Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks

Jun-Yan Zhu* Taesung Park* Phillip Isola Alexei A. Efros
Berkeley AI Research Laboratory (BAIR), UC Berkeley

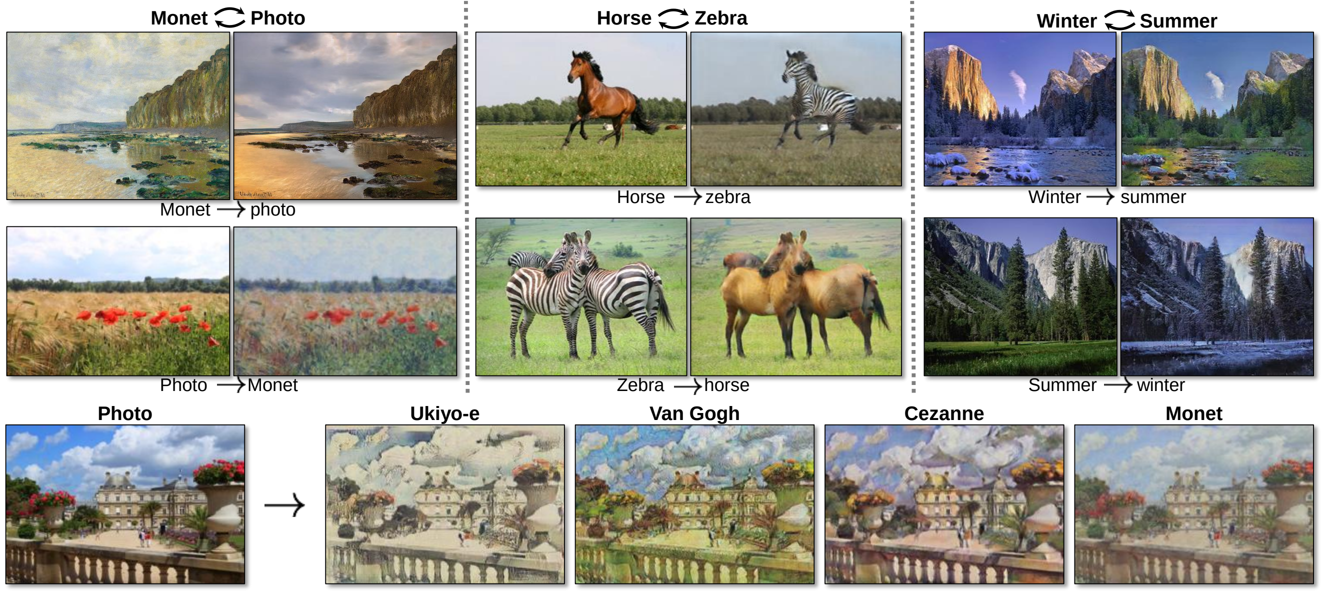

Figure 1: Image-to-image translation on problems where paired training data does not exist. We do not have photographs of what Monet saw when he painted the cliffs at Varengeville (top-left), but we can imagine what they may have looked like. Our algorithm learns to do the same.

Abstract

Image-to-image translation is a class of vision and graphics problems where the goal is to learn the mapping between an input image and an output image using a training set of aligned image pairs. However, for many tasks, paired training data will not be available. We present an approach for learning to translate an image from a source domain X to a target domain Y in the absence of paired examples. Our goal is to learn a mapping $G : X \rightarrow Y$ such that the distribution of images from $G(X)$ is indistinguishable from the distribution Y using an adversarial loss. Because this mapping is highly under-constrained, we couple it with an inverse mapping $F : Y \rightarrow X$ and introduce a cycle consistency loss *to push $F(G(X)) \approx X$ (and vice versa). Qualitative results are presented on several tasks where paired training data does not exist, including style transfer, object transfiguration, and season transfer. Quantitative comparisons against several prior methods demonstrate the superiority of our approach.*

1. Introduction

What did Monet see when he painted the cliffs at Varengeville (Figure 1, top-left)? We cannot travel back in time, but we can imagine: the teal rocks colored by damp algae, the dappled yellow clouds an impression of Monet's melancholy as he experienced the end of a beautiful day.

We can easily imagine this because we have seen hundreds of real sunsets and real seascapes. We have seen the glassy finish of tide pools on a calm afternoon, we know that back-lit rocks will arrive at our eye a deep gray. We can also exercise our imagination in the opposite direction. What would Monet have painted if he saw the poppies in the bottom-left of Figure 1? A brief stroll through a gallery of Monet paintings gives us ample ammunition for conjecture: likely he would have rendered the scene in pastel shades,

* indicates equal contributions

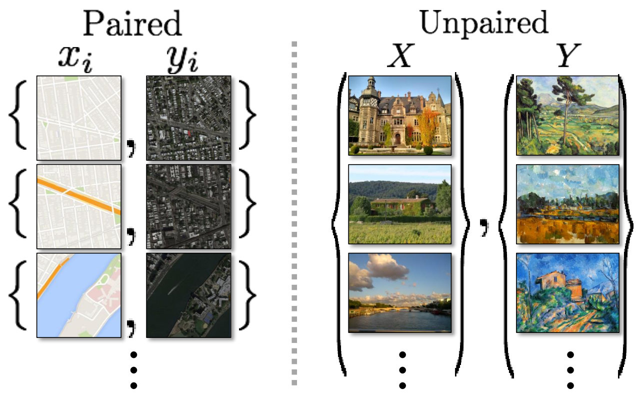

Figure 2: *Paired* training data (left) consists of training examples $\{x_i, y_i\}_{i=1}^N$, where the y_i that corresponds to each x_i is given. We instead consider *unpaired* training data (right), consisting of a source set $X = \{x_i\}_{i=1}^N$ and a target set $Y = \{y_j\}_{j=1}^M$, with no information provided as to which x_i matches which y_j.

with short and quick brush strokes, and a somewhat flattened dynamic range.

We can imagine all this despite never having seen a side by side example of a Monet painting next to a photo of the scene he painted. Instead we have knowledge of the set of Monet paintings and of the set of natural images. We can reason about the stylistic differences between these two sets, and thereby hallucinate what a scene would look like if we were to translate one style to the other.

Our goal in this paper is to develop a system that can learn to do the same, figuring out how to translate the style of one set of images to the style of another, in the absence of paired training examples.

This problem can be more broadly described as image-to-image translation [15], converting an image from one representation of a given scene, x, to another, y, e.g., greyscale to color, image to semantic labels, edge-map to photograph. Years of research in computer vision, image processing, and graphics have produced powerful translation systems in the supervised setting, where example image pairs $\{x, y\}$ are available, e.g., [7, 13, 15, 16, 20, 23, 33, 42, 44, 46]. However, obtaining paired training data can be difficult and expensive. For example, only a couple of datasets exist for tasks like semantic segmentation (e.g., [3]), and they are relatively small. Obtaining input-output pairs for graphics tasks like artistic stylization can be even more difficult since the desired output is highly complex, typically requiring artistic authoring. For many tasks, like object transfiguration (e.g., horse$\rightarrow$zebra, Figure 1 top-middle), the desired output is not even well-defined.

We therefore seek an algorithm that can learn to translate between domains without paired input-output examples (Figure 2, right). We assume there is some underlying relationship between the domains – for example, that they are two different renderings of the same underlying world – and seek to learn that relationship. Although we lack supervision in the form of paired examples, we can exploit supervision at the level of sets: we are given one set of images in domain X and a different set in domain Y. We may train a mapping $G : X \rightarrow Y$ such that the output $\hat{y} = G(x)$, $x \in X$, is indistinguishable from images $y \in Y$ by an adversary trained to classify $\hat{y}$ apart from y. In theory, this objective can induce an output distribution over $\hat{y}$ that matches the empirical distribution $p_Y(y)$ (note that, in general, this requires that G is stochastic) [11]. The optimal G thereby translates the domain X to a domain $\hat{Y}$ distributed identically as Y. However, such a translation does not guarantee that the individual inputs and outputs x and y are paired up in a meaningful way – there are infinitely many mappings G that will induce the same distribution over $\hat{y}$. Moreover, in practice, it may be difficult to optimize the adversarial objective in isolation: standard procedures often lead to the well-known problem of mode collapse, where all input images map to the same output image, and the optimization fails to make progress [10].

These issues call for adding more structure to our objective. We therefore exploit the property that translation should be "cycle consistent", in the sense that if we translate, e.g., a sentence from English to French, and then translate it back from French to English, we should arrive back at the original sentence. Mathematically, if we have a translator $G : X \rightarrow Y$ and another translator $F : Y \rightarrow X$, then G and F should be inverses of each other. We apply this structural assumption by training both the mapping G and F simultaneously, and adding a *cycle consistency loss* that encourages $F(G(x)) \approx x$ and $G(F(y)) \approx y$. Combining this loss with adversarial losses on domains X and Y yields our full objective for unpaired image-to-image translation.

We apply our method on a wide range of applications, including style transfer, object transfiguration, and attribute transfer. We also compare against previous approaches that rely either on hand-defined factorizations of style and content, or on shared embedding functions, and show that our method outperforms these baselines. Our code is available at `https://github.com/junyanz/CycleGAN`.

2. Related work

Generative Adversarial Networks (GANs) [11, 47] have achieved impressive results in image generation [27], image editing [50], feature learning [31], etc. Recent methods adopt the same idea for conditional image generation applications, such as text2image [28], image inpainting [26], and future prediction [25], and to other domains like videos [40] and 3D models [43]. The key to GANs' success is the idea of an *adversarial loss* which forces the generated images to be, in principle, indistinguishable from

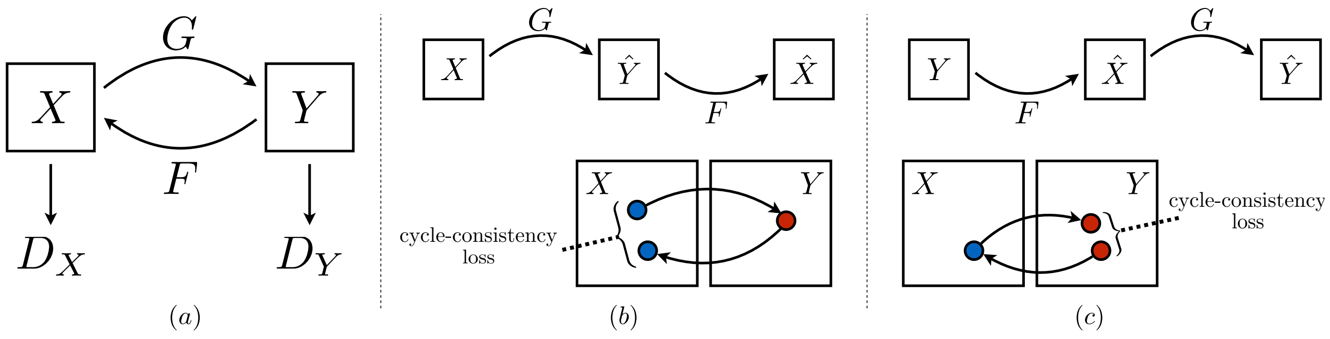

Figure 3: Overview: (a) our model contains two generative models $G : X \to Y$ and $F : Y \to X$, and associated adversarial discriminators D_X and D_Y. We introduce adversarial losses for both domains. In addition, we enforce cycle consistency in two directions: (b) forward cycle loss: $x \to G(x) \to F(G(x)) \approx x$, and (c) backward cycle loss: $y \to F(y) \to G(F(y)) \approx y$

real images. This is particularly powerful for image generation tasks, as this is exactly the objective that most of computer graphics aims to optimize. We adopt an adversarial loss to learn the mapping such that the translated image cannot be distinguished from images in the target domain.

Image-to-Image Translation The idea of image-to-image translation goes back at least to Hertzmann et al.'s Image Analogies [13], which learns a nonparametric texture model [6] from a single input-output training image pair. More recent approaches use a *dataset* of input-output examples to learn a parametric translation function using CNNs [23]. Our approach builds on the "pix2pix" framework of Isola et al. [15], which uses a conditional generative adversarial network [11] to learn a mapping from input to output images. Similar ideas have been applied to various tasks such as generating photographs from sketches [32] or from attribute and semantic layouts [17]. However, unlike these prior works, we learn the mapping without paired training examples.

Unpaired Image-to-Image Translation Several other methods also tackle the unpaired setting, where we seek to relate two data domains, X and Y. Rosales et al. [29] proposes a Bayesian framework that includes a prior based on a patch-based Markov random field computed from a source image, and a likelihood term obtained from multiple style images. More recently, CoupledGANs [22] and cross-modal scene networks [1] use a weight-sharing strategy to learn a common representation across domains. Concurrent to our method, Liu et al. [21] extends this framework with a combination of variational autoencoders [19] and generative adversarial networks. The other line of concurrent works [34, 37, 2] encourage the input and output to share certain "content" features even though they may differ in "style". They also use adversarial networks, with additional terms to enforce the output to be close to the input in a predefined metric space, such as class label space [2], image pixel space [34], and image feature space [37].

Unlike the above works, our formulation does not rely on any task-specific pre-defined similarity function between the input and output, nor do we assume that the input and output have to lie in the same low-dimensional embedding space. This makes our method a general-purpose solution for many vision and graphics tasks. We directly compare against several of prior methods in Section 5.1.

Neural Style Transfer [8, 16] is another way to perform image-to-image translation, which synthesizes a novel image by combining the content of one image with the style of another image (typically a painting) by matching the Gram matrix statistics of pre-trained deep features. Our main focus, on the other hand, is learning the mapping between two domains, rather than between two specific images, by trying to capture correspondences between higher-level appearance structures. Therefore, our method can be applied to other tasks, such as painting$\to$ photo, object transfiguration, etc. where single sample transfer methods do not perform well.

Cycle Consistency The idea of using transitivity as a way to regularize structured data has a long history. In visual tracking, enforcing simple forward-backward consistency has been a standard trick for decades [36]. More recently, higher-order cycle consistency has been used in structure from motion [45], 3D shape matching [14], co-segmentation [41], dense semantic alignment [48, 49], and depth estimation [9]. Of these, Zhou et al. [49] and Godard et al.[9] are most similar to our work, as they use a *cycle consistency loss* as a way of using transitivity to supervise CNN training. In this work, we are introducing a similar loss to push G and F to be consistent with each other.

3. Formulation

Our goal is to learn mapping functions between two domains $X = \{x_i\}_{i=1}^N$ and $Y = \{y_j\}_{j=1}^M$[1]. As illustrated in Figure 3 (a), our model includes two generative models $G : X \to Y$ and $F : Y \to X$. In addition, we in-

[1]For simplicity, we omit subscripts i and j in the following sections.

troduce two adversarial discriminators D_X and D_Y, where D_X aims to distinguish between images $\{x\}$ and translated images $\{F(y)\}$; in the same way, D_Y aims to discriminate between $\{y\}$ and $\{G(x)\}$. Our objective mainly contains two terms: an *adversarial loss* [11] for matching the distribution of generated images to the data distribution in the target domain; and a *cycle consistency loss* to further prevent the learned mappings G and F from contradicting each other.

3.1. Adversarial Loss

We apply adversarial losses [11] to both mapping functions. For the mapping function $G : X \rightarrow Y$ and its discriminator D_Y, we express the objective as:

$$\begin{aligned}\mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) =& \mathbb{E}_{y\sim p_{\text{data}}(y)}[\log D_Y(y)] \\ &+ \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log(1 - D_Y(G(x))], \end{aligned} \tag{1}$$

where G tries to generate images $G(x)$ that look similar to images from domain Y, while D_Y aims to distinguish between translated samples $G(x)$ and real samples y. G tries to minimize this objective against an adversarial D that tries to maximize it, i.e. $G^* = \arg\min_G \max_{D_Y} \mathcal{L}_{\text{GAN}}(G, D_Y, X, Y)$. We introduce a similar adversarial loss for the mapping function $F : Y \rightarrow X$ and its discriminator D_X as well: i.e. $F^* = \arg\min_F \max_{D_X} \mathcal{L}_{\text{GAN}}(F, D_X, Y, X)$.

3.2. Cycle Consistency Loss

As mentioned above, adversarial learning can match two empirical distributions [10]. However, with large enough capacity, a network can map the same set of input images to any random permutation of images in the target domain, where any of the learned mappings can induce an output distribution that matches the target distribution. Thus, an adversarial loss alone cannot guarantee that the learned function can map an individual input x_i to a desired output y_i. To further reduce the space of plausible mapping functions, we argue that the learned mapping functions should be cycle-consistent: as shown in Figure 3 (b), for each image x from domain X, the image translation cycle should be able to bring x back to the original image, i.e. $x \rightarrow G(x) \rightarrow F(G(x)) \approx x$. We call it *forward cycle consistency*. Similarly, as illustrated in Figure 3 (c), for each image y from domain Y, G and F should also satisfy the *backward cycle consistency*: $y \rightarrow F(y) \rightarrow G(F(y)) \approx y$. We formulate the cycle consistency loss as

$$\begin{aligned}\mathcal{L}_{\text{cyc}}(G, F) =& \mathbb{E}_{x\sim p_{\text{data}}(x)}[\|F(G(x)) - x\|_1] \\ &+ \mathbb{E}_{y\sim p_{\text{data}}(y)}[\|G(F(y)) - y\|_1], \end{aligned} \tag{2}$$

where L1 norm encourages sharper results compared to L2 [15]. We experimented with using both adversarial loss and L1 norm for measuring image reconstruction quality, but did not observe the advantage despite of the cost of extra computation.

3.3. Full Objective

Our full objective $\mathcal{L}(G, F, D_X, D_Y)$ is:

$$\mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) + \mathcal{L}_{\text{GAN}}(F, D_X, Y, X) + \lambda\mathcal{L}_{\text{cyc}}(G, F),$$

where λ controls the relative importance of the two objectives. We aim to solve:

$$G^*, F^* = \arg\min_{F,G} \max_{D_x, D_Y} \mathcal{L}(G, F, D_X, D_Y). \tag{3}$$

In Section 5.1.4, we compare our full method against the adversarial loss $\mathcal{L}_{\text{GAN}}$ alone and the cycle consistency loss alone $\mathcal{L}_{\text{cyc}}$, and empirically show that both objectives play critical roles in stabilizing training and arriving at high-quality results. We also evaluate our method with either only *forward cycle loss* or only *backward cycle loss*, and show that a single cycle is not be sufficient to regularize the training for this under-constrained problem.

4. Implementation

Network Architecture We adapt the architecture for our image generative network from Johnson et al. [16] who have shown impressive results for neural style transfer and super-resolution. This network contains two stride-2 convolutions, a few residual blocks [12], and two fractional-strided convolutions with stride $\frac{1}{2}$. We use 6 blocks for 128×128 images, and 9 blocks for 256×256 and higher-resolution training images. Similar to Johnson et al. [16], we use instance normalization [39]. Regarding the discriminator networks, we use a 70×70 PatchGAN [15, 34]. Different from vanilla GANs [11], PatchGAN learns to classify whether a 70×70 image patch is real or not. In comparison to vanilla GANs. PatchGAN has fewer parameters and can be applied to arbitrary images in a fully convolutional fashion [23].

Training details We apply two techniques from prior works to stabilize our model training procedure. First, for $\mathcal{L}_{GAN}$ (Equation 1), we replace the negative log likelihood objective by a least square loss [24] that minimizes the Pearson χ^2 divergence. This performs more stably during training and generates higher quality results. Equation 1 then becomes:

$$\begin{aligned}\mathcal{L}_{\text{LSGAN}}(G, D_Y, X, Y) =& \mathbb{E}_{y\sim p_{\text{data}}(y)}[(D_Y(y) - 1)^2] \\ &+ \mathbb{E}_{x\sim p_{\text{data}}(x)}[D_Y(G(x))^2], \end{aligned} \tag{4}$$

Second, to reduce model oscillation [10], we follow Shrivastava et al's strategy [34] and update discriminators

D_X and D_Y using a history of generated images rather than the ones produced by the latest generative networks. We keep an image buffer that stores 50 previously generated images.

For all the experiments, we set $\lambda = 10$ for the *cycle consistency loss* (Equation 2). We use the Adam solver [18] with batch size of 1. All networks were trained from from scratch, and trained with learning rate of 0.0002 for the first 100 epochs and linearly decaying rate to zero for the next 100 epochs. Please see more datasets and training details in the appendix (Section 7).

5. Results

We first compare our approach against recent methods for unpaired image-to-image translation on paired datasets where ground truth input-output pairs are available for evaluation. We then study the importance of both adversarial loss and cycle consistency loss, and compare our full method to several variants. Finally, we demonstrate the generality of our algorithm on a wide range of applications where paired data does not exist. For brevity, we name our method **CycleGAN**.

5.1. Quantitative Evaluation

Using the same evaluation datasets and metrics as "pix2pix" [15], we quantitatively test our method on the tasks of *semantic labels$\leftrightarrow$photo* on the Cityscapes dataset[3], and *map$\leftrightarrow$aerial photo* on data scraped from Google Maps.

5.1.1 Metrics

AMT perceptual studies On the maps$\leftrightarrow$photos task, we run "real vs fake" perceptual studies on Amazon Mechanical Turk (AMT) to assess the realism of our outputs. We follow the maps$\leftrightarrow$photos perceptual study protocol from [15], except we only gather data from 25 participants per algorithm. Note that the numbers we report here are not directly comparable to those in [15] as our images were processed slightly differently (our ground truth images were slightly blurrier than those in [15]), and the participant pool we tested may be differently distributed from those tested in [15]. Therefore, our numbers should be used to compare our current method against the baselines, rather than against [15].

FCN score Although perceptual studies may be the gold standard for assessing graphical realism, we also seek an automatic quantitative measure that does not require human experiments. For this we adopt the "FCN score" from [15], and use it to evaluate the Cityscapes labels$\rightarrow$photo task. The FCN metric evaluates how interpretable the generated photos are according to an off-the-shelf semantic segmentation algorithm (the fully-convolutional network, FCN, from [23]). The FCN predicts a label map for a generated photo. This label map can then be compared against the input ground truth labels using the standard semantic segmentation metrics described below.

Semantic segmentation metrics To evaluate the performance of photo$\rightarrow$labels, we use the standard metrics from the Cityscapes benchmark, including per-pixel accuracy, per-class accuracy, and mean class Intersection-Over-Union (Class IOU) [3].

5.1.2 Baselines

CoGAN [22] This method learns one GAN generator for domain X and one for domain Y. The two generators share weights on their first few layers, which encourages them to learn a shared latent representation, with the subsequent unshared layers rendering the representation into the specific styles of X and Y. Translation from X to Y can be achieved by finding the latent representation for an image X and then rendering it into style Y.

Pixel loss + GAN [34] Like our method, Shrivastava et al.[34] uses an adversarial loss to train a translation from X to Y. Whereas we regularize the mapping with our cycle-consistency loss, Shrivastava et al.[34] regularizes via the term $\|X - \hat{Y}\|_1$ that encourages the translation to be near an identity mapping.

Feature loss + GAN We also test a variant of [34] where the L1 loss is computed over deep image features (VGG-16 `relu4_2` [35]), sometimes called "Perceptual loss" [16], rather than over RGB pixel values.

BiGAN [5, 4] In the unconditional GAN setting [11], BiGAN [5] and ALI [4] propose to learn the inverse mapping function $F : X \rightarrow Z$ that projects a generated image back to a low-dimensional latent space Z when training the generative model $G : Z \rightarrow X$ and the adversarial discriminator D. Though originally designed for mapping a latent vector z to an image x, we implemented the same objective for mapping a source image x to a target image y.

pix2pix [15] We also compare against pix2pix [15], which is trained on paired data, to see how close we can get to this "upper bound" without using any paired training data.

For fair comparison, we implement all the baselines except the Coupled GAN [22] using the same architecture and implementation details as our method. CoupledGAN builds

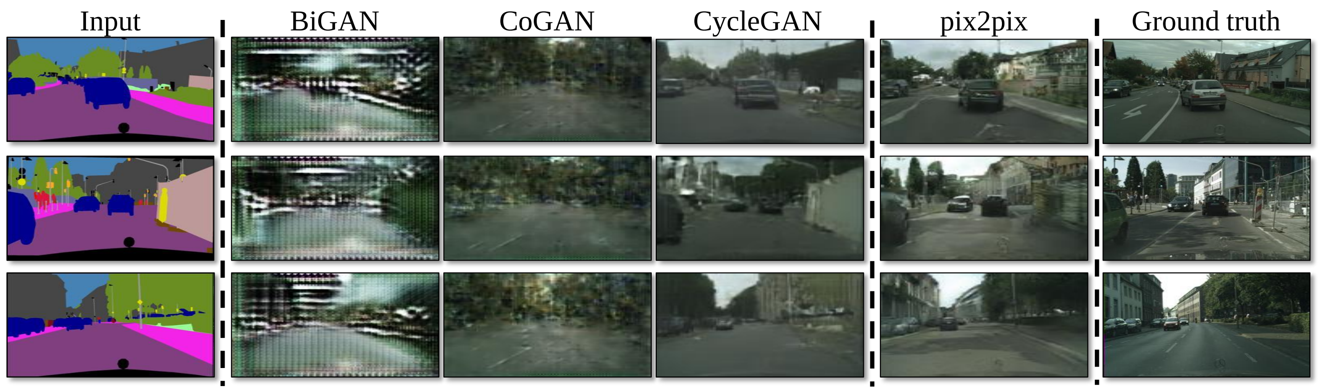

Figure 4: Different methods for mapping labels↔photos trained on cityscapes. From left to right: input, BiGAN [4, 5], CoupledGAN [22], CycleGAN (ours), pix2pix [15] trained on paired data, and ground truth.

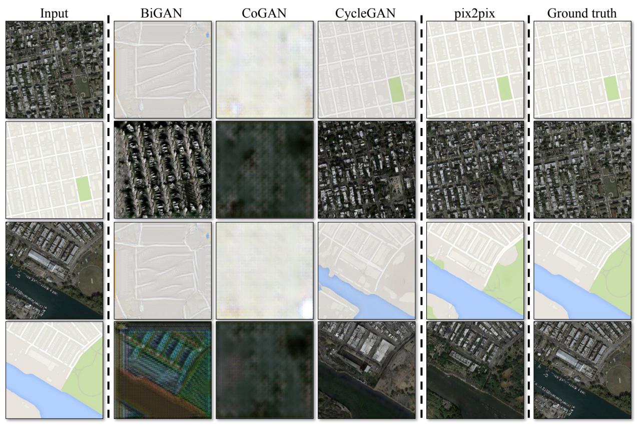

Figure 5: Different methods for mapping aerial photos↔maps on Google Maps. From left to right: input, BiGAN [4, 5], CoupledGAN [22], CycleGAN (ours), pix2pix [15] trained on paired data, and ground truth.

on generators that produce images from a shared latent representation, which is incompatible with our image-to-image architecture. We use their public implementation instead.

5.1.3 Comparison against baselines

As can be seen in Figure 4 and Figure 5, we were unable to achieve compelling results with any of the baselines. Our method, on the other hand, is able to produce translations that are often of similar quality to the fully supervised pix2pix. We exclude feature loss + GAN and pixel loss + GAN in the figures, as both of the methods fail to produce results close to the target domain.

Table 1 reports performance on the AMT perceptual realism task. Here, we see that our method can fool participants on around a quarter of trials, in both the map→photo direction and the photo→map direction. All baselines nearly never fooled our participants.

Loss	**Map → Photo** % Turkers labeled *real*	**Photo → Map** % Turkers labeled *real*
CoGAN [22]	0.6% ± 0.5%	0.9% ± 0.5%
BiGAN [5, 4]	2.1% ± 1.0%	1.9% ± 0.9%
Pixel loss + GAN [34]	0.7% ± 0.5%	2.6% ± 1.1%
Feature loss + GAN	1.2% ± 0.6%	0.3% ± 0.2%
CycleGAN (ours)	**26.8%** ± **2.8%**	**23.2%** ± **3.4%**

Table 1: AMT "real vs fake" test on maps↔aerial photos.

Loss	**Per-pixel acc.**	**Per-class acc.**	**Class IOU**
CoGAN [22]	0.40	0.10	0.06
BiGAN [5, 4]	0.19	0.06	0.02
L1+GAN [34]	0.20	0.10	0.0
Feature loss + GAN	0.07	0.04	0.01
CycleGAN (ours)	**0.52**	**0.17**	**0.11**
pix2pix [15]	0.71	0.25	0.18

Table 2: FCN-scores for different methods, evaluated on Cityscapes labels→photos. All the models are trained on 128×128 images

Loss	**Per-pixel acc.**	**Per-class acc.**	**Class IOU**
CoGAN [22]	0.45	0.11	0.08
BiGAN [5, 4]	0.41	0.13	0.07
Pixel loss+GAN [34]	0.47	0.11	0.07
Feature loss + GAN	0.50	0.10	0.06
CycleGAN (ours)	**0.58**	**0.22**	**0.16**
pix2pix [15]	0.85	0.40	0.32

Table 3: Classification performance of photo→labels for different methods on cityscapes.

Table 2 assesses the performance of the labels→photo task on the Cityscapes and Table 3 assesses the opposite mapping (photos→labels). In both cases, our method again outperforms the baselines.

5.1.4 Analysis of the loss function

In Table 4 and Table 5, we compare against ablations of our full loss. Removing the GAN loss substantially degrades results, as does removing the cycle-consistency loss. We therefore conclude that both terms are critical to our results. We also evaluate our method with cycle loss in only one direction and find that it often incurs training instability and

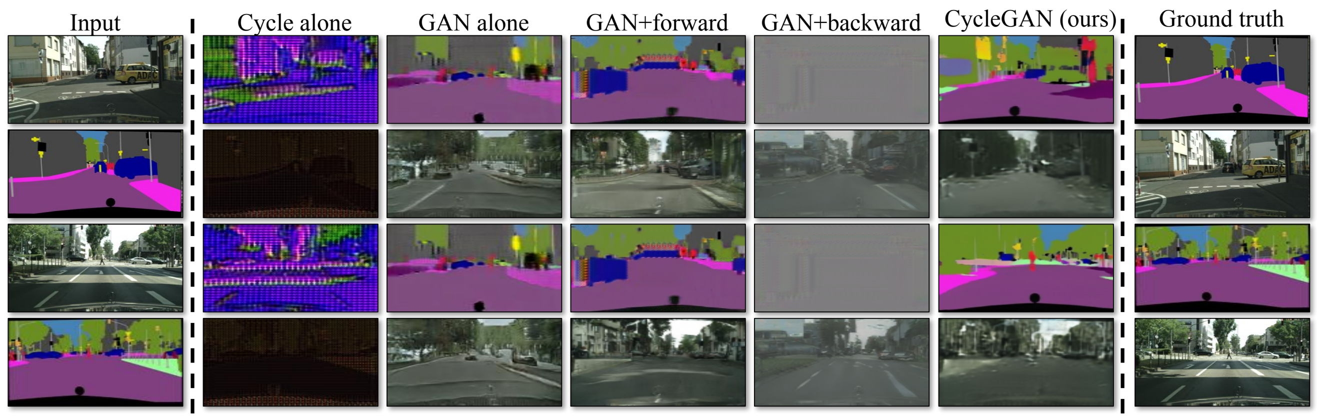

Figure 6: Different variants of our method for mapping labels↔photos trained on cityscapes. From left to right: input, cycle consistency loss alone, adversarial loss alone, GAN + forward cycle loss ($F(G(x)) \approx x$, labels→photos) GAN + backward cycle loss ($G(F(y)) \approx y$, photos→labels, CycleGAN (ours), and ground truth. Both Cycle alone and GAN + backward fail to produce images similar to the target domain. GAN alone and GAN + forward suffer from mode collapse, producing identical label maps regardless of input photos.

Loss	Per-pixel acc.	Per-class acc.	Class IOU
Cycle alone	0.22	0.07	0.02
GAN alone	0.52	0.11	0.08
GAN + forward cycle	**0.55**	**0.18**	**0.13**
GAN + backward cycle	0.41	0.14	0.06
CycleGAN (ours)	0.52	0.17	0.11

Table 4: Ablation study: FCN-scores for different variants of our method, evaluated on Cityscapes labels→photos.

Loss	Per-pixel acc.	Per-class acc.	Class IOU
Cycle alone	0.10	0.05	0.02
GAN alone	0.53	0.11	0.07
GAN + forward cycle	0.49	0.11	0.07
GAN + backward cycle	0.01	0.06	0.01
CycleGAN (ours)	**0.58**	**0.22**	**0.16**

Table 5: Ablation study: classification performance of photos→labels for different losses, evaluated on Cityscapes.

causes mode collapse, especially for the direction that has been removed. See Figure 6 for some qualitative examples.

5.1.5 Image reconstruction quality

In Figure 7, we show a few random samples of the reconstructed images $F(G(x))$ or $G(F(y))$. We observed that the reconstructed images were very close to the original inputs x and y, at both training and testing time, even in cases when one domain represents significantly more diverse information, such as the map ↔ aerial photos on Google Maps.

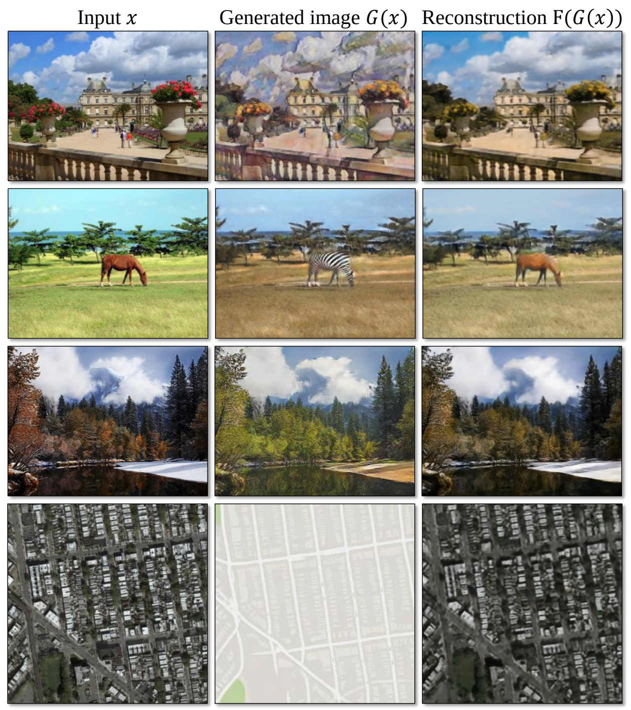

Figure 7: The reconstructed images from various experiments. Top row: test set result of photoUkiyo-e and photo→Cezanne. Middle left: horse→zebra. Middle right: training set result of winter→summer Yosemite. Bottom row: test set result of photo↔label on Google Maps on the ground-truth pairs in both directions.

5.2. Applications

We demonstrate our method on several applications where unpaired training data does not exist. Please refer to Section 7 for more details about the datasets.

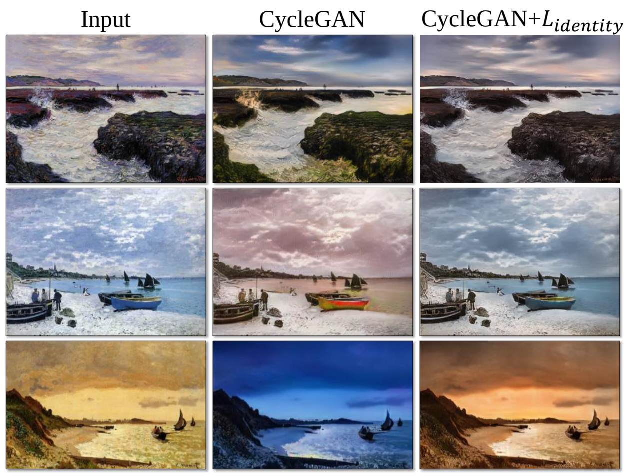

Figure 8: The effect of the *identity mapping loss* on Monet→ Photo. From left to right: input paintings, CycleGAN without identity mapping loss, CycleGAN with identity mapping loss. Identity mapping loss helps preserve the color of input paintings.

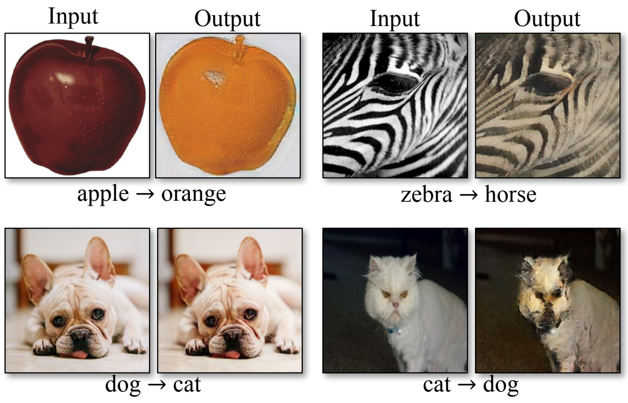

Figure 9: Typical failure cases of our method. Please see `https://people.eecs.berkeley.edu/~junyanz/projects/CycleGAN` for more comprehensive results.

Artistic style transfer (Figure 10) We train the model on landscape photographs downloaded from Flickr and WikiArt. Note that unlike recent work on "neural style transfer" [8], our method learns to mimic the style of an entire *set* of artworks, rather than transferring the style of a single selected piece of art. Therefore, we can learn to generate photos in the style of, e.g., Van Gogh, rather than just in the style of Starry Night. The size of the dataset for each artist/style was 526, 1073, 400, and 563 for Cezanne, Monet, Van Gogh, and Ukiyo-e. All artwork images were fetched from Wikiart.

Object transfiguration (Figure 11) The model is trained to translate one object class from Imagenet [30] to another (each class contains around 1000 training images). Turmukhambetov et al.[38] proposes a subspace model to translate one object into another object of the same category, while our method focuses on object transfiguration between two visually similar categories.

Season transfer (Figure 11) The model is trained on 854 winter photos and 1273 summer photos of Yosemite downloaded from Flickr.

Photo generation from paintings (Figure 12) For painting→photo, we find that it is helpful to introduce an additional loss to further stabilize the training procedure, as used in [37]. We require the generator to be the identity mapping if the real samples of the target domain are provided as the input to the generator: i.e. $\mathcal{L}_{\text{identity}}(G, F) = \mathbb{E}_{y\sim p_{\text{data}}(y)}[\|G(y) - y\|_1] + \mathbb{E}_{x\sim p_{\text{data}}(x)}[\|F(x) - x\|_1]$.

The identity mapping loss acts as an effective stabilizer at early stage of training. Without $\mathcal{L}_{\text{identity}}$, the generator G and F are free to change the tint of input images when there is no need to. For example, when learning the mapping between Monet's paintings and Flickr photographs, the generator often maps paintings of daytime to photographs taken during sunset, because such mapping is equally valid under the adversarial loss and cycle consistency loss. The effect of the identity mapping loss are shown in Figure 8.

In Figure 12, we show results on painting→photo with identity mapping loss.

Comparison with Gatys et al. [8] In Figure 13, we compare our results with neural style transfer [8] on photo stylization. For each row, we first use two representative artworks as the style images for [8]. We then compute the average the Gram Matrix across the target domain, and use this matrix to transfer the "average style". Our method is able to produce photos in the style of entire collection, ignoring specific color and texture details from individual image.

Figure 14 demonstrates similar comparisons for other translation tasks. For each application, we first show results from [8] using two different images from the target domain as style images. We then generate results with "average style" using all the images from the target domain. We observe that Gatys et al. [8] fails to produce photo-realistic results and often introduces artifacts, while our method succeed to generate natural looking results, similar to the target domain.

6. Discussion and limitations

Although our method can achieve compelling results in many cases, the results are far from uniformly positive. Several typical failure cases are shown in Figure 9. On translation tasks that involve color and texture changes, like many of those reported above, the method often succeeds. We have also explored tasks that require geometric changes, with little success. For example, on the task of dog→cat transfiguration, the learned translation degenerates to making minimal changes to the input (Figure 9). Handling more varied and extreme transformations, especially geometric changes, is an important open problem for future work.

We also observe a lingering gap between the results achievable with paired training data and those achieved by our unpaired method. In some cases, this gap may be very hard – or even impossible,– to close: for example, our method permutes the labels for tree and building in the output of the photos→labels task. To resolve this ambiguity may require some form of weak semantic supervision. Integrating weak or semi-supervised data may lead to substantially more powerful translators, at a fraction of the annotation cost of the fully-supervised systems.

Nonetheless, in many cases completely unpaired data is plentifully available and should be made use of. This paper pushes the boundaries of what is possible in this "unsupervised" setting.

Acknowledgments: We thank Aaron Hertzmann, Shiry Ginosar, Deepak Pathak, Bryan Russell, Eli Shechtman and Tinghui Zhou for helpful comments. This work was supported in part by NSF SMA-1514512, Google Research Award, Intel Corp, and hardware donation from nVIDIA. JYZ is supported by the Facebook Graduate fellowship and TP is supported by the Samsung Scholarship. All photographs are taken by AE, mostly in France.

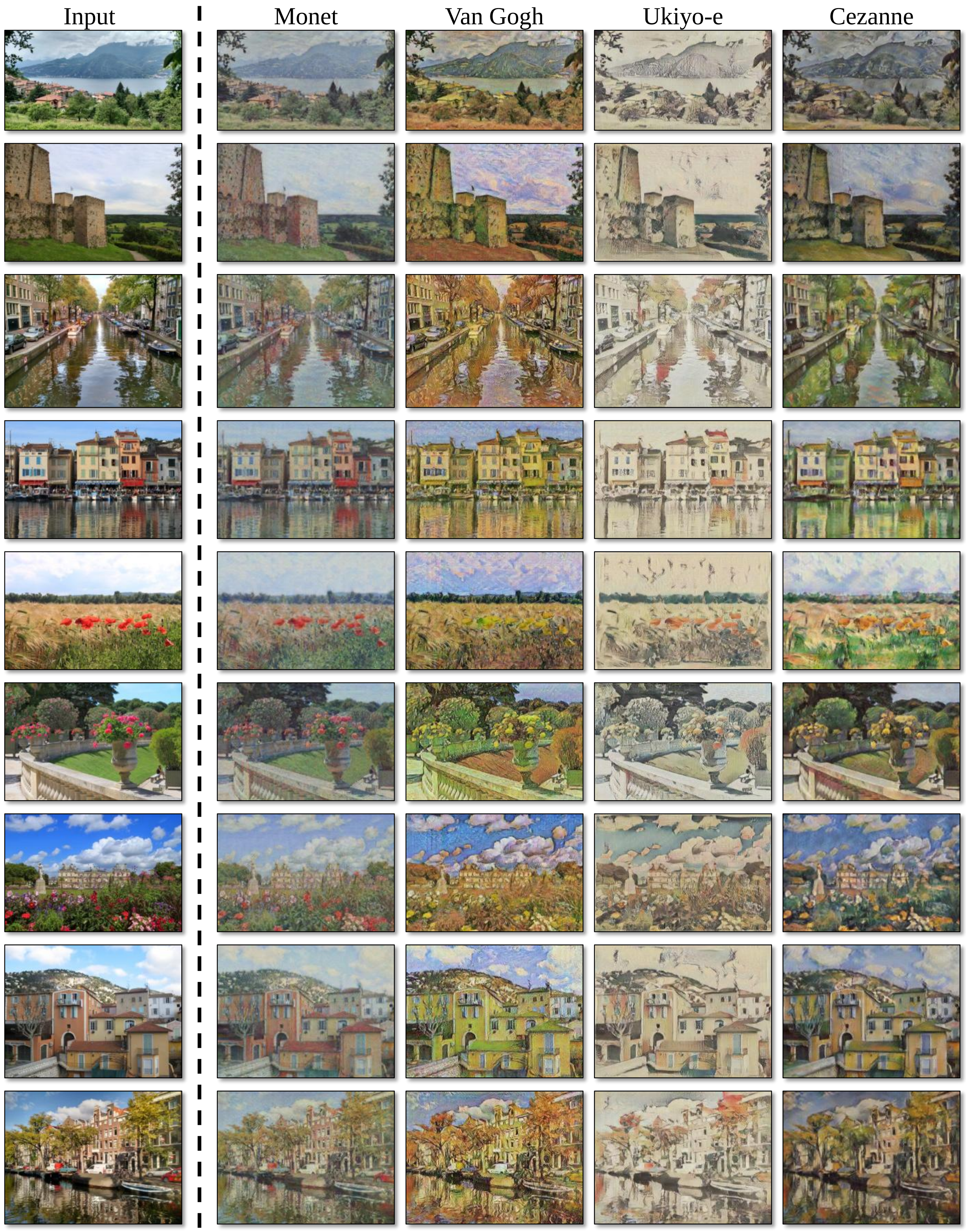

Figure 10: Stylizing Photos: We transfer input images into artistic styles of Monet, Van Gogh, ukiuo-e, and Cezanne. Input photos personally taken by the corresponding author in France. Please see `https://people.eecs.berkeley.edu/~junyanz/projects/CycleGAN` for additional examples.

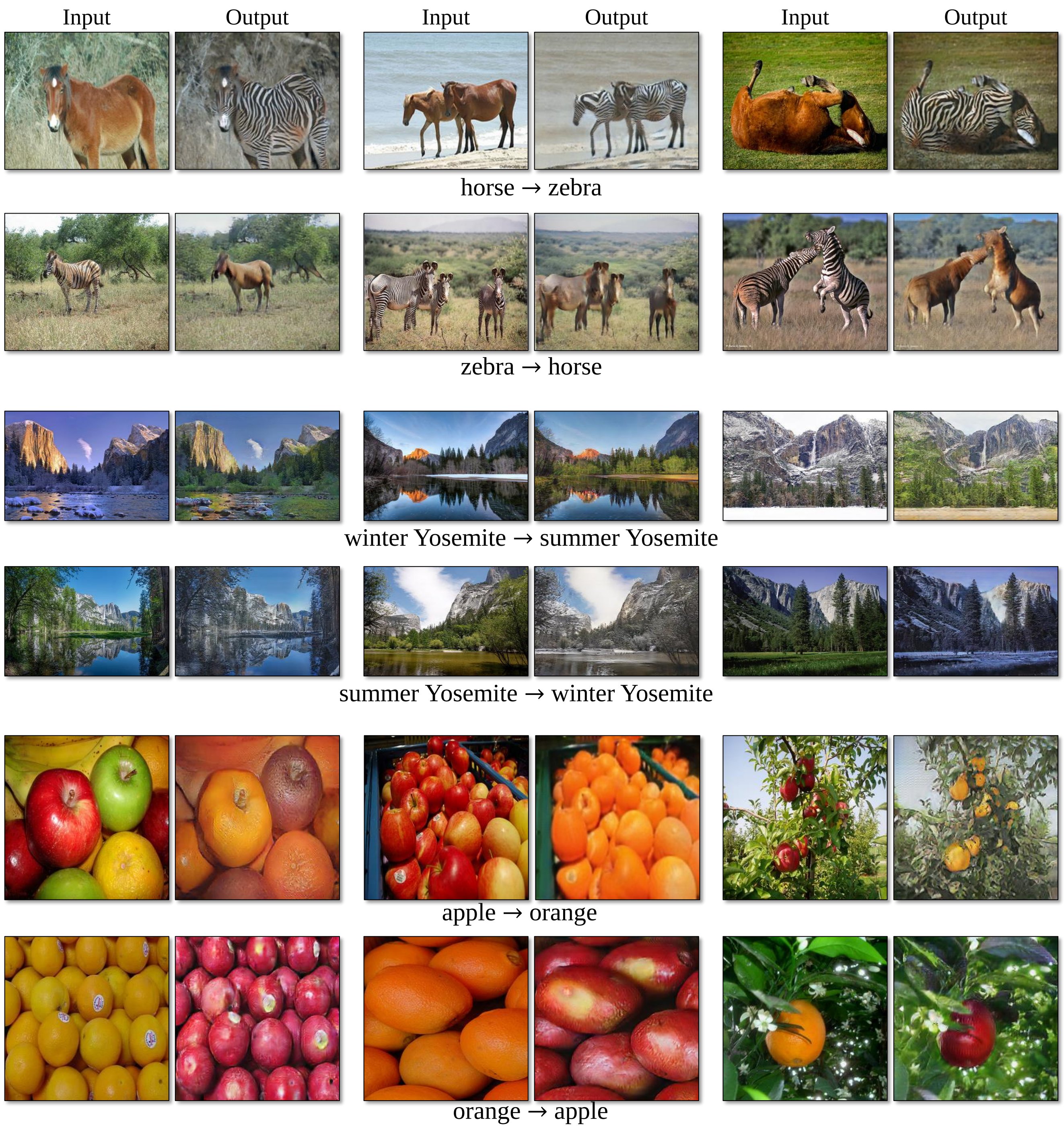

Figure 11: Our method in various applications. We show some successful cases of our result. In the top two rows, we show our result on object transfiguration between horses and zebra from ImageNet. The middle two rows show our result on transferring seasons of Yosemite in the Flickr photos. In the bottom two rows, we run our method on apples and oranges, fetched from ImageNet. Please see `https://people.eecs.berkeley.edu/~junyanz/projects/CycleGAN` for additional examples.

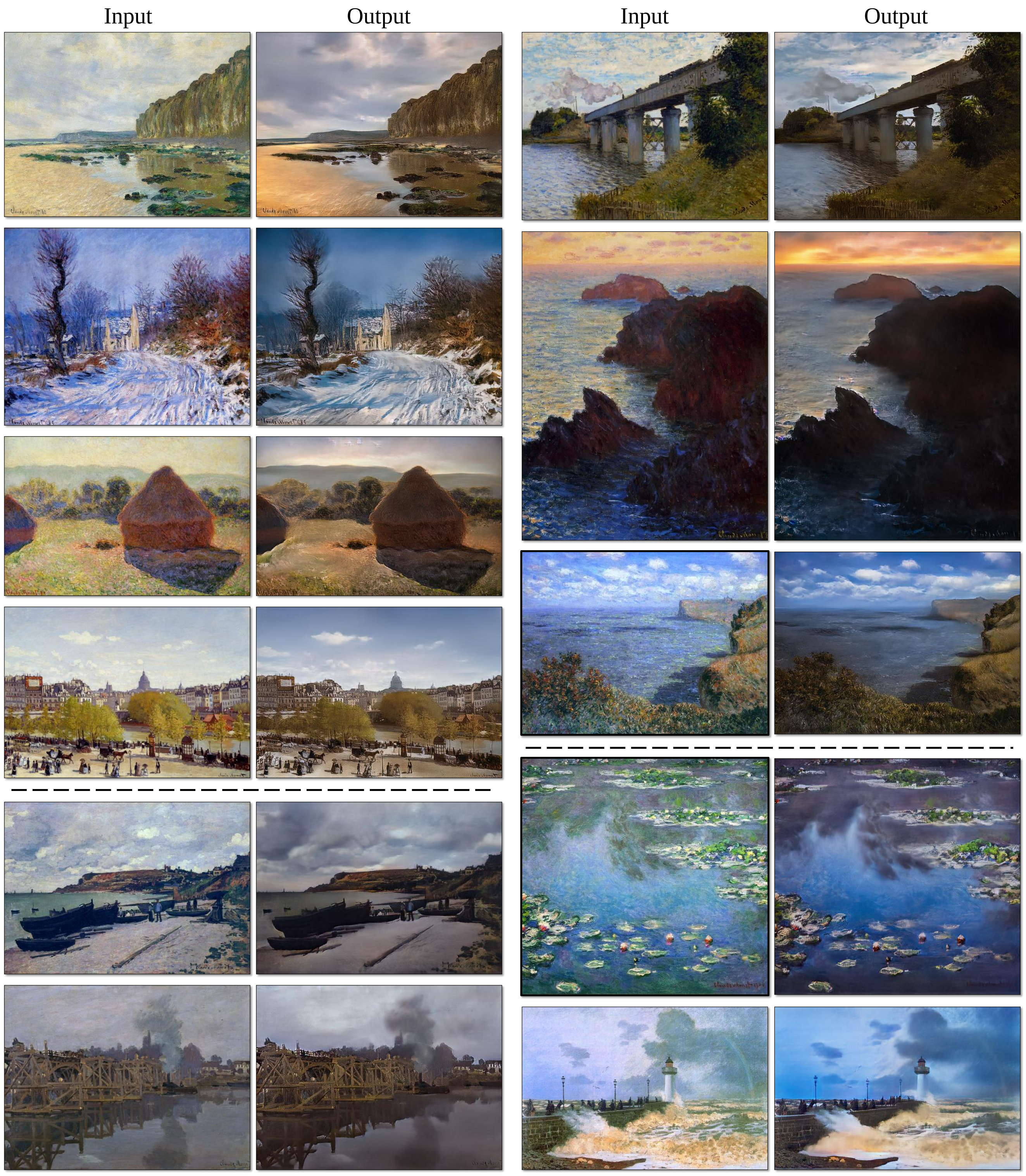

Figure 12: Monet→ Photo: Mapping Monet paintings to landscape photographs from Flickr. Please see `https://people.eecs.berkeley.edu/~junyanz/projects/CycleGAN` for additional examples.

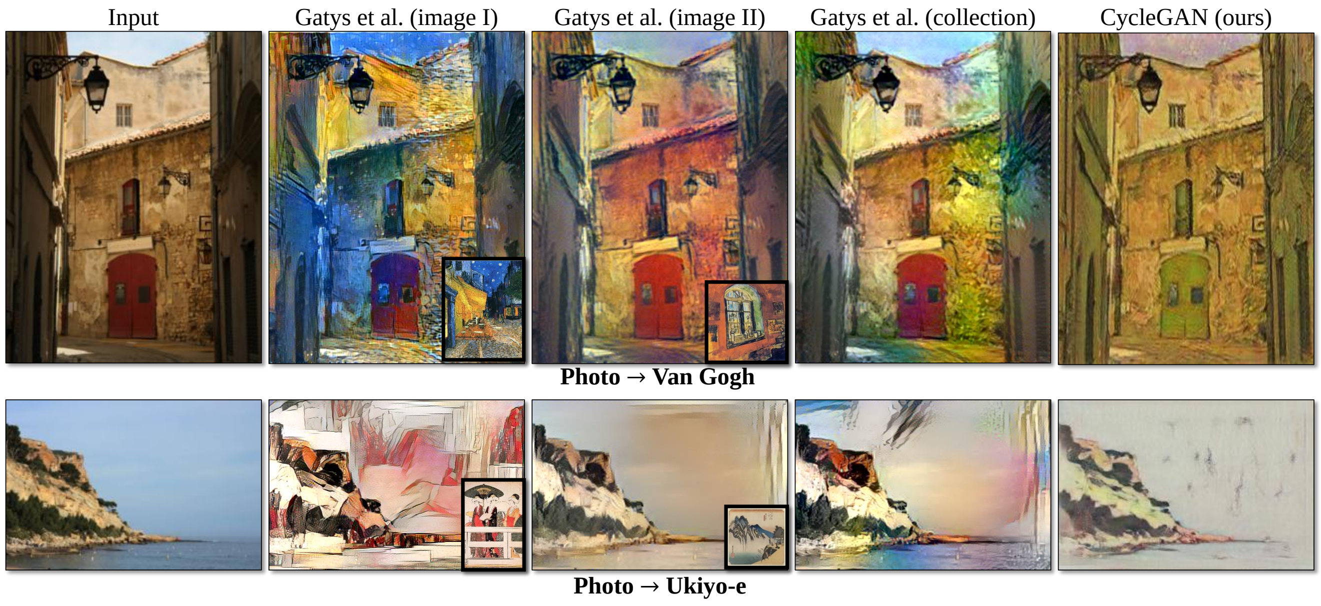

Figure 13: We compare our method with neural style transfer [8] on photo stylization. Left to right: input image, results from [8] using two different representative artworks as style images, results from [8] using the entire collection of the artist, and CycleGAN (ours)

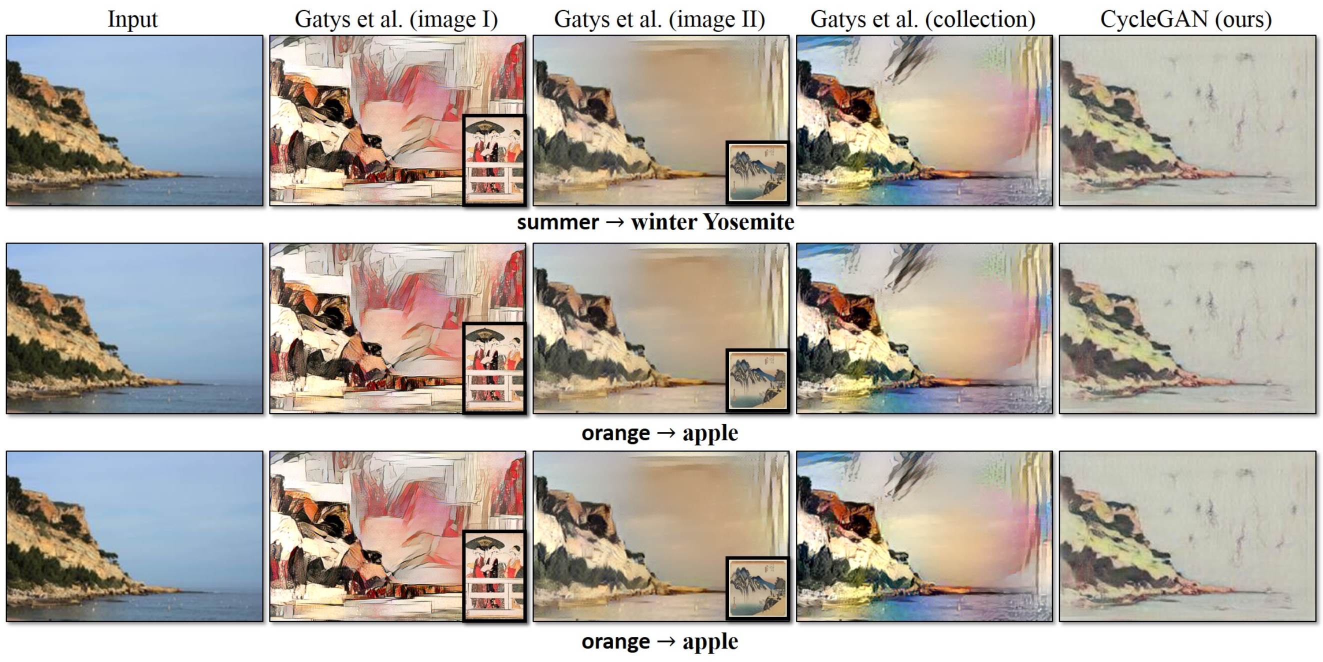

Figure 14: We compare our method with neural style transfer [8] on various applications. From top to bottom: orange→ apple, zebra→ horse, summer→ winter Yosemite, and Monet → photo. Left to right: input image, results from [8] using two different images from the target domains as style images, results from [8] using all the images from the target domain, and CycleGAN (ours)

References

[1] Y. Aytar, L. Castrejon, C. Vondrick, H. Pirsiavash, and A. Torralba. Cross-modal scene networks. *arXiv preprint arXiv:1610.09003*, 2016.

[2] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. *arXiv preprint arXiv:1612.05424*, 2016.

[3] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016.

[4] J. Donahue, P. Krähenbühl, and T. Darrell. Adversarial feature learning. *arXiv preprint arXiv:1605.09782*, 2016.

[5] V. Dumoulin, I. Belghazi, B. Poole, A. Lamb, M. Arjovsky, O. Mastropietro, and A. Courville. Adversarially learned inference. *arXiv preprint arXiv:1606.00704*, 2016.

[6] A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In *ICCV*, volume 2, pages 1033–1038. IEEE, 1999.

[7] D. Eigen and R. Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *ICCV*, pages 2650–2658, 2015.

[8] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. *CVPR*, 2016.

[9] C. Godard, O. Mac Aodha, and G. J. Brostow. Unsupervised monocular depth estimation with left-right consistency. *arXiv preprint arXiv:1609.03677*, 2016.

[10] I. Goodfellow. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016.

[11] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *NIPS*, 2014.

[12] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[13] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin. Image analogies. pages 327–340. ACM, 2001.

[14] Q.-X. Huang and L. Guibas. Consistent shape maps via semidefinite programming. In *Computer Graphics Forum*, volume 32, pages 177–186. Wiley Online Library, 2013.

[15] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. *arXiv preprint arXiv:1611.07004*, 2016.

[16] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, pages 694–711. Springer, 2016.

[17] L. Karacan, Z. Akata, A. Erdem, and E. Erdem. Learning to generate images of outdoor scenes from attributes and semantic layouts. *arXiv preprint arXiv:1612.00215*, 2016.

[18] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[19] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *ICLR*, 2014.

[20] P.-Y. Laffont, Z. Ren, X. Tao, C. Qian, and J. Hays. Transient attributes for high-level understanding and editing of outdoor scenes. *ACM Transactions on Graphics (TOG)*, 33(4):149, 2014.

[21] M.-Y. Liu, T. Breuel, and J. Kautz. Unsupervised image-to-image translation networks. *arXiv preprint arXiv:1703.00848*, 2017.

[22] M.-Y. Liu and O. Tuzel. Coupled generative adversarial networks. In *NIPS*, pages 469–477, 2016.

[23] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, pages 3431–3440, 2015.

[24] X. Mao, Q. Li, H. Xie, R. Y. Lau, and Z. Wang. Multi-class generative adversarial networks with the l2 loss function. *arXiv preprint arXiv:1611.04076*, 2016.

[25] M. Mathieu, C. Couprie, and Y. LeCun. Deep multi-scale video prediction beyond mean square error. *ICLR*, 2016.

[26] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. *CVPR*, 2016.

[27] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[28] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee. Generative adversarial text to image synthesis. *arXiv preprint arXiv:1605.05396*, 2016.

[29] R. Rosales, K. Achan, and B. J. Frey. Unsupervised image translation. In *iccv*, pages 472–478, 2003.

[30] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015.

[31] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. *arXiv preprint arXiv:1606.03498*, 2016.

[32] P. Sangkloy, J. Lu, C. Fang, F. Yu, and J. Hays. Scribbler: Controlling deep image synthesis with sketch and color. *arXiv preprint arXiv:1612.00835*, 2016.

[33] Y. Shih, S. Paris, F. Durand, and W. T. Freeman. Data-driven hallucination of different times of day from a single outdoor photo. *ACM Transactions on Graphics (TOG)*, 32(6):200, 2013.

[34] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb. Learning from simulated and unsupervised images through adversarial training. *arXiv preprint arXiv:1612.07828*, 2016.

[35] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[36] N. Sundaram, T. Brox, and K. Keutzer. Dense point trajectories by gpu-accelerated large displacement optical flow. In *ECCV*, pages 438–451. Springer, 2010.

[37] Y. Taigman, A. Polyak, and L. Wolf. Unsupervised cross-domain image generation. *arXiv preprint arXiv:1611.02200*, 2016.

[38] D. Turmukhambetov, N. D. Campbell, S. J. Prince, and J. Kautz. Modeling object appearance using context-conditioned component analysis. In *CVPR*, pages 4156–4164, 2015.

[39] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

[40] C. Vondrick, H. Pirsiavash, and A. Torralba. Generating videos with scene dynamics. In *NIPS*, pages 613–621, 2016.

[41] F. Wang, Q. Huang, and L. J. Guibas. Image co-segmentation via consistent functional maps. In *ICCV*, pages 849–856, 2013.

[42] X. Wang and A. Gupta. Generative image modeling using style and structure adversarial networks. *ECCV*, 2016.

[43] J. Wu, C. Zhang, T. Xue, B. Freeman, and J. Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *NIPS*, pages 82–90, 2016.

[44] S. Xie and Z. Tu. Holistically-nested edge detection. In *ICCV*, 2015.

[45] C. Zach, M. Klopschitz, and M. Pollefeys. Disambiguating visual relations using loop constraints. In *CVPR*, pages 1426–1433. IEEE, 2010.

[46] R. Zhang, P. Isola, and A. A. Efros. Colorful image colorization. *ECCV*, 2016.

[47] J. Zhao, M. Mathieu, and Y. LeCun. Energy-based generative adversarial network. *arXiv preprint arXiv:1609.03126*, 2016.

[48] T. Zhou, Y. Jae Lee, S. X. Yu, and A. A. Efros. Flowweb: Joint image set alignment by weaving consistent, pixel-wise correspondences. In *CVPR*, pages 1191–1200, 2015.

[49] T. Zhou, P. Krahenbuhl, M. Aubry, Q. Huang, and A. A. Efros. Learning dense correspondence via 3d-guided cycle consistency. In *CVPR*, pages 117–126, 2016.

[50] J.-Y. Zhu, P. Krähenbühl, E. Shechtman, and A. A. Efros. Generative visual manipulation on the natural image manifold. In *ECCV*, 2016.

7. Appendix

7.1. Training details

All networks were trained from scratch, and trained with learning rate of 0.0002 for the first 100 epochs and linearly decaying rate to zero for the next 100 epochs. Weights were initialized from a Gaussian distribution with mean 0 and standard deviation 0.02. All images were scaled to square except *Monet painting→photo*.

Citiscapes label↔Photo 2975 training images from the Cityscapes training set [3] with image size 128. We used the Cityscapes val set for testing.

Maps↔aerial photograph 1096 training images scraped from Google Maps [15] with image size 256 by 256. Images were sampled from in and around New York City. Data was then split into train and test about the median latitude of the sampling region (with a buffer region added to ensure that no training pixel appeared in the test set).

Horse↔Zebra and Apple↔Orange The images for each class were downloaded from ImageNet using keywords *wild horse*, *zebra*, *apple*, and *navel orange*. The images were scaled to size of 256px. The training set size of each class was horse: 939, zebra: 1177, apple: 996, orange: 1020.

Summer↔Winter Yosemite The images were downloaded using Flickr API. The summer Yosemite photos were searched by using the tag *yosemite* and the *datetaken* field between July and September. The winter Yosemite were search by using the tag *snow, yosemite* and *datetaken* between December and February. The black-and-white photos were pruned. The images were scaled to size of 256px. The training size of each class was summer: 1273, winter: 854.

Photo↔Art for style transfer The art images were downloaded from Wikiart.org by crawling. Some artworks that were sketches or too obscene were pruned by hand. The photos are downloaded from Flickr using combination of tags *landscape*, *landscapephotography*. The black-and-white photos were pruned. The images were scaled to size 256. The training set size of each class was Monet: 1074, Cezanne: 584, Van Gogh: 401, Ukiyo-e: 1433, Photographs:6853. The Monet dataset was especially pruned to include only landscape paintings, and Van Gogh included only their later works that represent their artistic styles.

Monet→Photo In order to achieve high resolution, random square crop of the rectangular images were used for training to conserve memory. To generate results, the images of width 512px with correct aspect ratio was passed to the generator network as input. The weight for the identity mapping loss was 0.5.

Renoir↔Selfie The art images were downlaoded from Wikiart.org, and the selfies were downloaded from Flickr using the tag *selfie*. The images were scaled to width of 512px keeping the aspect ratio. The training set size of each class was Renoir: 1405, Selfie: 6805. The identity mapping loss of weight 0.5 was used.

7.2. Network architectures

Code and models are available at `https://github.com/junyanz/CycleGAN`.

Generator architectures We adapt our architectures from Johnson et al. [16]. We use 6 blocks for 128×128 training images, and 9 blocks for 256×256 or higher-resolution training images. Below, we follow the naming convention used in the Johnson el al.'s Github repository[2]

Let `c7s1-k` denote a 7×7 Convolution-BatchNorm-ReLU layer with k filters and stride 1. `dk` denotes a 3×3 Convolution-BatchNorm-Dropout-ReLU layer with k filters, and stride 2. Reflection padding was used to reduce artifacts. `Rk` denotes a residual block that contains two 3×3 convolutional layers with the same number of filters on both layer. `uk` denotes a 3×3 fractional-strided-Convolution-BatchNorm-Dropout-ReLU layer with k filters, and stride $\frac{1}{2}$.

The network with 6 blocks consists of:
`c7s1-32,d64,d128,R128,R128,R128,R128,R128,R128,u64,u32,c7s1-3`

The network with 9 blocks consists of:
`c7s1-32,d64,d128,R128,R128,R128,R128,R128,R128,R128,R128,R128,u64,u32,c7s1-3`

Discriminator architectures For discriminator networks, we use 70×70 PatchGAN [15]. Let `Ck` denote a 4×4 Convolution-BatchNorm-LeakyReLU layer with k filters and stride 2. After the last layer, we apply a convolution andto produce a 1 dimensional output. We do not use BatchNorm for the first `C64` layer. We use leaky ReLUs with slope 0.2. The discriminator architecture is:
`C64-C128-C256-C512`

[2] `https://github.com/jcjohnson/fast-neural-style`.