we select $19,531$ images with both width and height greater than 576 px for convenience. For training, the input images are first cropped as a 576×576 square at random positions independently, then they are concatenated and further applied random crop to obtain 144×144 patches. The input images are cropped as a 576×576 square at the center for testing.

2. Additional Experiments

The main paper generally focuses on multiple-and-single RIC tasks (conversion between multiple images and a single embedding image). For single-and-single RIC tasks like invertible colorization and invertible image rescaling, which are also critical applications of the RIC family, we present the results in Table 3 with the corresponding baseline methods [8, 9].

2.1. Invertible Grayscale

Similar to [8], we train and test our method using the Visual Object Classes Challenge 2012 (VOC2012) dataset, which contains $17,125$ color images in the dataset in total. We select the first $13,758$ images for training, and the remaining images are used for testing. We train on ran-

Tasks	Baseline		Ours	
	PSNR	SSIM	PSNR	SSIM
Grayscale	36.02	0.9681	**39.81**	**0.9803**
Rescaling	**44.32**	**0.9908**	41.77	0.9867

Table 3: Additional results on single-and-single RIC tasks.

domly cropped images with a resolution of 144×144 and test on full resolution images. One particular thing for invertible grayscale is that the input channel number is 3 (an RGB image) and that the embedding channel number is 1 (a grayscale image). In this sense, this experiment further demonstrates that our IICNet is indeed a generic framework for different RIC tasks.

2.2. Invertible Image Rescaling

Similar to IRN [9], we train and test our method using the DIV2K dataset [1], which consists of 800 training samples and 100 validation samples. We treat the official training samples as the train set and the 100 official validation samples as the test set. We only train a model to build a conversion between the high-resolution images and the corresponding $\times 2$ bicubic downsampled low-resolution images. During training, the input images are randomly cropped with a resolution of 144×144, and we do full resolution testing on the final trained model.

3. Additional Results

3.1. Image Comparison

To better demonstrate both the quantitative and qualitative performance of our method over the baseline method [10], we include more visual comparison results of the embedding images in Figure 1 and the restored frames in Figure 2.

3.2. Demo Video

We further make a demo video to show some visual results of our method on videos and GIF files. The demo video contains mainly three parts. The first part is about the spatial-temporal video embedding task, where our approach can embed high-resolution and high-fps videos into low-resolution and low-fps ones. We obtain this by embedding multiple high-resolution consecutive frames (a block of frames) into one low-resolution frame in a block-by-block manner. The second part is to embed small GIF files into still images, in which we can restore a small GIF from a single still embedding image. The last part is about monionizing binocular videos to embed binocular left-and-right videos into left-monocular ones, which is also performed in a per-frame manner.

Range	Embedding		Restored	
	PSNR	SSIM	PSNR	SSIM
11	37.097	0.9345	34.448	0.9342
13	36.874	0.9307	33.693	0.9232
15	36.665	0.9257	32.543	0.9080

Table 4: Ablation studies on different embedding ranges.

Setting	Embedding		Restored	
	PSNR	SSIM	PSNR	SSIM
PR	37.585	0.9584	36.914	0.9540
PP	37.554	0.9578	37.079	0.9558
HR	37.377	0.9568	36.824	0.9541
HP	37.486	0.9575	36.972	0.9550

Table 5: Ablation studies on different settings of downscaling layers and splitting strategies.

4. Additional Ablation Studies

4.1. Embedding Ranges

Due to memory limits, we can only conduct experiments with up to 9 input images using a batch size of 2 and input resolution of 144 during training. To further show the robustness of our framework to embed more images, we report experimental results of transforming multiple images with embedding range $N = 11, 13, 15$ into one embedding image. Results are shown in Table 4. Note that they are trained using the same experimental setup as described in Section 1, except that the batch size is 1.

4.2. Downscaling Layers and Splitting Strategies

We present an ablation study to investigate different downscaling layers and splitting strategies in the invertible downscaling modules.

For downscaling layers, we examine the pixel shuffling layer (squeezing operation) [2] and the Haar wavelet transformation layer [6]. Both offer an invertible operation to halve the resolution of the input images. Letting the input tensor as $\boldsymbol{x} \in \mathbb{R}^{C \times H \times W}$, the downscaling layer transforms the input tensor $\boldsymbol{x}$ from $\mathbb{R}^{C \times H \times W}$ to $\mathbb{R}^{4C \times \frac{H}{2} \times \frac{W}{2}}$. Specifically, a pixel shuffling layer simply does pixel shuffling by transforming each 2×2 square into the channel dimension with stride 2 spatially; a Haar Layer explicitly obtains one low-frequency and three high-frequency decomposition components by doing vertically and horizontally low-pass and high-pass filtering, namely the approximation, horizontal, vertical, and diagonal images.

We also examine two splitting strategies for the coupling layers (top parts and bottom parts). One is the strategy we present in our paper, denoted as "Ref". which is to split

Zhu et al. [10] Ours

Figure 1: Visual result comparisons on embedding images. (Zoomed in for details.)

the input tensor of coupling layers into reference and non-reference parts. Another is to split the input tensor according to a proportion of $1:3$, denoted as "Prop", where the 1 means all the first sub-images for the pixel shuffling layer or all the approximation images for the Haar layer; the 3 means all the remaining sub-images for the pixel shuffling layer or all the horizontal, vertical, and diagonal images for the Haar Layer.

There are four different combinational settings for Spatial-Temporal Video Embedding:

1) PR: pixel shuffling layer + "Ref";

2) PP: pixel shuffling layer + "Prop";

3) HR: Haar layer + "Ref";

4) HP: Haar layer + "Prop".

We train on the DAVIS 2017 video dataset [5] to embed $N = 3$ frames into one 2 times lower-resolution image, with a time step of 5. Note that the results are tested with a time step of 1. The quantitative comparison results reported in Table 5 show that there is no significant difference among the choices of downscaling layers and splitting strategies.

5. Implementation Details

Our experiments use Dense Block as the basic bottleneck design, which consists of 5 densely connected Conv-LeakyReLU layers with a channel growth rate of 32.

The relation module sequentially contains N independent headers of Dense Block followed by a 1×1 Conv., N convolutional layers with 3×3 kernels for independent

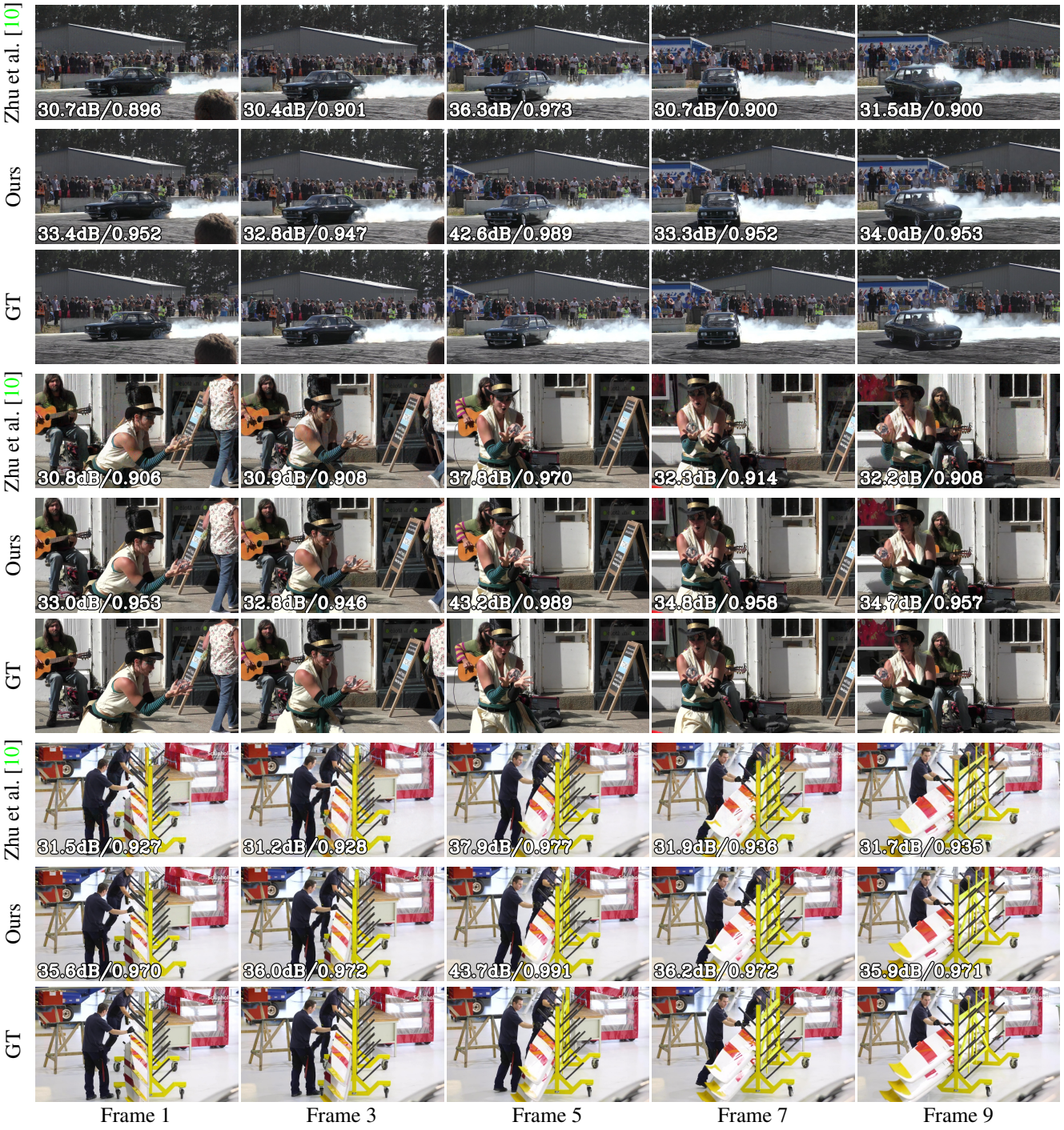

Figure 2: Visual result comparisons on restored frames. (Zoomed in for details.)

transformation, and N independent tailers of a 1×1 Conv. followed by a Dense Block, where the intermediate hidden channel dimension is 64.

For the INN architecture, we use different number of stacked invertible building blocks as described in Table 1, where we use Dense Block as the basic bottleneck design for the feedforward functions g, h_1, h_2 in coupling layers.

References

[1] Eirikur Agustsson and Radu Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017. 1, 3

[2] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint*

arXiv:1605.08803, 2017. 3

[3] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceddings of the International Conference on Learning Representations*, 2015. 1

[4] Jiaheng Liu, Guo Lu, Zhihao Hu, and Dong Xu. A unified end-to-end framework for efficient deep image compression. *arXiv preprint arXiv:2002.03370*, 2020. 2

[5] Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, Alex Sorkine-Hornung, and Luc Van Gool. The 2017 davis challenge on video object segmentation. *arXiv preprint arXiv:1704.00675*, 2018. 4

[6] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages I:511–I:518, 2001. 3

[7] Yingqian Wang, Longguang Wang, Jungang Yang, Wei An, and Yulan Guo. Flickr1024: A large-scale dataset for stereo image super-resolution. In *International Conference on Computer Vision Workshops*, pages 3852–3857, 2019. 1

[8] Menghan Xia, Xueting Liu, and Tien-Tsin Wong. Invertible grayscale. *ACM Transactions on Graphics (SIGGRAPH Asia)*, 37(6):246:1–246:10, 2018. 2

[9] Mingqing Xiao, Shuxin Zheng, Chang Liu, Yaolong Wang, Di He, Guolin Ke, Jiang Bian, Zhouchen Lin, and Tie-Yan Liu. Invertible image rescaling. In *Proceedings of the European Conference on Computer Vision*, pages 126–144, 2020. 2, 3

[10] Qianshu Zhu, Chu Han, Guoqiang Han, Tien-Tsin Wong, and Shengfeng He. Video snapshot: Single image motion expansion via invertible motion embedding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020. 1, 2, 3, 4, 5