Reduce Information Loss in Transformers for Pluralistic Image Inpainting Supplementary Material

Qiankun Liu[1*] Zhentao Tan[1] Dongdong Chen[2] Qi Chu[1†] Xiyang Dai[2]
Yinpeng Chen[2] Mengchen Liu[2] Lu Yuan[2] Nenghai Yu[1]
[1]University of Science and Technology of China
[2]Microsoft Cloud + AI
{liuqk3, tzt}@mail.ustc.edu.cn, {qchu, ynh}@ustc.edu.cn
cddlyf@gmail.com, {xiyang.dai, yiche, mengcliu, luyuan}@microsoft.com

1. Overview

In this supplementary material, we provide more implementation details, experimental results and analysis, including:

- training of P-VQVAE (Section 2).
- sampling strategy for image inpainting (Section 3).
- network architecture of different models (Section 4).
- more results on different datasets (Section 5).
- more discussions on PUT (Section 6), including the inference speed of PUT and some artifacts in inpainted results.

2. Training of P-VQVAE

Given an image $\mathbf{x}$ and two different masks $\mathbf{m}$ and $\mathbf{m}'$, the input of P-VQVAE is $\hat{\mathbf{x}} = \mathbf{x} \otimes \mathbf{m}$. The overall loss for the training of P-VQVAE is:

$$L_{vae} = \mathcal{L}_{rec}(\hat{\mathbf{x}}, \hat{\mathbf{x}}^R) + \parallel \mathrm{sg}[\hat{\mathbf{f}}] \ominus \hat{\mathbf{e}} \parallel_2^2 + \beta \parallel \mathrm{sg}[\hat{\mathbf{e}}] \ominus \hat{\mathbf{f}} \parallel_2^2, \tag{1}$$

where $\hat{\mathbf{f}} = \mathcal{E}(\hat{\mathbf{x}})$ denotes the feature vectors extracted by the encoder and $\hat{\mathbf{e}}$ is quantized vectors for $\hat{\mathbf{f}}$. $\hat{\mathbf{x}}^R = \mathcal{D}(\hat{\mathbf{e}}, \mathbf{m} \otimes \mathbf{m}', \hat{\mathbf{x}} \otimes \mathbf{m}')$ is the reconstructed image and $\mathrm{sg}[\cdot]$ refers to a stop-gradient operation that blocks gradients from flowing into its argument.

The last term in Eq. (1) is the so-called *commitment loss* [15] with weighting factor $\beta = 0.25$. It is responsible for passing gradient information from decoder to encoder. The second term in Eq. (1) is the codebook loss for the optimization of latent vectors. Following previous works in [13, 15], we replace the second term with the Exponential Moving Average (EMA) to optimize $\mathbf{e}$ and $\mathbf{e}'$. Specifically, at each iteration t, the latent vector $\mathbf{e}_k$ is updated as:

$$\begin{cases} n_k^t = n_k^{t-1} * \gamma + n_k * (1-\gamma), \\ \bar{\mathbf{e}}_k^t = \bar{\mathbf{e}}_k^{t-1} * \gamma + \sum_j^{n_k} (\hat{\mathbf{f}}^k)_j * (1-\gamma), \\ \mathbf{e}_k^t = \frac{\bar{\mathbf{e}}_k^t}{n_k^t}, \end{cases} \tag{2}$$

where $\hat{\mathbf{f}}^k$ denotes the set of feature vectors in $\hat{\mathbf{f}}$ that assigned to $\mathbf{e}_k$ and n_k is the number of feature vectors in $\hat{\mathbf{f}}^k$. γ is the

Algorithm 1: Sampling Strategy for Pluralistic Image Inpainting

Input : $\hat{\mathbf{x}} \in \mathbb{R}^{H\times W\times 3}$: masked image needs to be inpainted
$\mathbf{m} \in \{0,1\}^{H\times W\times 3}$: the mask indicating whether a pixel is masked/missing or not
$\mathcal{K}$: top-$\mathcal{K}$ for Gibbs sampling
Output: $\hat{\mathbf{x}}^I \in \mathbb{R}^{H\times W\times 3}$: the inpainted image

Step1: get indicator mask, feature vectors, quantized tokens
  $\mathbf{m}^{\downarrow} \in \{0,1\}^{\frac{H}{r}\times\frac{W}{r}\times 1}$: calculated from $\mathbf{m}$
  $\hat{\mathbf{f}} \in \mathbb{R}^{\frac{H}{r}\times\frac{W}{r}\times C} \leftarrow \mathcal{E}(\hat{\mathbf{x}})$
  $\hat{\mathbf{t}} \in \mathbb{N}^{\frac{H}{r}\times\frac{W}{r}} \leftarrow \mathcal{I}(\hat{\mathbf{f}}, \mathbf{e}, \mathbf{e}', \mathbf{m}^{\downarrow})$ // Sec. 3.1 in the paper
  $\hat{\mathbf{t}}^I \leftarrow \hat{\mathbf{t}}$
Step2: sample tokens for masked patches
  while $\sum_{i,j} \mathbf{m}^{\downarrow}_{i,j} < \frac{HW}{r^2}$ **do**
    $\hat{\mathbf{p}} \in [0,1]^{\frac{H}{r}\times\frac{W}{r}\times K} \leftarrow \mathcal{T}(\hat{\mathbf{f}})$ // probabilities, Sec. 3.2 in the paper
    // select the patch with maximum probability
    $i', j' \leftarrow \mathrm{argmax}_{i,j}(1-\mathbf{m}^{\downarrow}_{i,j}) \cdot \max \hat{\mathbf{p}}_{i,j,:}$
    // sample the token from the top-$\mathcal{K}$ elements in $\hat{\mathbf{p}}_{i',j',:}$
    $k \leftarrow \textsc{GibbsSampling}(\hat{\mathbf{p}}_{i',j',:}, \mathcal{K})$
    // update some variables
    $\hat{\mathbf{t}}^I_{i',j'} \leftarrow k, \mathbf{m}^{\downarrow}_{i',j'} \leftarrow 1, \hat{\mathbf{f}}_{i',j'} \leftarrow \mathbf{e}_k$
Step3: reconstruct the image
  $\hat{\mathbf{e}}^I \in\leftarrow \textsc{VectorRetrieval}(\hat{\mathbf{t}}^I, \mathbf{e})$
  $\hat{\mathbf{x}}^I \leftarrow \mathcal{D}(\hat{\mathbf{e}}^I, \mathbf{m}, \hat{\mathbf{x}})$
Return $\hat{\mathbf{x}}^I$

*Work done during an internship at Microsoft
†Corresponding author

Module	Layer	Parameter size / Stride	Output size
P-Enc	Linear	192×256	$32 \times 32 \times 256$
	Linear ResBlock	$\begin{pmatrix} 256 \times 128 \\ 128 \times 256 \end{pmatrix} \times 8$	$32 \times 32 \times 256$
	Linear	256×256	$32 \times 32 \times 256$
D-Codes	$\mathbf{e}$	512×256	-
	$\mathbf{e}'$	512×256	-
MSG-Dec	Conv	$256 \times 3 \times 3 \times 256/1$	$32 \times 32 \times 256$
	Conv ResBlock	$\begin{pmatrix} 256 \times 3 \times 3 \times 128/1 \\ 128 \times 3 \times 3 \times 256/1 \end{pmatrix} \times 8$	$32 \times 32 \times 256$
	Deconv (Conv)	$256 \times 4 \times 4 \times 256/2$ $(256 \times 4 \times 4 \times 256/2)$	$64 \times 64 \times 256$ $(32 \times 32 \times 256)$
	Deconv (Conv)	$256 \times 4 \times 4 \times 128/2$ $(128 \times 4 \times 4 \times 256/2)$	$128 \times 128 \times 128$ $(64 \times 64 \times 256)$
	Deconv (Conv)	$128 \times 4 \times 4 \times 64/2$ $(64 \times 4 \times 4 \times 128/2)$	$256 \times 256 \times 64$ $(128 \times 128 \times 128)$
	Conv† (Conv)	$64 \times 3 \times 3 \times 3/1$ $(3 \times 3 \times 3 \times 64/1)$	$256 \times 256 \times 3$ $(256 \times 256 \times 64)$

Table 1. Architecture of P-VQVAE. For MSG-Dec, the bracketed layers in the bottom four rows denotes the layers in reference branch. Except the convolution layer marked by †, all the other layers are followed by a ReLU [9] activation function. The structure of Linear and Conv ResBlocks are shown in Figure 1.

Module	Layer	Parameter size / Stride	Output size
Conv-Enc	Conv	$3 \times 4 \times 4 \times 64/2$	$128 \times 128 \times 64$
	Conv	$64 \times 4 \times 4 \times 128/2$	$64 \times 64 \times 128$
	Conv	$128 \times 4 \times 4 \times 256/2$	$32 \times 32 \times 256$
	Conv ResBlock	$\begin{pmatrix} 256 \times 3 \times 3 \times 128/1 \\ 128 \times 3 \times 3 \times 256/1 \end{pmatrix} \times 8$	$32 \times 32 \times 256$
	Conv	$256 \times 3 \times 3 \times 256$	$32 \times 32 \times 256$

Table 2. Architecture of the encoder in P-VQVAE$^{\text{conv}}$. The learnable codebook and decoder are the same with those in P-VQVAE in Table 1. All layers are followed by a ReLU [9] activation function.

decay parameter with the value between 0 and 1. We set $\gamma = 0.99$ in all our experiments.

The first term in Eq. (1) is the reconstruction loss and $\mathcal{L}_{rec}(\cdot, \cdot)$ is the function to get the difference between the inputted and reconstructed images. It consists of five parts, including L1 loss between the pixel values in two images (denoted as $\mathcal{L}_{pixel}$) and the gradients of two images (denoted as $\mathcal{L}_{grad}$), the adversarial loss [5] $\mathcal{L}_{adv}$, as well as the perceptual loss [7] $\mathcal{L}_{perc}$ and style loss [4] $\mathcal{L}_{style}$ between the two images. The design of the last three losses are inspired by the work in [10]. In the following, we describe the aforementioned losses in detail. Among them:

$$\mathcal{L}_{pixel} = \mathcal{M}(|\hat{\mathbf{x}} \ominus \hat{\mathbf{x}}^R|), \tag{3}$$

$$\mathcal{L}_{grad} = \mathcal{M}(|\text{grad}[\hat{\mathbf{x}}] \ominus \text{grad}[\hat{\mathbf{x}}^R]|), \tag{4}$$

where $\mathcal{M}(\cdot)$ refers to a mean-value operation, $\text{grad}[\cdot]$ is the function calculating the gradient of the given image.

The adversarial loss $\mathcal{L}_{adv}$ is computed with the help of a discriminator network $\mathcal{D}_{adv}(\cdot)$:

$$\mathcal{L}_{adv} = -\mathcal{M}(\log[1 \ominus \mathcal{D}_{adv}(\hat{\mathbf{x}}^R)]) - \mathcal{M}(\log[\mathcal{D}_{adv}(\hat{\mathbf{x}})]), \tag{5}$$

where $\log[\cdot]$ denotes element-wise logarithm operation. The architecture of the discriminator network is the same with that in [10].

The conceptual loss $\mathcal{L}_{perc}$ and style loss $\mathcal{L}_{style}$ are computed based on the activation maps from VGG-19 [14]:

$$\mathcal{L}_{perc} = \sum_{l}^{L_{perc}} \mathcal{M}(|\phi_l(\hat{\mathbf{x}}) \ominus \phi_l(\hat{\mathbf{x}}^R)|) \tag{6}$$

$$\mathcal{L}_{style} = \sum_{l}^{L_{style}} \mathcal{M}(|\mathcal{G}(\phi_l(\hat{\mathbf{x}})) \ominus \mathcal{G}(\phi_l(\hat{\mathbf{x}}^R))|) \tag{7}$$

where $\phi_l(\cdot)$ corresponds to different layers in VGG-19 [14], $\mathcal{G}(\cdot)$ denotes the function that gets the Gram matrix of its argument. For $\mathcal{L}_{perc}$ and $\mathcal{L}_{style}$, we set $L_{perc} = \{\text{relu1\_1}, \text{relu2\_1}, \text{relu3\_1}, \text{relu4\_1}, \text{relu5\_1}\}$ and $L_{perc} = \{\text{relu2\_2}, \text{relu3\_4}, \text{relu4\_4}, \text{relu5\_2}\}$. The overall reconstruction loss is:

$$\begin{aligned} \mathcal{L}_{rec} = \mathcal{L}_{pixel} + \lambda_g \mathcal{L}_{grad} + \lambda_a \mathcal{L}_{adv} \\ + \lambda_p \mathcal{L}_{perc} + \lambda_s \mathcal{L}_{style} \end{aligned} \tag{8}$$

In our implementation, we set $\lambda_g = 5$, $\lambda_a = 0.1$, $\lambda_p = 0.1$ and $\lambda_s = 250$.

Dataset	n'	h	D	D'	Param.
FFHQ [8]	30	8	512	64	95.0M
Places2 [18]	35	8	512	64	110.7M
ImageNet [2]	35	8	1024	128	441.7M

Table 3. UQ-Transformer with different model sizes for different datasets. n' and h are the number of transformer block and attention head. D is the dimensionality of feature vectors that before and after each transformer block. D' is the dimensionality of feature vector in each attention head.

3. Sampling Strategy for Image Inpainting

The overall procedure can be divided into three steps: 1) get the feature vectors $\hat{\mathbf{f}}$ from the masked image $\hat{\mathbf{x}}$ using encoder and get the tokens $\hat{\mathbf{t}}$ by quantizing $\hat{\mathbf{f}}$ with latent vectors in dual-codebook. The tokens for masked patches are not required; 2) get the tokens for masked patches using transformer. Note that the tokens are iteratively sampled with Gibbs sampling following previous transformer-based works [3, 11, 12]; 3) retrieve quantized vectors $\hat{\mathbf{e}}^I$ from codebook $\mathbf{e}$ based on the tokens and reconstruct the inpainted image $\hat{\mathbf{x}}^I$ using decoder by referencing to masked image $\hat{\mathbf{x}}$. The detailed sampling strategy is shown in Algorithm 1.

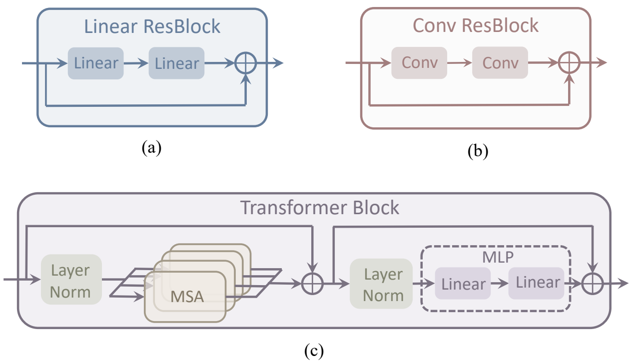

Figure 1. Architecture of different blocks. For Linear and Conv ResBlocks, each layer is followed by a ReLU [9] activation function. For transformer block, there is a GELU [6] activation function between the two linear layers. MSA: Multi-head Self-Attention. MLP: Multi-Layer Perceptron.

4. Network Architecture

4.1. Auto-Encoder

For different datasets, we use P-VQVAE with the same model size, and the architecture of our default P-VQVAE is shown in Table 1. The structure of Linear and Conv ResBlocks are shown in Figure 1 (a) and (b). In the paper, Section 4.3, several models are designed to show the effectiveness of different components in our method, including PUT^{conv}, PUT^{one}, $\text{PUT}^{\text{no\_ref}}$, PUT^{qua0} and PUT^{tok}. The auto-encoders in the last two models are the same with our default P-VQVAE. However, the auto-encoders in PUT^{conv} , PUT^{one} and $\text{PUT}^{\text{no\_ref}}$ are different. For the auto-encoder in PUT^{conv} (denoted as $\text{P-VQVAE}^{\text{conv}}$), all the linear layers in the encoder are replaced with convolution layers, and the input image is processed in a sliding window manner. Other modules in $\text{P-VQVAE}^{\text{conv}}$ are the same with those in P-VQVAE. The architecture of encoder in $\text{P-VQVAE}^{\text{conv}}$ (denoted as Conv-Enc) is shown in Table 2. The architecture of the auto-encoder in PUT^{one} is the same with P-VQVAE, except only one codebook $\mathbf{e}$ is used for training and testing. While for the auto-encdoer in $\text{PUT}^{\text{no\_ref}}$, it can be obtained from P-VQVAE by removing the reference branch in decoder.

4.2. Transformer

The architecture of transformer block is depicted in Figure 1 (c). There are several (denoted as n') successive transformer blocks in UQ-Transformer. Within each transformer block, the input features will be enhanced by self-attention. Formally, let $\bar{\mathbf{f}} \in \mathbb{R}^{\frac{HW}{r^2}\times D}$ be the input of transformer block. At the b-th transformer block, the feature vectors

Models \ Datasets	FFHQ [8]	Places2 [18]	ImageNet [2]
UQ-Transformer (# tokens/second)	37.138	32.048	17.186
P-VQVAE (# images/second)	62.949		

Table 4. Inference speed of different models. Tested on RTX 3090. The time consumption of P-VQVAE includes extracting feature vectors from image, quantizing feature vectors to latent vectors, and reconstructing the input image.

are processed as:

$$\begin{aligned} \tilde{\mathbf{f}}^{b-1} &= \bar{\mathbf{f}}^{b-1} + \text{MSA}(\text{LN}(\bar{\mathbf{f}}^{b-1})), \\ \bar{\mathbf{f}}^{b} &= \tilde{\mathbf{f}}^{b-1} + \text{MLP}(\text{LN}(\tilde{\mathbf{f}}^{b-1})), \end{aligned} \tag{9}$$

where $\text{LN}(\cdot)$, $\text{MLP}(\cdot)$, $\text{MSA}(\cdot)$ denote layer normalization [1], multi-layer perceptron and multi-head self-attention respectively. More specifically, given input $\mathbf{f} \in \mathbb{R}^{\frac{HW}{r^2}\times D}$, $\text{MSA}(\cdot)$ could be formated as:

$$\begin{aligned} \mathbf{h}_j &= \text{softmax}(\frac{(\mathbf{f}\mathbf{w}_{\text{q}}^j)(\mathbf{f}\mathbf{w}_{\text{k}}^j)^T}{\sqrt{D'}})(\mathbf{f}\mathbf{w}_{\text{v}}^j), \\ \text{MSA}(\mathbf{f}) &= [\mathbf{h}_0; \mathbf{h}_1; ...; \mathbf{h}_{h-1}]\mathbf{w}_{\text{o}}, \end{aligned} \tag{10}$$

where h is the number of head, $\mathbf{w}_{\text{q}}^j, \mathbf{w}_{\text{k}}^j, \mathbf{w}_{\text{v}}^j \in \mathbb{R}^{D\times D'}$, $\mathbf{w}_{\text{o}} \in \mathbb{R}^{hD'\times D}$ are the learnable parameters. $[\cdot; ...; \cdot]$ is the operation that concatenates the given arguments along the last dimension. By changing the values of h, D, D' and n', we can easily scale the size of UQ-Transformer.

We use UQ-Transformer with different model sizes for different datasets, which are shown in Table 3. As a reminder, the configuration of transformers are the same with those in ICT [17].

5. More Results

We show more qualitative comparisons for FFHQ [8] (Figure 3), Places2 [18] (Figure 4) and ImageNet [2] (Figure 5 and Figure 6).

6. More Discussions

Inference speed. As mentioned in Section 5 in the paper, the main limitation of PUT is the inference speed, which is also a common issue of existing transformer-based auto-regressive methods [3, 12, 16, 17]. Here we present the inference speed of PUT in Table 4. Note that the time consumption of inpainting a masked image depends on the area of masked regions.

Artifacts. We experimentally find that there sometimes contain some artifacts in the generated results of PUT, as

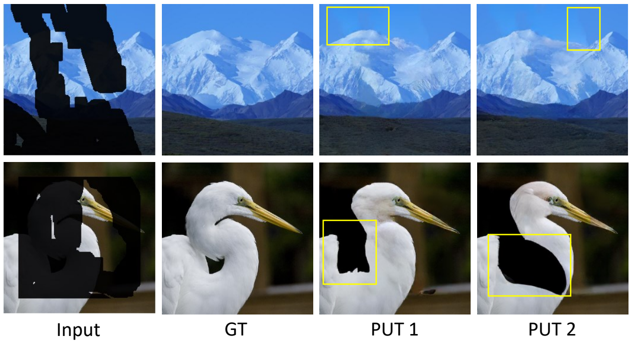

Figure 2. Results with artifacts. Top: color distortion. Bottom: black regions. Please pay attention to the contents in yellow rectangles.

shown in Figure 2. These artifacts can be divided into two categories. 1) Color distortion: the color of generated contents my not be consistent with the color of provided contents in the image. 2) Black region: PUT may produce black regions if the provided masked image contain lots of black pixels.

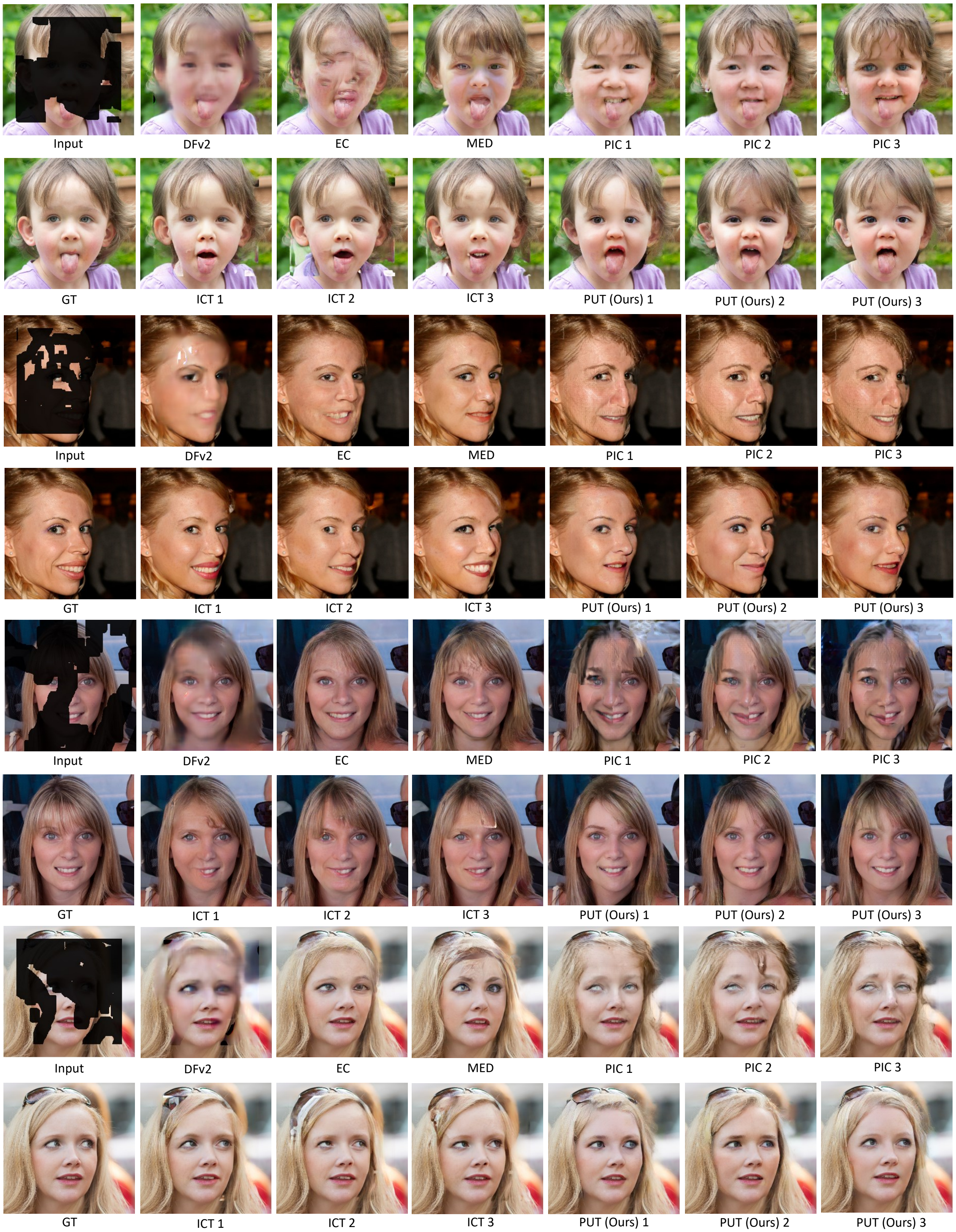

Figure 3. Qualitative comparisons between different methods on FFHQ [8].

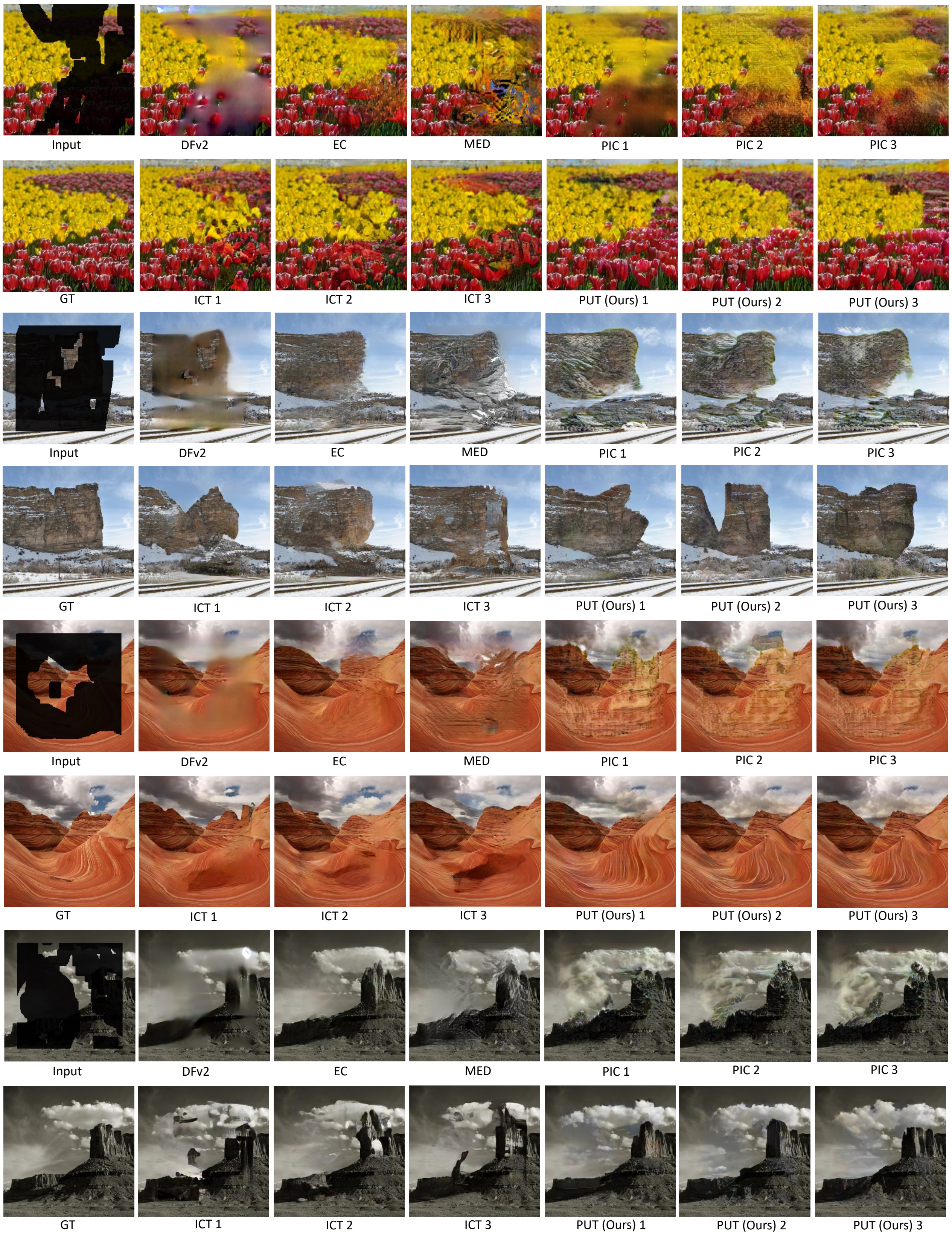

Figure 4. Qualitative comparisons between different methods on Places2 [18].

Figure 5. Qualitative comparisons between different methods on ImageNet [2].

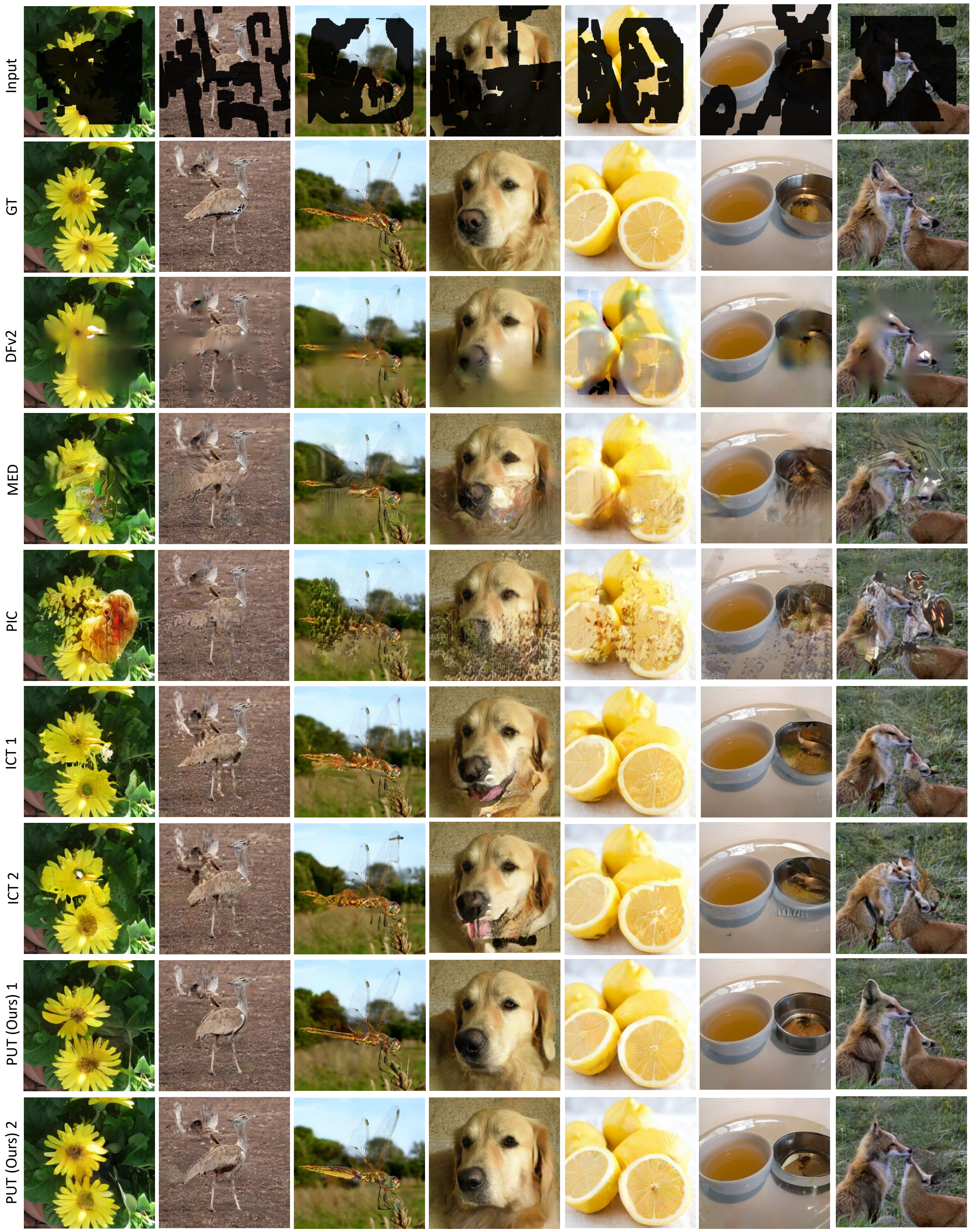

Figure 6. Qualitative comparisons between different methods on ImageNet [2].

References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 3

[2] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 2, 3, 7, 8

[3] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12873–12883, 2021. 2, 3

[4] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2414–2423, 2016. 2

[5] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014. 2

[6] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016. 3

[7] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pages 694–711. Springer, 2016. 2

[8] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4401–4410, 2019. 2, 3, 5

[9] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010. 2, 3

[10] Kamyar Nazeri, Eric Ng, Tony Joseph, Faisal Qureshi, and Mehran Ebrahimi. Edgeconnect: Structure guided image inpainting using edge prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019. 2

[11] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019. 2

[12] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. *arXiv preprint arXiv:2102.12092*, 2021. 2, 3

[13] Ali Razavi, Aaron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. In *Advances in neural information processing systems*, pages 14866–14876, 2019. 1

[14] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 2

[15] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6309–6318, 2017. 1

[16] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 3

[17] Ziyu Wan, Jingbo Zhang, Dongdong Chen, and Jing Liao. High-fidelity pluralistic image completion with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4692–4701, 2021. 3

[18] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE transactions on pattern analysis and machine intelligence*, 40(6):1452–1464, 2017. 2, 3, 6