A APPENDIX

A.1 EVALUATION METRICS

We consider two commonly used metric for evaluating image synthesis quality. (i) FID (Heusel et al., 2017) compares the statistics (mean and variances of Gaussian distributions) between the generated samples and real samples. FID is consistent with increasing disturbances and human judgment. Lower scores indicate that the model can generate higher quality images. To evaluate FID, we adopt the open-source code https://github.com/mseitzer/pytorch-fid. Specifically, we follow the dataset split protocol in Zhang et al. (2020) to compute FID. (ii) KID (Bińkowski et al., 2018) improves FID as an unbiased estimator, making it more reliable when there are fewer available test images. We use generated images translated from all test images in the source domain vs. test images in the target domain to compute KID. Lower KID values indicate that images are better translated. We use the same code as in Kim et al. (2020) to compute KID, which is open-sourced at https://github.com/taki0112/GAN_Metrics-Tensorflow.

For AMT evaluation, we follow the setting in FQ-GAN. Each testing image is judged by six users, who are asked to select the best-translated image to target domain where two standards should be considered, visual image quality, and the extent of background information is kept. We inform the participants of the name of the target domain and six example images of the target domain as a visual illustration.

Design	**Output shape**
RGB image $x \in \mathbb{R}^{3\times256\times256}$	$3 \times 256 \times 256$
FromRGB $3 \rightarrow ch/2$	$ch/2 \times 256 \times 256$
BlockDown $ch/2 \rightarrow ch$	$ch \times 128 \times 128$
FromRGB $3 \rightarrow ch$	$ch \times 128 \times 128$
BlockDown $ch \rightarrow 2ch$	$2ch \times 64 \times 64$
BlockDown $2ch \rightarrow 4ch$	$4ch \times 32 \times 32$
BlockDown $4ch \rightarrow 4ch$	$4ch \times 16 \times 16$
BlockDown $4ch \rightarrow 4ch$	$4ch \times 8 \times 8$
BlockDown $4ch \rightarrow 4ch$	$4ch \times 4 \times 4$
Block $4ch \rightarrow 4ch$	$4ch \times 1 \times 1$
Flatten & Linear	1

Table 5: The neural network architecture which can input the maximum resolution 256×256. ch is the channel multiplier and the default is 32 across all experiments. When the model grows the resolution from 128 to 256, the gray-shadowed FromRGB is abandoned and two new blocks above that are added. And so on for other resolution transitions.

A.2 ARCHITECTURES AND SETTINGS

The architecture detail up to 256×256 resolution is given in Table 5. Each BlockDown contains the 2-layer convolution operations where the first layer won't change feature size whereas the second layer comes with an average pooling 2×2 for down-sampling. The kernel size is 3×3 with stride 1. The last Block (without downsampling) first uses a 3×3 kernel and then uses a 4×4 kernel. The residual connection used in BlockDown is a single convolution and an average pooling of size 2×2. We don't apply the residual connection in the last Block. The stride is set to be 1 in all convolutions. FromRGB is a block for transforming RGB images to feature maps. Thus, in Section 3, the $\texttt{Primal}(\cdot)$ consists of a FromRGB and a BlockDown while the $\texttt{Fade}(\cdot)$ consists of an average pooling operation and a FromRGB block.

We train the network using Adam optimizer with $\beta_1 = 0.5$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. We set the learning rate schedule as $\alpha = \{8 : 0.001, 16 : 0.001, 32 : 0.001, 64 : 0.0012, 128 : 0.0015\}$ and the data feeding schedule as $N = \{8 : 50\text{K}, 16 : 75\text{K}, 32 : 100\text{K}, 64 : 125\text{K}, 128 : 150\text{K}\}$. In terms of the sampling hyperparameters, we set a schedule regrading the number of steps $T = \{8 : 15,\ 16 : 30,\ 32 : 50,\ 64 : 50,\ 128 : 60\}$, step size 1.0 and the standard deviation of an additive noise $\eta_t^\epsilon = 2e^{-2} - 2e^{-2}/(T - t + 1)$ in most experiments. As most experiments were implemented on the less powerful TITAN X (12 GB), we don't sweep over these values and hyper-parameters are very likely not to be optimal due to limited computational resources. Nonetheless, the presentation of experimental results is adequate to validate the proposed approach.

A.3 More image synthesis results

Figure 8 shows the generated images of which the EBM is modeled on CIFAR-10 and ImageNet-1K (32×32). Table 6 describes the quantitative results of image generation on ImageNet-1k from which CF-EBM is observed to achieve competitive results with a much smaller model.

In Figure 9, we visualize the short-run MCMC initialized from uniform noises for CelebA. Figure 10 shows the generated samples and their nearest neighbors from the training data. As is shown, the synthesized images are not present in the training data.

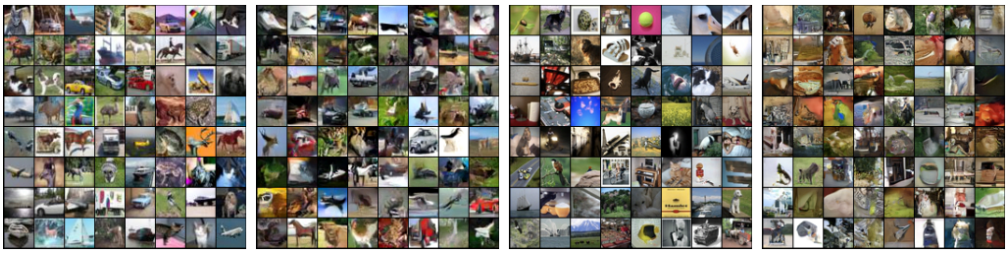

Figure 8: From Left to Right: ground truth of CIFAR-10, generated samples of CIFAR-10, ground truth of ImageNet-1K (32×32), and generated samples of ImageNet-1K (32×32).

Approach	Models	FID	#Params
Unconditional	PixelCNN (Van den Oord et al., 2016)	40.51	$>$ 35M
	PixelIQN-small	37.62	$>$35M
	PixelIQN-big	26.56	$>$50M
	CF-EBM (ours)	26.31	5.82M
Conditional	PixelCNN	33.27	-
	PixelIQN (Ostrovski et al., 2018)	22.99	
	EBM (Du & Mordatch, 2019)	14.31	
Self-supervision	SS-GAN (Chen et al., 2019b)	17.10	-
	MS-GAN (Tran et al., 2019)	12.30	

Table 6: FID on ImageNet-1K (32×32). The channel multiplier is $ch = 48$.

A.3.1 The effect of data perturbation

As almost all EBM related generative models add Gaussian noise $\mathcal{N}(0, \sigma^2\mathbf{I})$ to perturb the data for the purpose of training stabilization, it is necessary to investigate the effect of such perturbation. Nijkamp et al. (2019) has already reported that a larger perturbation leads to lower FID score where the minimum noise standard deviation is $\sigma = 0.03$. Also, the work Song & Ermon (2020) applies a sequence of decayed perturbation from 1 to 0.01. We claim that the model will memorize the leaked noisy information even if the final noise is small. As a result, the synthesized samples are noisy and foggy. Figure 11 shows the visualization effect at different levels of perturbation on the observed data. Note that, even the noise as small as $\sigma = 0.01$ can cause apparent visualization difference. Moreover, the noise term for sampling is usually not decayed to 0 such that each sample is noisy

Figure 9: Short-run MCMC generative sequences on CelebA (50 steps).

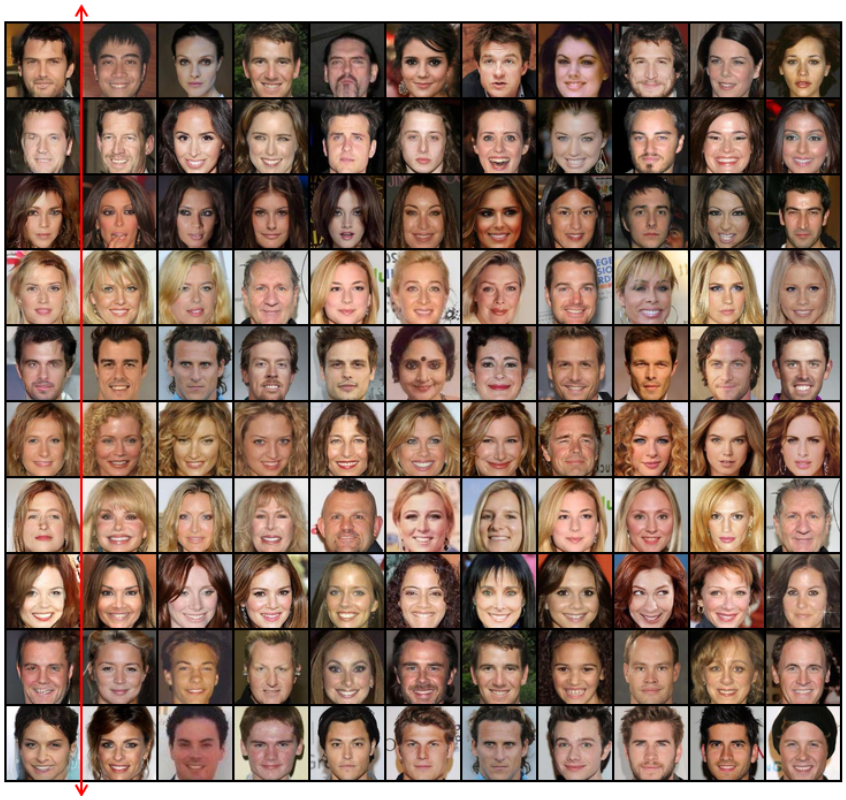

Figure 10: Nearest neighbors on CelebA (64×64). Generated samples are placed on the left side of red vertical line whereas 10 nearest neighbors are on the right side.

in the continuous image space. From Figure 2 in Nijkamp et al. (2019), Figure 1 in Song & Ermon (2020) and Figure 2 in Grathwohl et al. (2020), we observe the perturbation will be finally reflected in the generated samples. Therefore, the data perturbation should be taken as the last trick for generative modeling, or be discarded.

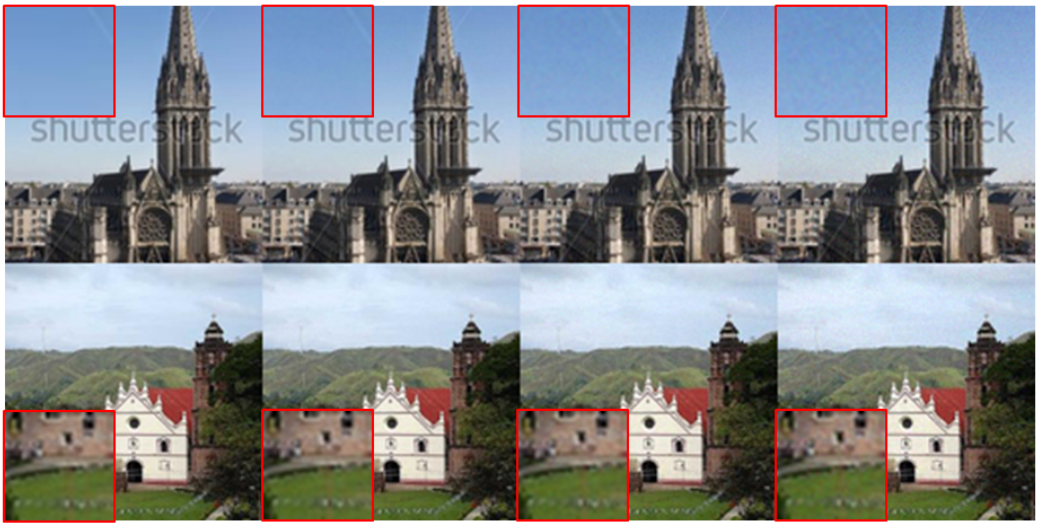

Figure 11: The visualization of different levels of perturbation, with red rectangle areas zoomed in. (LSUN Church 128×128, Yu et al. (2015))

A.4 Ablations

We mainly examine the activation functions, normalization and layer connections to see the model performance differences.

A.4.1 Activations

We show formulas of the following five activations and also give their derivatives:

- **Rectified Linear Unit** (ReLU) (Nair & Hinton, 2010): $f(x) = \max(0, x)$, and the derivative:

$$\frac{\partial f(x)}{\partial x} = \begin{cases} 1 & \text{if } x > 0, \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

- **Leaky Rectified Linear Unit** (LeakyReLU) (Maas et al., 2013): $f(x) = \max(0, x) + \alpha \cdot \min(0, x)$, and the derivative:

$$\frac{\partial f(x)}{\partial x} = \begin{cases} 1 & \text{if } x > 0, \\ \alpha & \text{otherwise.} \end{cases} \tag{6}$$

  where $\alpha = 0.2$ is the common setting.

- **Continuously Differential Exponential Linear Unit** (CELU) (Barron, 2017): $f(x) = \max(0, x) + \min(0, \alpha \cdot (\exp(x/\alpha) - 1)$, and the derivative:

$$\frac{\partial f(x)}{\partial x} = \begin{cases} 1 & \text{if } x > 0, \\ \alpha(\exp(x/\alpha) - 1) & \text{otherwise.} \end{cases} \tag{7}$$

- **Swish** (Ramachandran et al., 2016): $f(x) = x \cdot \text{Sigmoid}(x)$, and the derivative is:

$$\frac{\partial f(x)}{\partial x} = (2 - \text{Sigmoid}(x))\text{Sigmoid}(x) \tag{8}$$

- **Gaussian Error Linear Unit** (GELU) (Hendrycks & Gimpel, 2016): $f(x) = x \cdot \Phi(x)$ where $\Phi(\cdot)$ is the cumulative distribution function for Gaussian distribution. It is successfully applied in natural language processing like BERT. The derivative can be approximated as:

$$\frac{\partial f(x)}{\partial x} = (2.702 - \text{Sigmoid}(1.702x))\text{Sigmoid}(1.702x) \tag{9}$$

Figure 12 visualizes above activations and their derivatives. Obviously, both ReLU and LeakyReLU are non-smooth around 0. Image synthesis results on CIFAR-10 are shown in Table 7. We find that the use of LeakyReLU and ReLU diverge at an early time. These phenomenons are also observed on CelebA experiment. Therefore, smooth activations indeed improve the training stability and sample quality of EBM.

Activation	FID
ReLU	N/A
LeakyReLU	78.21
GELU	28.71
CELU($\alpha = 1$)	21.50
Swish	16.71

Table 7: On the impact of activation functions.

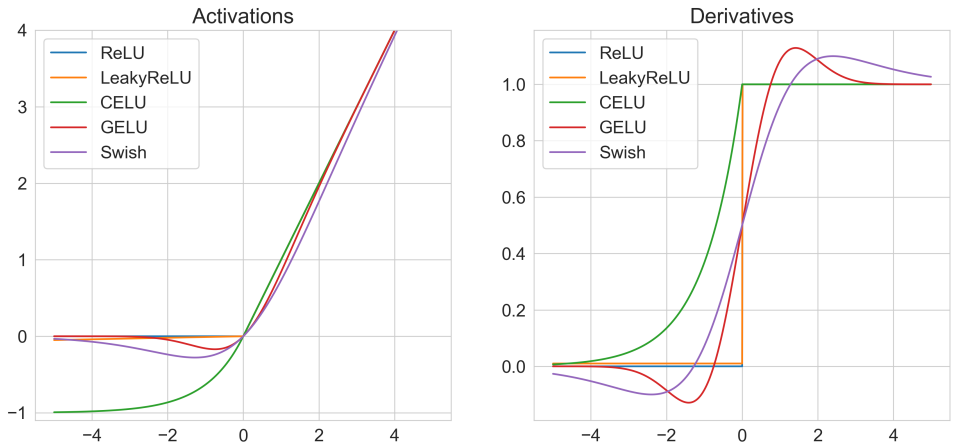

Figure 12: Activations and their derivatives

A.4.2 Normalization

Since the objective in Eq. (2) is similar to that in Wasserstein GAN, and the EBM acts as a discriminator, we thus naturally only consider spectral normalization (Miyato et al., 2018) to improve the performance and stability. SN constrains the Lipschitz constant of the learnable neural network parameters, which is widely used to stabilize the training of the discriminator network in GAN.

However, batch normalization is not appropriate here. Because each update of Langevin dynamics will rely on the running mean/variance of $\mathbf{x} \sim p_{data}$. Even if the statistic is not updated in Eq. (3), the distribution of $p_\theta(\mathbf{x}_t)$ is changing all the time along a chain. As a result, the pre-computed statistics of batch normalization don't fit each $\mathbf{x}_t$. Other normalizations, like instance normalization and layer normalization, are not appropriate in a network of which the target is to discriminate two distributions.

A.4.3 Ablate the components

We first examine how each component affects the model performances, including progressive growing, residual-connections, and spectral normalization. Results are shown in Table 8. From (a)-(c), we find

that spectral normalization and residual connection improved the sample quality. (d) Besides, we also compare the time efficiency regarding the training with and without coarse-to-fine settings. It is observed that the model without coarse-to-fine learning requires approximately $\mathbf{3}\times$ more time than the counterpart. (e) When the coarse EBM is fixed during training higher resolution, the FID is worse than that of a single fine EBM.

Configuration	**FID**
(a) Basic CF-EBM	32.01
(b) + Spectral normalization	20.83
(c) + Residual connection	**16.71**
(d) – Coarse-to-fine	24.67
(e) Fix coarse EBM	21.25

Table 8: Ablation results on CIFAR10.

Models	**FID**
Glow (Kingma & Dhariwal, 2018)	68.93
SN-GAN (Miyato et al., 2018)	25.95
WGAN-GP (Gulrajani et al., 2017)	22.57
CR-GAN (Zhang et al., 2020)	**16.97**
CF-EBM (ours)	23.50

Table 9: FID for image generation on CelebA-HQ (128×128). Results of GAN-based approaches are taken from Zhang et al. (2020).

A.5 Scale Up CF-EBMs

To scale up the EBM generative model to be capable of sampling higher resolution images, we integrate intermediate CF-EBMs that have been learned in Algorithm 1. When the training enters into modeling higher resolution, we fix the coarse-level CF-EBM, which only produces initial points for fine-level CF-EBM samplers. In this scenario, the number of Langevin steps can be reduced to 15 to expedite sampling at each scale, meaning that generating a 256×256 images requires 90 steps. On a single TITAN X GPU (12GB), the total training time (200k iterations) of CelebA-HQ 128×128 and 256×256 are about 120 hours and 235 hours, respectively. On a single TITAN V100 GPU, these costs can be reduced to 55 hours and 100 hours (we tested it once).

In this experiment, we fix the initial noise variance $\eta_0^\epsilon = 0.03$. We observe that, in the testing phase, choosing a different value of η_0^ϵ largely affects the synthetic results, as shown in Figure 13. A larger η_0^ϵ tends to add details to look more realistic but have more noisy images. Figure 14 shows more generated samples on CelebA-HQ 256×256. Table 9 compares the FID of different models on CelebA-HQ 128×128. Note that our approach obtains competitive results without any regularization term.

A.6 Unpaired Image-to-Image translation

A.6.1 Dataset

In this paper, we experiment on five popular image-to-image translation datasets:

selfie2anime It is first introduced in (Kim et al., 2020). The selfie and anime datasets each contains 3400 training images and 100 testing images.

photo2vangogh and apple2orange These datasets are used in (Zhu et al., 2017). The training dataset size of each class: 6,287 (photo), 400 (vangogh), 995 (apple) and 1019 (navel orange). The test datasets consist of 751 (photo), 400 (vangogh), 266 (apple) and 248 (navel orange).

Figure 13: Effect of changing the initial noise level η_0^ϵ at the testing phase.

Figure 14: CelebA-HQ 256×256 generated samples, with linearly cooling the "temperature" η_t^ϵ from 0.025 to 0.

Yosemite summer2winter This dataset is used in (Zhu et al., 2017). The training size of each class: 1273 (summer) and 854 (winter). The testing size of each class: 309 (summer) and 238 (winter).

cat2dog These datasets are used in DRIT (Lee et al., 2018a). The numbers of data for each class are 871 (cat) and 1,364 (dog). Follow (Kim et al., 2020), we use 100 (cat) and 100 (dog) as test data.

Experimental settings For the unpaired image-to-image translation task, we only made two changes compared with the settings used in image generation: (i) the channel multiplier is $ch = 16$; (ii) the learning rate is multiplied by 10 only for $dog \rightarrow cat$.

A.6.2 More Uncurated Image-to-Image Translation Examples

Figure 15: Translation dynamics: $orange \rightarrow apple$.

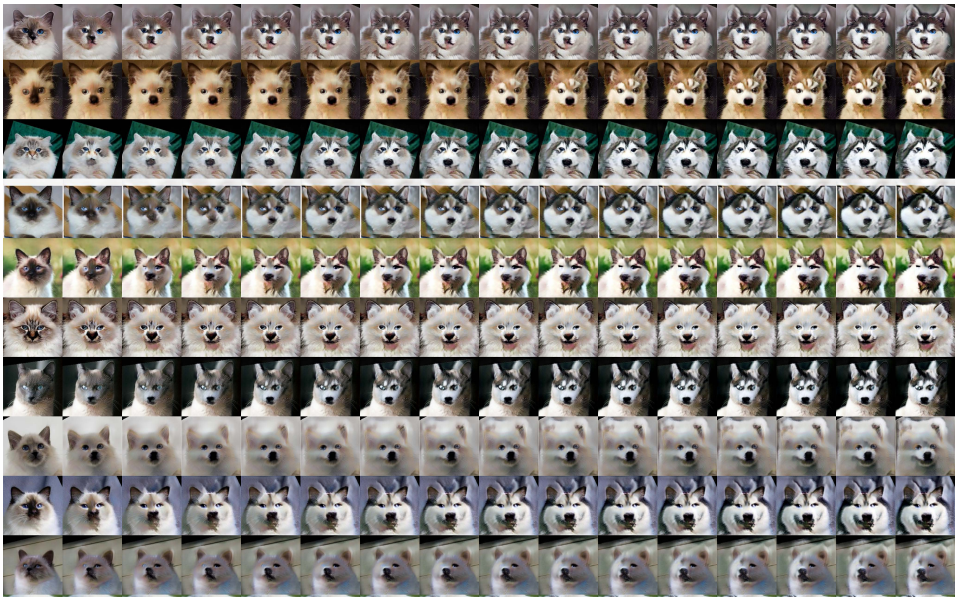

Figure 16: Translation dynamics: $cat \rightarrow dog$.

Figure 17: Translation dynamics: $photo \rightarrow vangogh$.

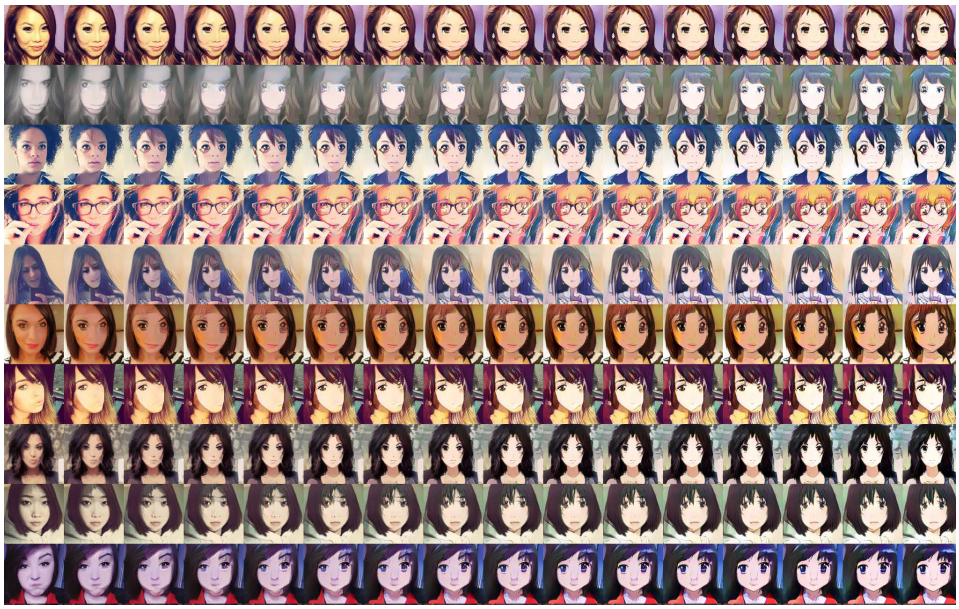

Figure 18: Translation dynamics: $selfie \rightarrow anime$.

Figure 19: Uncurated translation result: $photo \rightarrow Vangogh$

Figure 20: Uncurated translation result: $Vangogh \rightarrow photo$

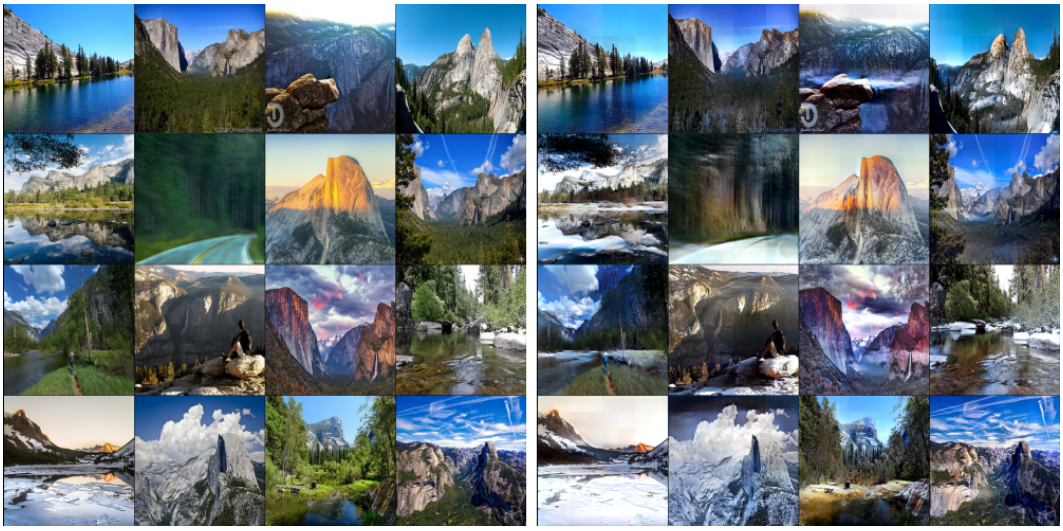

Figure 21: Uncurated translation result: Yosemite $summer \rightarrow winter$

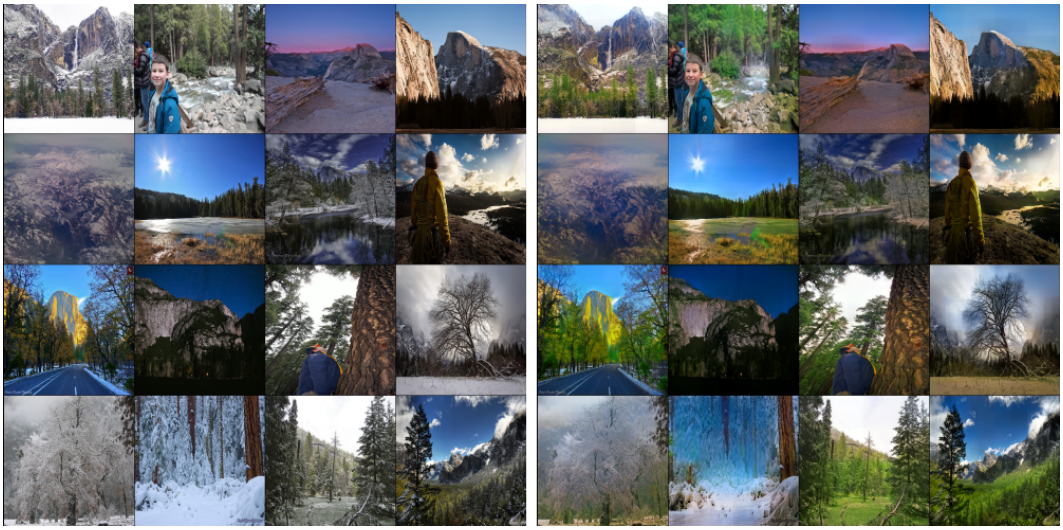

Figure 22: Uncurated translation result: Yosemite $winter \rightarrow summer$