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

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

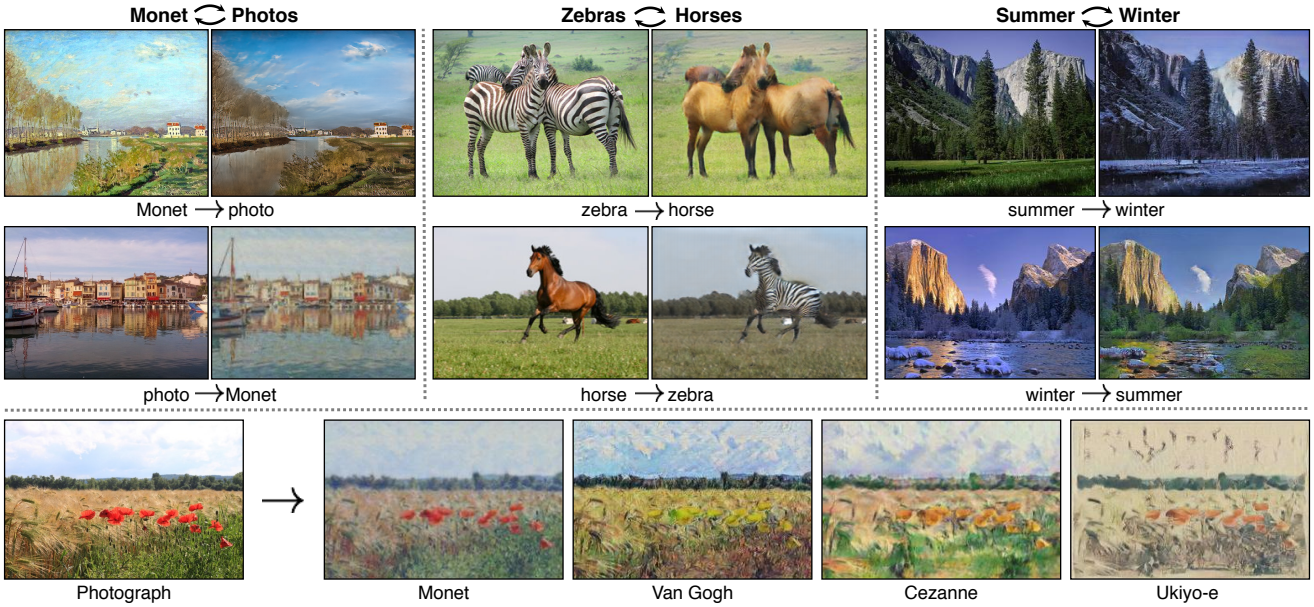

Figure 1: Given any two unordered image collections X and Y, our algorithm learns to automatically "translate" an image from one into the other and vice versa: *(left)* Monet paintings and landscape photos from Flickr; *(center)* zebras and horses from ImageNet; *(right)* summer and winter Yosemite photos from Flickr. Example application *(bottom)*: using a collection of paintings of famous artists, our method learns to render natural photographs into the respective styles.

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 enforce $F(G(X)) \approx X$ (and vice versa). Qualitative results are presented on several tasks where paired training data does not exist, including collection style transfer, object transfiguration, season transfer, photo enhancement, etc. Quantitative comparisons against several prior methods demonstrate the superiority of our approach.*

1. Introduction

What did Claude Monet see as he placed his easel by the bank of the Seine near Argenteuil on a lovely spring day in 1873 (Figure 1, top-left)? A color photograph, had it been invented, may have documented a crisp blue sky and a glassy river reflecting it. Monet conveyed his *impression* of this same scene through wispy brush strokes and a bright palette.

What if Monet had happened upon the little harbor in Cassis on a cool summer evening (Figure 1, bottom-left)? A brief stroll through a gallery of Monet paintings makes it possible to imagine how he would have rendered the scene: perhaps in pastel shades, with abrupt dabs of paint, 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 landscape photographs. We can reason about the stylistic differences between these

* indicates equal contribution

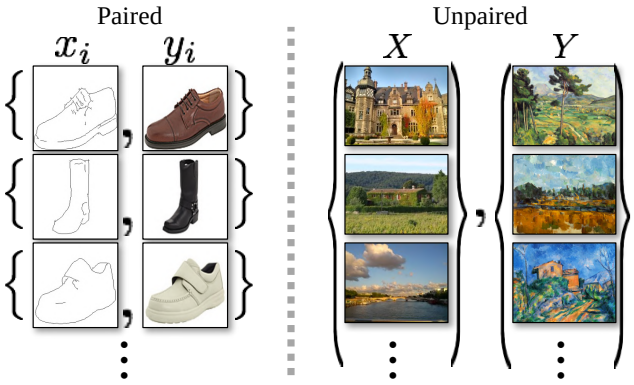

Figure 2: *Paired* training data (left) consists of training examples $\{x_i, y_i\}_{i=1}^N$, where the correspondence between x_i and y_i exists [22]. We instead consider *unpaired* training data (right), consisting of a source set $\{x_i\}_{i=1}^N$ ($x_i \in X$) and a target set $\{y_j\}_{j=1}^M$ ($y_j \in Y$), with no information provided as to which x_i matches which y_j.

two sets, and thereby imagine what a scene might look like if we were to "translate" it from one set into the other.

In this paper, we present a method that can learn to do the same: capturing special characteristics of one image collection and figuring out how these characteristics could be translated into the other image collection, all in the absence of any paired training examples.

This problem can be more broadly described as image-to-image translation [22], converting an image from one representation of a given scene, x, to another, y, e.g., grayscale to color, image to semantic labels, edge-map to photograph. Years of research in computer vision, image processing, computational photography, and graphics have produced powerful translation systems in the supervised setting, where example image pairs $\{x_i, y_i\}_{i=1}^N$ are available (Figure 2, left), e.g., [11, 19, 22, 23, 28, 33, 45, 56, 58, 62]. 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., [4]), 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., zebra↔horse, 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 scene – 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 $p_{data}(y)$ (in general, this requires G to be stochastic) [16]. The optimal G thereby translates the domain X to a domain $\hat{Y}$ distributed identically to Y. However, such a translation does not guarantee that an individual input x and output 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, we have found it 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 [15].

These issues call for adding more structure to our objective. Therefore, we 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 [3]. 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, and both mappings should be bijections. We apply this structural assumption by training both the mapping G and F simultaneously, and adding a *cycle consistency loss* [64] 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 to a wide range of applications, including collection style transfer, object transfiguration, season transfer and photo enhancement. 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. We provide both PyTorch and Torch implementations. Check out more results at our website.

2. Related work

Generative Adversarial Networks (GANs) [16, 63] have achieved impressive results in image generation [6, 39], image editing [66], and representation learning [39, 43, 37]. Recent methods adopt the same idea for conditional image generation applications, such as text2image [41], image inpainting [38], and future prediction [36], as well as to other domains like videos [54] and 3D data [57]. The key to GANs' success is the idea of an *adversarial loss* that forces the generated images to be, in principle, indistinguishable from real photos. This loss is particularly powerful for image generation tasks, as this is exactly the objective that much of computer graphics aims to optimize. We adopt an adversarial loss to learn the mapping such that the translated

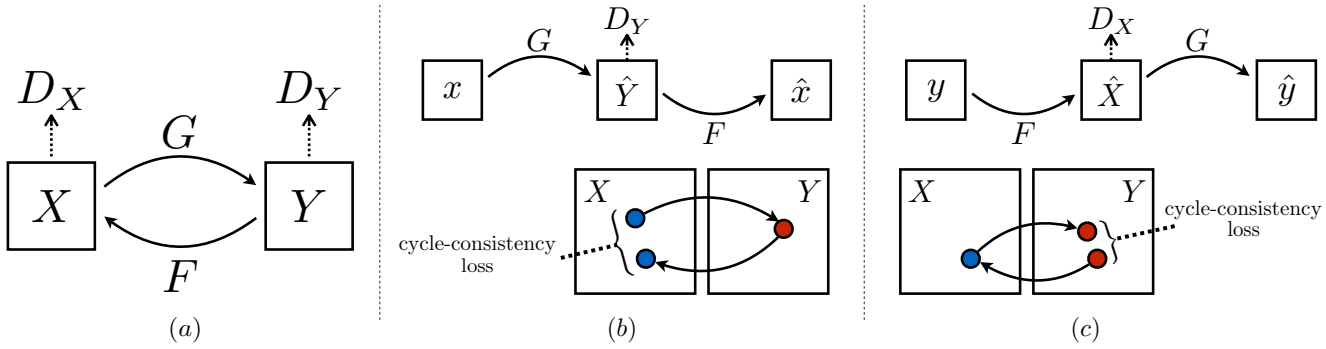

Figure 3: (a) Our model contains two mapping functions $G : X \rightarrow Y$ and $F : Y \rightarrow X$, and associated adversarial discriminators D_Y and D_X. D_Y encourages G to translate X into outputs indistinguishable from domain Y, and vice versa for D_X and F. To further regularize the mappings, we introduce two *cycle consistency losses* that capture the intuition that if we translate from one domain to the other and back again we should arrive at where we started: (b) forward cycle-consistency loss: $x \rightarrow G(x) \rightarrow F(G(x)) \approx x$, and (c) backward cycle-consistency loss: $y \rightarrow F(y) \rightarrow G(F(y)) \approx y$

images 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 [19], who employ a non-parametric texture model [10] on 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 (e.g., [33]). Our approach builds on the "pix2pix" framework of Isola et al. [22], which uses a conditional generative adversarial network [16] to learn a mapping from input to output images. Similar ideas have been applied to various tasks such as generating photographs from sketches [44] or from attribute and semantic layouts [25]. However, unlike the above prior work, we learn the mapping without paired training examples.

Unpaired Image-to-Image Translation Several other methods also tackle the unpaired setting, where the goal is to relate two data domains: X and Y. Rosales et al. [42] propose 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, CoGAN [32] 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. [31] extends the above framework with a combination of variational autoencoders [27] and generative adversarial networks [16]. Another line of concurrent work [46, 49, 2] encourages the input and output to share specific "content" features even though they may differ in "style". These methods 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 [46], and image feature space [49].

Unlike the above approaches, our formulation does not rely on any task-specific, predefined 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 prior and contemporary approaches in Section 5.1.

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 [24, 48]. In the language domain, verifying and improving translations via "back translation and reconciliation" is a technique used by human translators [3] (including, humorously, by Mark Twain [51]), as well as by machines [17]. More recently, higher-order cycle consistency has been used in structure from motion [61], 3D shape matching [21], cosegmentation [55], dense semantic alignment [65, 64], and depth estimation [14]. Of these, Zhou et al. [64] and Godard et al. [14] 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. Concurrent with our work, in these same proceedings, Yi et al. [59] independently use a similar objective for unpaired image-to-image translation, inspired by dual learning in machine translation [17].

Neural Style Transfer [13, 23, 52, 12] 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) based on matching the Gram matrix statistics of pre-trained deep features. Our primary focus, on the other hand, is learning the mapping between two image collections, 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→ photo, object transfiguration, etc. where single sample transfer methods do not perform well. We compare these two methods in Section 5.2.

3. Formulation

Our goal is to learn mapping functions between two domains X and Y given training samples $\{x_i\}_{i=1}^N$ where $x_i \in X$ and $\{y_j\}_{j=1}^M$ where $y_j \in Y$[1]. We denote the data distribution as $x \sim p_{data}(x)$ and $y \sim p_{data}(y)$. As illustrated in Figure 3 (a), our model includes two mappings $G : X \rightarrow Y$ and $F : Y \rightarrow X$. In addition, we introduce 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 contains two types of terms: *adversarial losses* [16] for matching the distribution of generated images to the data distribution in the target domain; and *cycle consistency losses* to prevent the learned mappings G and F from contradicting each other.

3.1. Adversarial Loss

We apply adversarial losses [16] 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 aims to minimize this objective against an adversary D that tries to maximize it, i.e., $\min_G \max_{D_Y} \mathcal{L}_{\text{GAN}}(G, D_Y, X, Y)$. We introduce a similar adversarial loss for the mapping function $F : Y \rightarrow X$ and its discriminator D_X as well: i.e., $\min_F \max_{D_X} \mathcal{L}_{\text{GAN}}(F, D_X, Y, X)$.

3.2. Cycle Consistency Loss

Adversarial training can, in theory, learn mappings G and F that produce outputs identically distributed as target domains Y and X respectively (strictly speaking, this requires G and F to be stochastic functions) [15]. 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, adversarial losses 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 possible 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 this *forward cycle consistency*. Similarly, as illustrated in Figure 3 (c), for each image y from domain Y, G and F should also satisfy *backward cycle consistency*: $y \rightarrow F(y) \rightarrow G(F(y)) \approx y$. We incentivize this behavior using a *cycle consistency loss*:

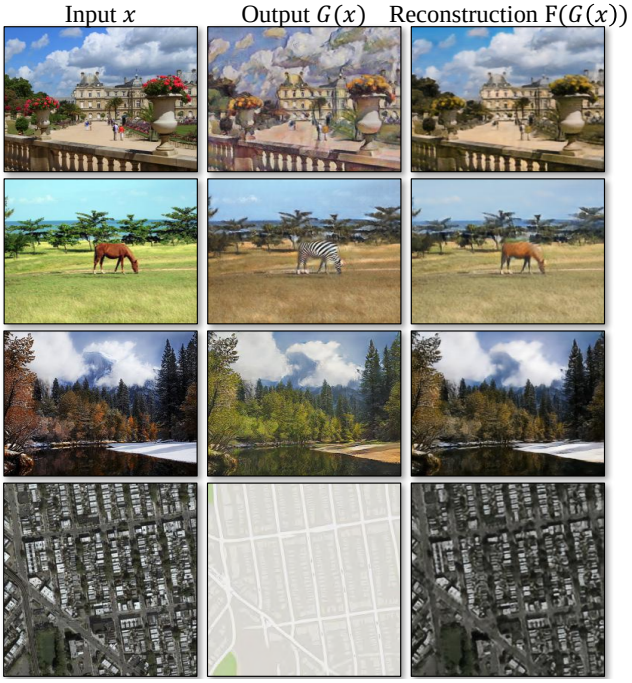

Figure 4: The input images x, output images $G(x)$ and the reconstructed images $F(G(x))$ from various experiments. From top to bottom: photo↔Cezanne, horses↔zebras, winter→summer Yosemite, aerial photos↔Google maps.

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

In preliminary experiments, we also tried replacing the L1 norm in this loss with an adversarial loss between $F(G(x))$ and x, and between $G(F(y))$ and y, but did not observe improved performance.

The behavior induced by the cycle consistency loss can be observed in Figure 4: the reconstructed images $F(G(x))$ end up matching closely to the input images x.

3.3. Full Objective

Our full objective is:

$$\begin{aligned}\mathcal{L}(G, F, D_X, D_Y) = &\mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) \\ &+ \mathcal{L}_{\text{GAN}}(F, D_X, Y, X) \\ &+ \lambda \mathcal{L}_{\text{cyc}}(G, F),\end{aligned} \tag{3}$$

[1] We often omit the subscript i and j for simplicity.

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

$$G^*, F^* = \arg\min_{G,F} \max_{D_x, D_Y} \mathcal{L}(G, F, D_X, D_Y). \quad (4)$$

Notice that our model can be viewed as training two "autoencoders" [20]: we learn one autoencoder $F \circ G : X \rightarrow X$ jointly with another $G \circ F : Y \rightarrow Y$. However, these autoencoders each have special internal structures: they map an image to itself via an intermediate representation that is a translation of the image into another domain. Such a setup can also be seen as a special case of "adversarial autoencoders" [34], which use an adversarial loss to train the bottleneck layer of an autoencoder to match an arbitrary target distribution. In our case, the target distribution for the $X \rightarrow X$ autoencoder is that of the domain Y.

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

4. Implementation

Network Architecture We adopt the architecture for our generative networks from Johnson et al. [23] who have shown impressive results for neural style transfer and super-resolution. This network contains three convolutions, several residual blocks [18], two fractionally-strided convolutions with stride $\frac{1}{2}$, and one convolution that maps features to RGB. We use 6 blocks for 128×128 images and 9 blocks for 256×256 and higher-resolution training images. Similar to Johnson et al. [23], we use instance normalization [53]. For the discriminator networks we use 70×70 PatchGANs [22, 30, 29], which aim to classify whether 70×70 overlapping image patches are real or fake. Such a patch-level discriminator architecture has fewer parameters than a full-image discriminator and can work on arbitrarily-sized images in a fully convolutional fashion [22].

Training details We apply two techniques from recent works to stabilize our model training procedure. First, for $\mathcal{L}_{\text{GAN}}$ (Equation 1), we replace the negative log likelihood objective by a least-squares loss [35]. This loss is more stable during training and generates higher quality results. In particular, for a GAN loss $\mathcal{L}_{\text{GAN}}(G, D, X, Y)$, we train the G to minimize $\mathbb{E}_{x \sim p_{\text{data}}(x)}[(D(G(x)) - 1)^2]$ and train the D to minimize $\mathbb{E}_{y \sim p_{\text{data}}(y)}[(D(y) - 1)^2] + \mathbb{E}_{x \sim p_{\text{data}}(x)}[D(G(x))^2]$.

Second, to reduce model oscillation [15], we follow Shrivastava et al.'s strategy [46] and update the discriminators using a history of generated images rather than the ones produced by the latest generators. We keep an image buffer that stores the 50 previously created images.

For all the experiments, we set $\lambda = 10$ in Equation 3. We use the Adam solver [26] with a batch size of 1. All networks were trained from scratch with a learning rate of 0.0002. We keep the same learning rate for the first 100 epochs and linearly decay the rate to zero over the next 100 epochs. Please see the appendix (Section 7) for more details about the datasets, architectures, and training procedures.

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 the adversarial loss and the cycle consistency loss and compare our full method against 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 refer to our method as `CycleGAN`. The PyTorch and Torch code, models, and full results can be found at our website.

5.1. Evaluation

Using the same evaluation datasets and metrics as "pix2pix" [22], we compare our method against several baselines both qualitatively and quantitatively. The tasks include semantic labels↔photo on the Cityscapes dataset [4], and map↔aerial photo on data scraped from Google Maps. We also perform ablation study on the full loss function.

5.1.1 Evaluation Metrics

AMT perceptual studies On the map↔aerial photo task, we run "real vs fake" perceptual studies on Amazon Mechanical Turk (AMT) to assess the realism of our outputs. We follow the same perceptual study protocol from Isola et al. [22], except we only gather data from 25 participants per algorithm we tested. Participants were shown a sequence of pairs of images, one a real photo or map and one fake (generated by our algorithm or a baseline), and asked to click on the image they thought was real. The first 10 trials of each session were practice and feedback was given as to whether the participant's response was correct or incorrect. The remaining 40 trials were used to assess the rate at which each algorithm fooled participants. Each session only tested a single algorithm, and participants were only allowed to complete a single session. The numbers we report here are not directly comparable to those in [22] as our ground truth images were processed slightly differently [2] and the participant pool we tested may be differently dis-

[2] We train all the models on 256×256 images while in pix2pix [22], the model was trained on 256×256 patches of 512×512 images, and

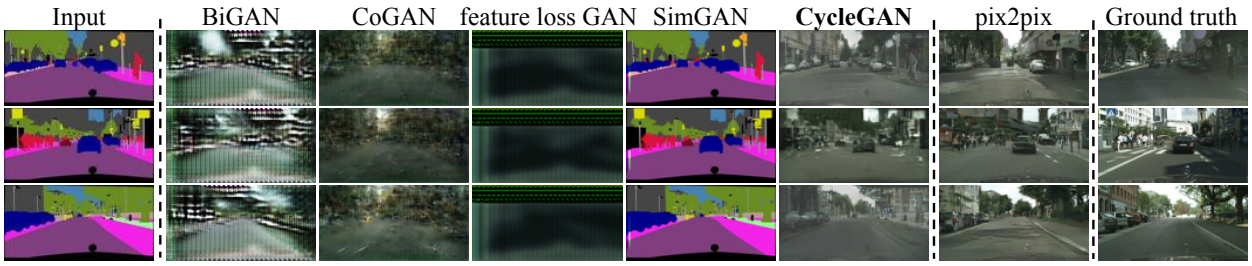

Figure 5: Different methods for mapping labels↔photos trained on Cityscapes images. From left to right: input, BiGAN/ALI [7, 9], CoGAN [32], feature loss + GAN, SimGAN [46], CycleGAN (ours), pix2pix [22] trained on paired data, and ground truth.

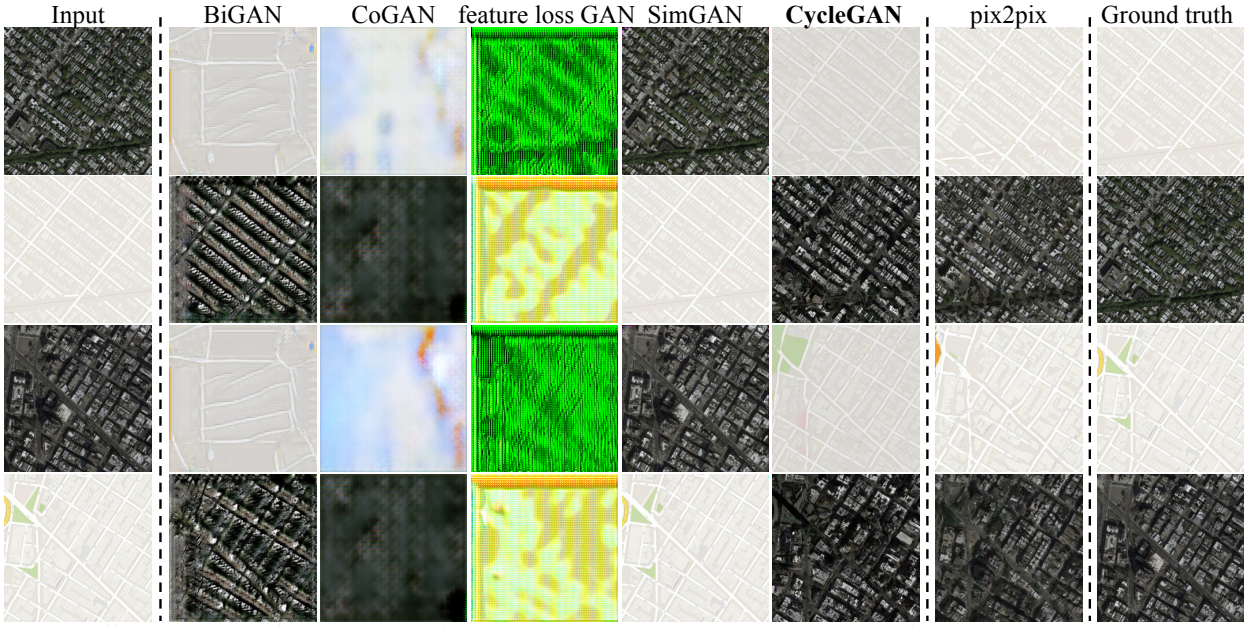

Figure 6: Different methods for mapping aerial photos↔maps on Google Maps. From left to right: input, BiGAN/ALI [7, 9], CoGAN [32], feature loss + GAN, SimGAN [46], CycleGAN (ours), pix2pix [22] trained on paired data, and ground truth.

tributed from those tested in [22] (due to running the experiment at a different date and time). Therefore, our numbers should only be used to compare our current method against the baselines (which were run under identical conditions), rather than against [22].

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 [22], and use it to evaluate the Cityscapes labels→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 [33]). The FCN predicts a label map for a generated photo. This label map can then be compared against the input ground truth labels using standard semantic segmentation metrics described below. The intuition is that if we generate a photo from a label map of "car on the road", then we have succeeded if the FCN applied to the generated photo detects "car on the road".

Semantic segmentation metrics To evaluate the performance of photo→labels, we use the standard metrics from the Cityscapes benchmark [4], including per-pixel accuracy, per-class accuracy, and mean class Intersection-Over-Union (Class IOU) [4].

5.1.2 Baselines

CoGAN [32] This method learns one GAN generator for domain X and one for domain Y, with tied weights on the first few layers for shared latent representations. Translation from X to Y can be achieved by finding a latent representation that generates image X and then rendering this latent representation into style Y.

SimGAN [46] Like our method, Shrivastava et al.[46] uses an adversarial loss to train a translation from X to Y.

run convolutionally on the 512×512 images at test time. We choose 256×256 in our experiments as many baselines cannot scale up to high-resolution images, and CoGAN cannot be tested fully convolutionally.

Loss	Map → Photo % Turkers labeled *real*	Photo → Map % Turkers labeled *real*
CoGAN [32]	0.6% ± 0.5%	0.9% ± 0.5%
BiGAN/ALI [9, 7]	2.1% ± 1.0%	1.9% ± 0.9%
SimGAN [46]	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 at 256 × 256 resolution.

Loss	Per-pixel acc.	Per-class acc.	Class IOU
CoGAN [32]	0.40	0.10	0.06
BiGAN/ALI [9, 7]	0.19	0.06	0.02
SimGAN [46]	0.20	0.10	0.04
Feature loss + GAN	0.06	0.04	0.01
CycleGAN (ours)	**0.52**	**0.17**	**0.11**
pix2pix [22]	0.71	0.25	0.18

Table 2: FCN-scores for different methods, evaluated on Cityscapes labels→photo.

Loss	Per-pixel acc.	Per-class acc.	Class IOU
CoGAN [32]	0.45	0.11	0.08
BiGAN/ALI [9, 7]	0.41	0.13	0.07
SimGAN [46]	0.47	0.11	0.07
Feature loss + GAN	0.50	0.10	0.06
CycleGAN (ours)	**0.58**	**0.22**	**0.16**
pix2pix [22]	0.85	0.40	0.32

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

The regularization term $\|x - G(x)\|_1$ i s used to penalize making large changes at pixel level.

Feature loss + GAN We also test a variant of SimGAN [46] where the L1 loss is computed over deep image features using a pretrained network (VGG-16 `relu4_2` [47]), rather than over RGB pixel values. Computing distances in deep feature space, like this, is also sometimes referred to as using a "perceptual loss" [8, 23].

BiGAN/ALI [9, 7] Unconditional GANs [16] learn a generator $G : Z \rightarrow X$, that maps a random noise z to an image x. The BiGAN [9] and ALI [7] propose to also learn the inverse mapping function $F : X \rightarrow Z$. Though they were 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 [22] We also compare against pix2pix [22], which is trained on paired data, to see how close we can get to this "upper bound" without using any paired data.

For a fair comparison, we implement all the baselines using the same architecture and details as our method, except for CoGAN [32]. CoGAN builds on generators that produce images from a shared latent representation, which is incompatible with our image-to-image network. We use the public implementation of CoGAN instead.

5.1.3 Comparison against baselines

As can be seen in Figure 5 and Figure 6, we were unable to achieve compelling results with any of the baselines. Our method, on the other hand, can produce translations that are often of similar quality to the fully supervised pix2pix.

Table 1 reports performance regarding the AMT perceptual realism task. Here, we see that our method can fool participants on around a quarter of trials, in both the maps→aerial photos direction and the aerial photos→maps direction at 256 × 256 resolution[3]. All the baselines almost never fooled participants.

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

Loss	Per-pixel acc.	Per-class acc.	Class IOU
Cycle alone	0.22	0.07	0.02
GAN alone	0.51	0.11	0.08
GAN + forward cycle	**0.55**	**0.18**	**0.12**
GAN + backward cycle	0.39	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→photo.

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 photo→labels for different losses, evaluated on Cityscapes.

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 the cycle loss in only one direction: GAN + forward cycle loss $\mathbb{E}_{x \sim p_{\text{data}}(x)}[\|F(G(x)) - x\|_1]$, or GAN + backward cycle loss $\mathbb{E}_{y \sim p_{\text{data}}(y)}[\|G(F(y)) - y\|_1]$ (Equation 2) and find that it often incurs training instability and causes mode collapse, especially for the direction of the mapping that was removed. Figure 7 shows several qualitative examples.

5.1.5 Image reconstruction quality

In Figure 4, we show a few random samples of the reconstructed images $F(G(x))$. We observed that the reconstructed images were often close to the original inputs x, at both training and testing time, even in cases where one domain represents significantly more diverse information, such as map↔aerial photos.

[3] We also train CycleGAN and pix2pix at 512 × 512 resolution, and observe the comparable performance: maps→aerial photos: CycleGAN: 37.5% ± 3.6% and pix2pix: 33.9% ± 3.1%; aerial photos→maps: CycleGAN: 16.5% ± 4.1% and pix2pix: 8.5% ± 2.6%

Figure 7: 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-consistency loss ($F(G(x)) \approx x$), GAN + backward cycle-consistency loss ($G(F(y)) \approx y$), CycleGAN (our full method), 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 the input photo.

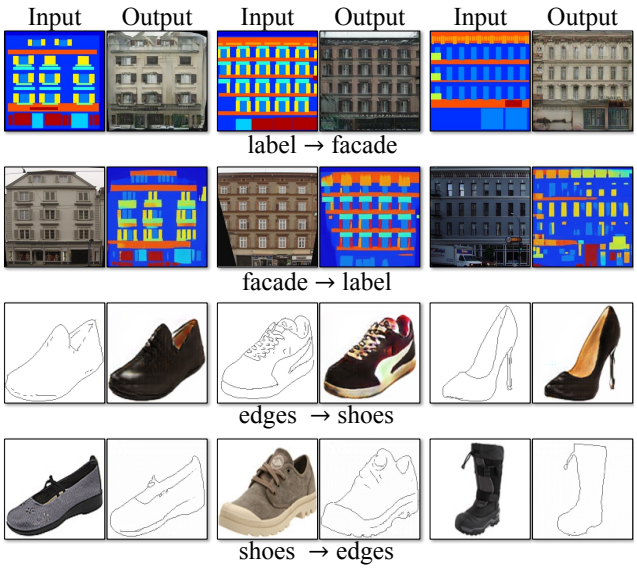

Figure 8: Example results of CycleGAN on paired datasets used in "pix2pix" [22] such as architectural labels↔photos and edges↔shoes.

5.1.6 Additional results on paired datasets

Figure 8 shows some example results on other paired datasets used in "pix2pix" [22], such as architectural labels↔photos from the CMP Facade Database [40], and edges↔shoes from the UT Zappos50K dataset [60]. The image quality of our results is close to those produced by the fully supervised pix2pix while our method learns the mapping without paired supervision.

5.2. Applications

We demonstrate our method on several applications where paired training data does not exist. Please refer to the appendix (Section 7) for more details about the datasets. We observe that translations on training data are often more appealing than those on test data, and full results of all applications on both training and test data can be viewed on our project website.

Collection style transfer (Figure 10 and Figure 11) We train the model on landscape photographs downloaded from Flickr and WikiArt. Unlike recent work on "neural style transfer" [13], our method learns to mimic the style of an entire *collection* 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.

Object transfiguration (Figure 13) The model is trained to translate one object class from ImageNet [5] to another (each class contains around 1000 training images). Turmukhambetov et al. [50] propose 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 13) 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 encourage the mapping to preserve color composition between the input and output. In particular, we adopt the technique of Taigman et al. [49] and regularize the generator to be near an identity mapping when real samples of the target domain are provided as the input to the generator: i.e., $\mathcal{L}_{\text{identity}}(G, F) = \mathbb{E}_{y \sim p_{\text{data}}(y)}[\|G(y) - y\|_1] + \mathbb{E}_{x \sim p_{\text{data}}(x)}[\|F(x) - x\|_1]$.

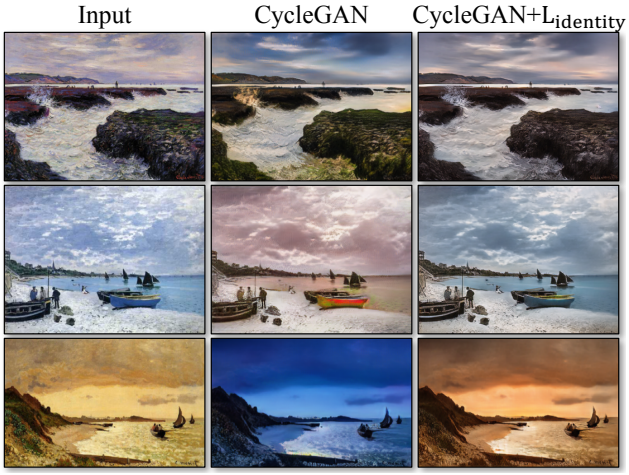

Figure 9: The effect of the *identity mapping loss* on Monet's painting→ photos. From left to right: input paintings, CycleGAN without identity mapping loss, CycleGAN with identity mapping loss. The identity mapping loss helps preserve the color of the input paintings.

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 a mapping may be equally valid under the adversarial loss and cycle consistency loss. The effect of this *identity mapping loss* are shown in Figure 9.

In Figure 12, we show additional results translating Monet's paintings to photographs. This figure and Figure 9 show results on paintings that were included in the *training set*, whereas for all other experiments in the paper, we only evaluate and show test set results. Because the training set does not include paired data, coming up with a plausible translation for a training set painting is a nontrivial task. Indeed, since Monet is no longer able to create new paintings, generalization to unseen, "test set", paintings is not a pressing problem.

Photo enhancement (Figure 14) We show that our method can be used to generate photos with shallower depth of field. We train the model on flower photos downloaded from Flickr. The source domain consists of flower photos taken by smartphones, which usually have deep DoF due to a small aperture. The target contains photos captured by DSLRs with a larger aperture. Our model successfully generates photos with shallower depth of field from the photos taken by smartphones.

Comparison with Gatys et al. [13] In Figure 15, we compare our results with neural style transfer [13] on photo stylization. For each row, we first use two representative artworks as the style images for [13]. Our method, on the other hand, can produce photos in the style of entire *collection*. To compare against neural style transfer of an entire collection, we compute the average Gram Matrix across the target domain and use this matrix to transfer the "average style" with Gatys et al [13].

Figure 16 demonstrates similar comparisons for other translation tasks. We observe that Gatys et al. [13] requires finding target style images that closely match the desired output, but still often fails to produce photorealistic results, while our method succeeds to generate natural-looking results, similar to the target domain.

6. Limitations and Discussion

Although our method can achieve compelling results in many cases, the results are far from uniformly positive. Figure 17 shows several typical failure cases. On translation tasks that involve color and texture changes, as 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 into making minimal changes to the input (Figure 17). This failure might be caused by our generator architectures which are tailored for good performance on the appearance changes. Handling more varied and extreme transformations, especially geometric changes, is an important problem for future work.

Some failure cases are caused by the distribution characteristics of the training datasets. For example, our method has got confused in the horse → zebra example (Figure 17, right), because our model was trained on the *wild horse* and *zebra* synsets of ImageNet, which does not contain images of a person riding a horse or zebra.

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 sometimes permutes the labels for tree and building in the output of the photos→labels task. Resolving this ambiguity may require some form of weak semantic supervision. Integrating weak or semi-supervised data may lead to substantially more powerful translators, still 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, Richard Zhang, and Tinghui Zhou for many helpful comments. This work was supported in part by NSF SMA-1514512, NSF IIS-1633310, a Google Research Award, Intel Corp, and hardware donations from NVIDIA. JYZ is supported by the Facebook Graduate Fellowship and TP is supported by the Samsung Scholarship. The photographs used for style transfer were taken by AE, mostly in France.

Figure 10: Collection style transfer I: we transfer input images into the artistic styles of Monet, Van Gogh, Cezanne, and Ukiyo-e. Please see our website for additional examples.

Figure 11: Collection style transfer II: we transfer input images into the artistic styles of Monet, Van Gogh, Cezanne, Ukiyo-e. Please see our website for additional examples.

Figure 12: Relatively successful results on mapping Monet's paintings to a photographic style. Please see our website for additional examples.

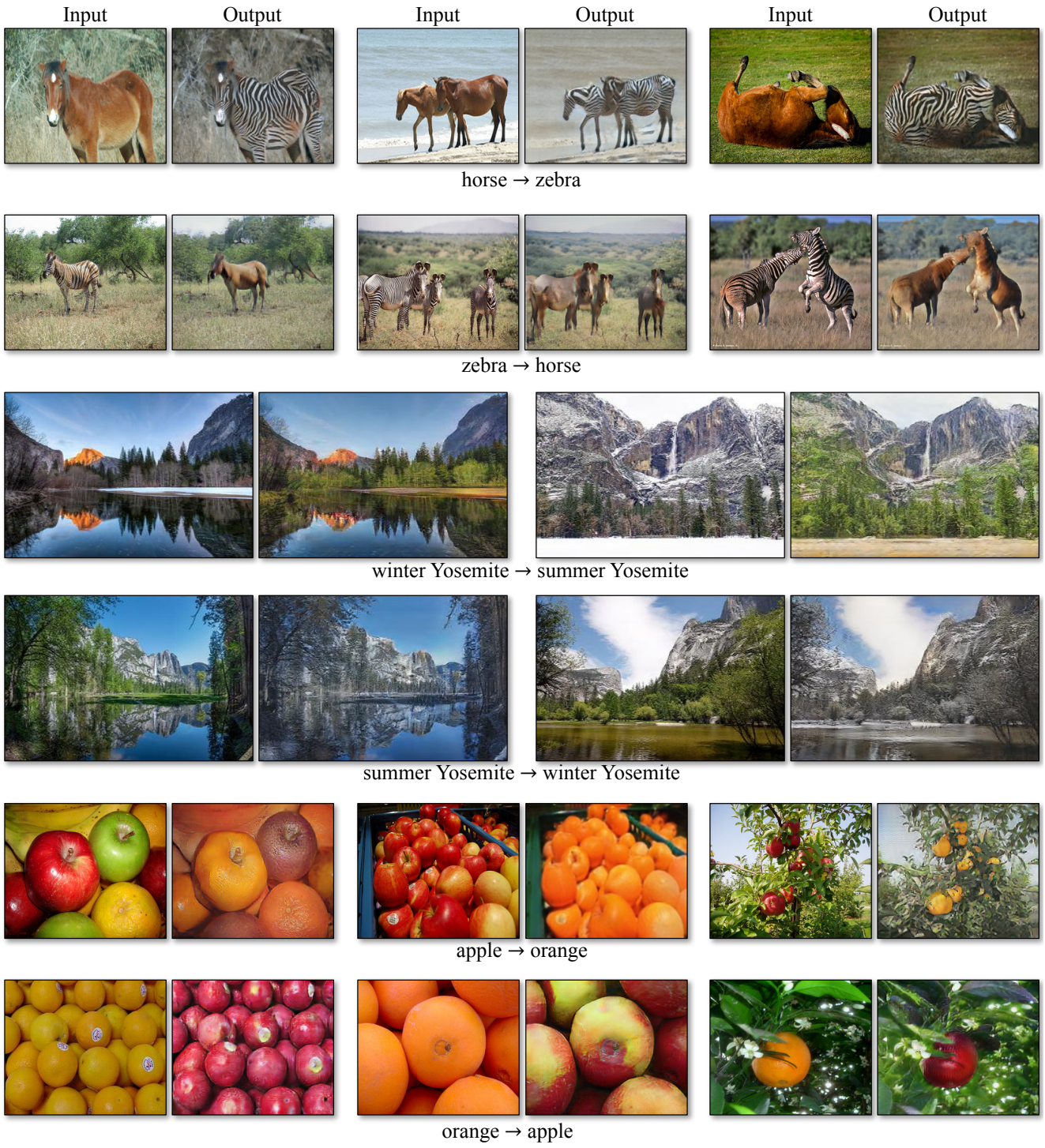

Figure 13: Our method applied to several translation problems. These images are selected as relatively successful results – please see our website for more comprehensive and random results. In the top two rows, we show results on object transfiguration between horses and zebras, trained on 939 images from the *wild horse* class and 1177 images from the *zebra* class in Imagenet [5]. Also check out the horse→zebra demo video. The middle two rows show results on season transfer, trained on winter and summer photos of Yosemite from Flickr. In the bottom two rows, we train our method on 996 *apple* images and 1020 *navel orange* images from ImageNet.

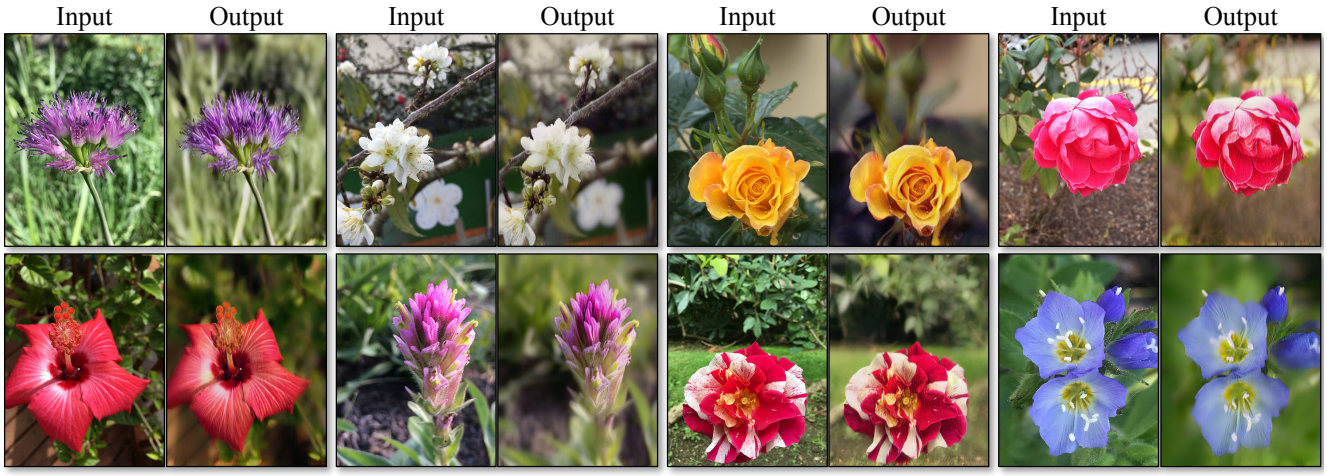

Figure 14: Photo enhancement: mapping from a set of smartphone snaps to professional DSLR photographs, the system often learns to produce shallow focus. Here we show some of the most successful results in our test set – average performance is considerably worse. Please see our website for more comprehensive and random examples.

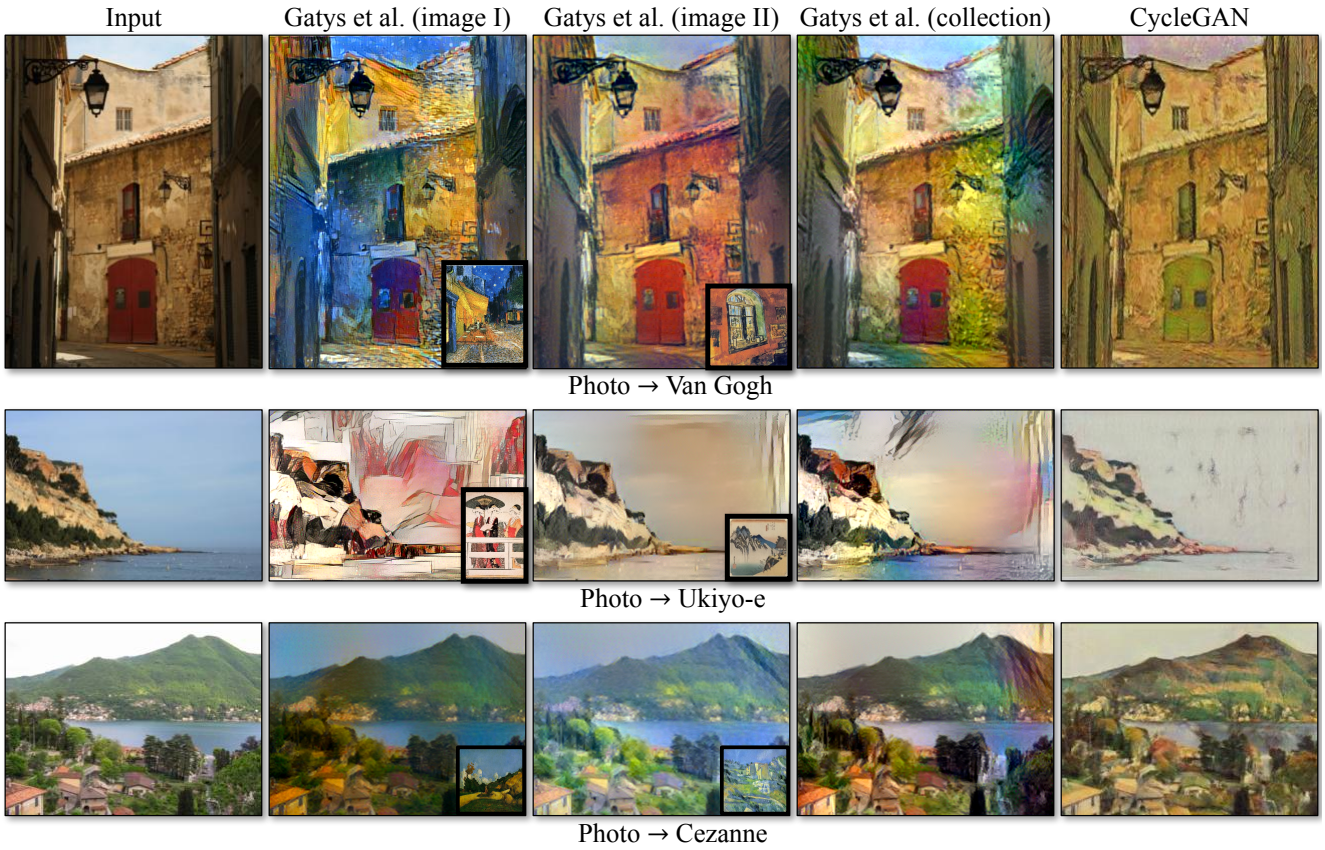

Figure 15: We compare our method with neural style transfer [13] on photo stylization. Left to right: input image, results from Gatys et al. [13] using two different representative artworks as style images, results from Gatys et al. [13] using the entire collection of the artist, and CycleGAN (ours).

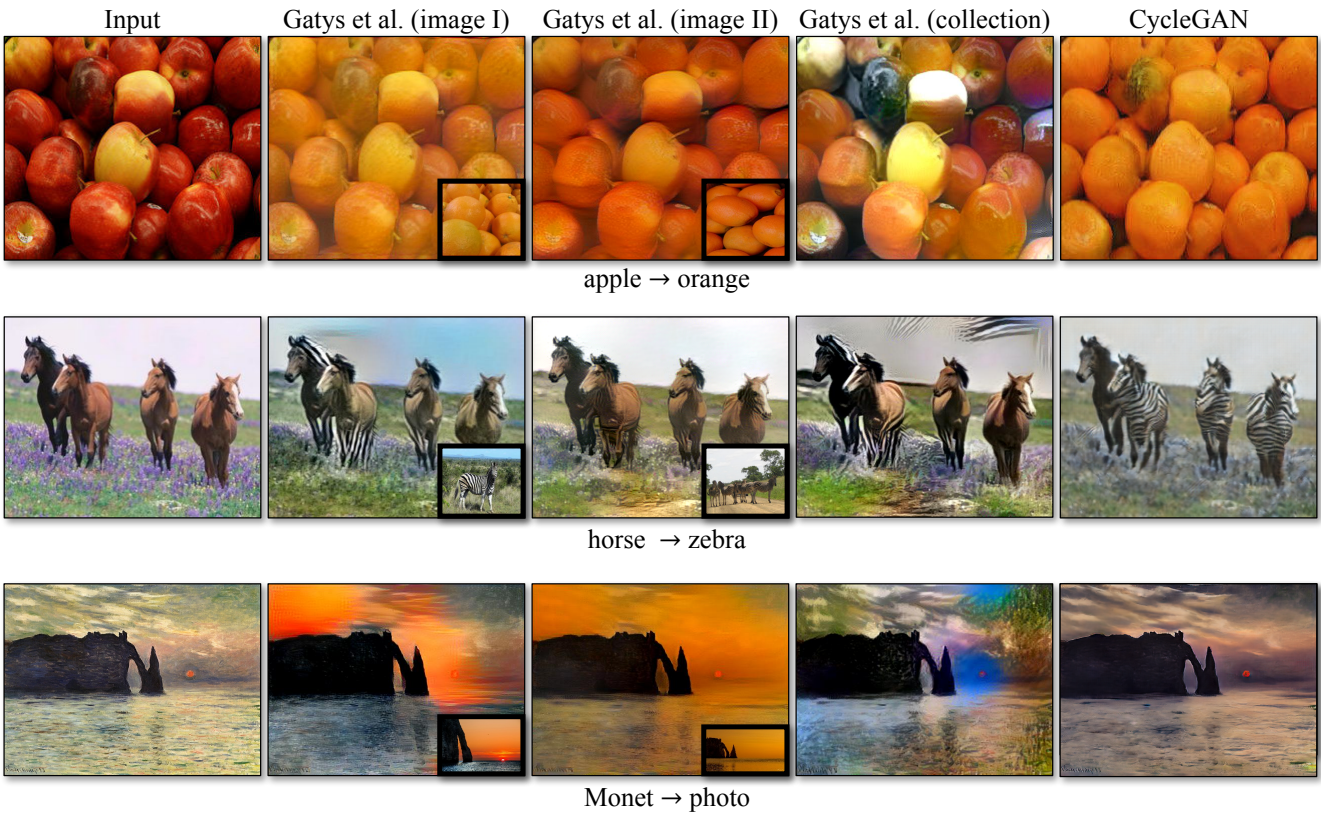

Figure 16: We compare our method with neural style transfer [13] on various applications. From top to bottom: apple→orange, horse→zebra, and Monet→photo. Left to right: input image, results from Gatys et al. [13] using two different images as style images, results from Gatys et al. [13] using all the images from the target domain, and CycleGAN (ours).

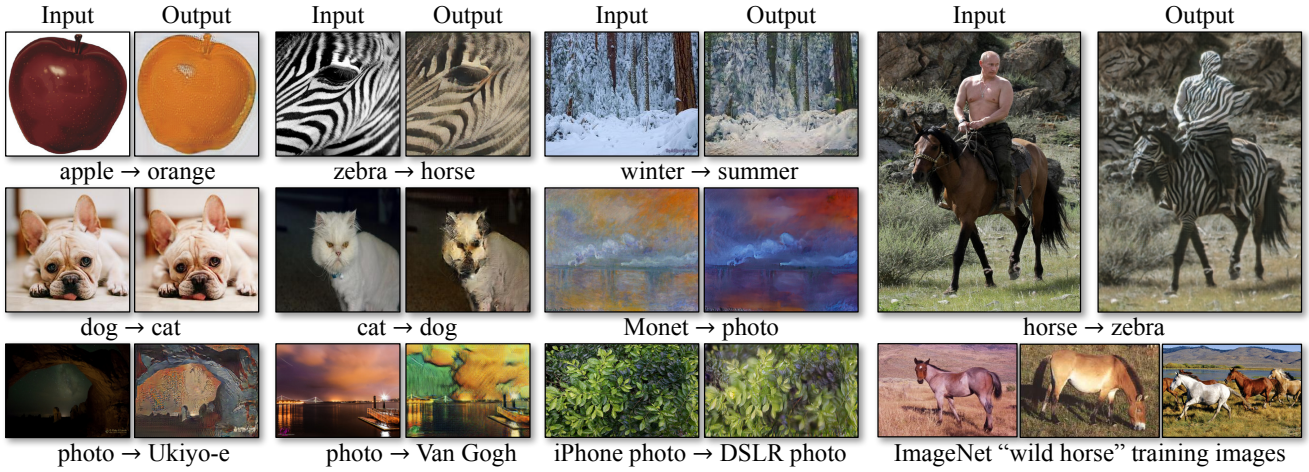

Figure 17: Typical failure cases of our method. Left: in the task of dog→cat transfiguration, CycleGAN can only make minimal changes to the input. Right: CycleGAN also fails in this horse → zebra example as our model has not seen images of horseback riding during training. Please see our website for more comprehensive results.

References

[1] Y. Aytar, L. Castrejon, C. Vondrick, H. Pirsiavash, and A. Torralba. Cross-modal scene networks. *PAMI*, 2016. 3

[2] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. In *CVPR*, 2017. 3

[3] R. W. Brislin. Back-translation for cross-cultural research. *Journal of cross-cultural psychology*, 1(3):185–216, 1970. 2, 3

[4] 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. 2, 5, 6, 18

[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 8, 13, 18

[6] E. L. Denton, S. Chintala, R. Fergus, et al. Deep generative image models using a laplacian pyramid of adversarial networks. In *NIPS*, 2015. 2

[7] J. Donahue, P. Krähenbühl, and T. Darrell. Adversarial feature learning. In *ICLR*, 2017. 6, 7

[8] A. Dosovitskiy and T. Brox. Generating images with perceptual similarity metrics based on deep networks. In *NIPS*, 2016. 7

[9] V. Dumoulin, I. Belghazi, B. Poole, A. Lamb, M. Arjovsky, O. Mastropietro, and A. Courville. Adversarially learned inference. In *ICLR*, 2017. 6, 7

[10] A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In *ICCV*, 1999. 3

[11] D. Eigen and R. Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *ICCV*, 2015. 2

[12] L. A. Gatys, M. Bethge, A. Hertzmann, and E. Shechtman. Preserving color in neural artistic style transfer. *arXiv preprint arXiv:1606.05897*, 2016. 3

[13] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. *CVPR*, 2016. 3, 8, 9, 14, 15

[14] C. Godard, O. Mac Aodha, and G. J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *CVPR*, 2017. 3

[15] I. Goodfellow. NIPS 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016. 2, 4, 5

[16] 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. 2, 3, 4, 7

[17] D. He, Y. Xia, T. Qin, L. Wang, N. Yu, T. Liu, and W.-Y. Ma. Dual learning for machine translation. In *NIPS*, 2016. 3

[18] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 5

[19] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin. Image analogies. In *SIGGRAPH*, 2001. 2, 3

[20] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006. 5

[21] Q.-X. Huang and L. Guibas. Consistent shape maps via semidefinite programming. In *Symposium on Geometry Processing*, 2013. 3

[22] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017. 2, 3, 5, 6, 7, 8, 18

[23] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, 2016. 2, 3, 5, 7, 18

[24] Z. Kalal, K. Mikolajczyk, and J. Matas. Forward-backward error: Automatic detection of tracking failures. In *ICPR*, 2010. 3

[25] 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. 3

[26] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 5

[27] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *ICLR*, 2014. 3

[28] P.-Y. Laffont, Z. Ren, X. Tao, C. Qian, and J. Hays. Transient attributes for high-level understanding and editing of outdoor scenes. *ACM TOG*, 33(4):149, 2014. 2

[29] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR*, 2017. 5

[30] C. Li and M. Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks. *ECCV*, 2016. 5

[31] M.-Y. Liu, T. Breuel, and J. Kautz. Unsupervised image-to-image translation networks. In *NIPS*, 2017. 3

[32] M.-Y. Liu and O. Tuzel. Coupled generative adversarial networks. In *NIPS*, 2016. 3, 6, 7

[33] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 2, 3, 6

[34] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey. Adversarial autoencoders. In *ICLR*, 2016. 5

[35] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. P. Smolley. Least squares generative adversarial networks. In *CVPR*. IEEE, 2017. 5

[36] M. Mathieu, C. Couprie, and Y. LeCun. Deep multi-scale video prediction beyond mean square error. In *ICLR*, 2016. 2

[37] M. F. Mathieu, J. Zhao, A. Ramesh, P. Sprechmann, and Y. LeCun. Disentangling factors of variation in deep representation using adversarial training. In *NIPS*, 2016. 2

[38] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. *CVPR*, 2016. 2

[39] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *ICLR*, 2016. 2

[40] R. Š. Radim Tyleček. Spatial pattern templates for recognition of objects with regular structure. In *Proc. GCPR*, Saarbrucken, Germany, 2013. 8, 18

[41] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee. Generative adversarial text to image synthesis. In *ICML*, 2016. 2

[42] R. Rosales, K. Achan, and B. J. Frey. Unsupervised image translation. In *ICCV*, 2003. 3

[43] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training GANs. In *NIPS*, 2016. 2

[44] P. Sangkloy, J. Lu, C. Fang, F. Yu, and J. Hays. Scribbler: Controlling deep image synthesis with sketch and color. In *CVPR*, 2017. 3

[45] Y. Shih, S. Paris, F. Durand, and W. T. Freeman. Data-driven hallucination of different times of day from a single outdoor photo. *ACM TOG*, 32(6):200, 2013. 2

[46] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb. Learning from simulated and unsupervised images through adversarial training. In *CVPR*, 2017. 3, 5, 6, 7

[47] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 7

[48] N. Sundaram, T. Brox, and K. Keutzer. Dense point trajectories by gpu-accelerated large displacement optical flow. In *ECCV*, 2010. 3

[49] Y. Taigman, A. Polyak, and L. Wolf. Unsupervised cross-domain image generation. In *ICLR*, 2017. 3, 8

[50] D. Turmukhambetov, N. D. Campbell, S. J. Prince, and J. Kautz. Modeling object appearance using context-conditioned component analysis. In *CVPR*, 2015. 8

[51] M. Twain. The jumping frog: in english, then in french, and then clawed back into a civilized language once more by patient. *Unremunerated Toil*, 3, 1903. 3

[52] D. Ulyanov, V. Lebedev, A. Vedaldi, and V. Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. In *ICML*, 2016. 3

[53] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016. 5

[54] C. Vondrick, H. Pirsiavash, and A. Torralba. Generating videos with scene dynamics. In *NIPS*, 2016. 2

[55] F. Wang, Q. Huang, and L. J. Guibas. Image co-segmentation via consistent functional maps. In *ICCV*, 2013. 3

[56] X. Wang and A. Gupta. Generative image modeling using style and structure adversarial networks. In *ECCV*, 2016. 2

[57] 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*, 2016. 2

[58] S. Xie and Z. Tu. Holistically-nested edge detection. In *ICCV*, 2015. 2

[59] Z. Yi, H. Zhang, T. Gong, Tan, and M. Gong. Dualgan: Unsupervised dual learning for image-to-image translation. In *ICCV*, 2017. 3

[60] A. Yu and K. Grauman. Fine-grained visual comparisons with local learning. In *CVPR*, 2014. 8, 18

[61] C. Zach, M. Klopschitz, and M. Pollefeys. Disambiguating visual relations using loop constraints. In *CVPR*, 2010. 3

[62] R. Zhang, P. Isola, and A. A. Efros. Colorful image colorization. In *ECCV*, 2016. 2

[63] J. Zhao, M. Mathieu, and Y. LeCun. Energy-based generative adversarial network. In *ICLR*, 2017. 2

[64] T. Zhou, P. Krahenbuhl, M. Aubry, Q. Huang, and A. A. Efros. Learning dense correspondence via 3d-guided cycle consistency. In *CVPR*, 2016. 2, 3

[65] T. Zhou, Y. J. Lee, S. Yu, and A. A. Efros. Flowweb: Joint image set alignment by weaving consistent, pixel-wise correspondences. In *CVPR*, 2015. 3

[66] J.-Y. Zhu, P. Krähenbühl, E. Shechtman, and A. A. Efros. Generative visual manipulation on the natural image manifold. In *ECCV*, 2016. 2

7. Appendix

7.1. Training details

We train our networks from scratch, with a learning rate of 0.0002. In practice, we divide the objective by 2 while optimizing D, which slows down the rate at which D learns, relative to the rate of G. We keep the same learning rate for the first 100 epochs and linearly decay the rate to zero over the next 100 epochs. Weights are initialized from a Gaussian distribution $\mathcal{N}(0, 0.02)$.

Cityscapes label↔Photo 2975 training images from the Cityscapes training set [4] with image size 128×128. We used the Cityscapes val set for testing.

Maps↔aerial photograph 1096 training images were scraped from Google Maps [22] with image size 256×256. 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).

Architectural facades labels↔photo 400 training images from the CMP Facade Database [40].

Edges→shoes around $50,000$ training images from UT Zappos50K dataset [60]. The model was trained for 5 epochs.

Horse↔Zebra and Apple↔Orange We downloaded the images from ImageNet [5] using keywords *wild horse*, *zebra*, *apple*, and *navel orange*. The images were scaled to 256×256 pixels. The training set size of each class: 939 (horse), 1177 (zebra), 996 (apple), and 1020 (orange).

Summer↔Winter Yosemite The images were downloaded using Flickr API with the tag *yosemite* and the *date-taken* field. Black-and-white photos were pruned. The images were scaled to 256×256 pixels. The training size of each class: 1273 (summer) and 854 (winter).

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

Monet's paintings→photos To achieve high resolution while conserving memory, we used random square crops of the original images for training. To generate results, we passed images of width 512 pixels with correct aspect ratio to the generator network as input. The weight for the identity mapping loss was 0.5λ where λ was the weight for cycle consistency loss. We set $\lambda = 10$.

Flower photo enhancement Flower images taken on smartphones were downloaded from Flickr by searching for the photos taken by *Apple iPhone 5, 5s, or 6*, with search text *flower*. DSLR images with shallow DoF were also downloaded from Flickr by search tag *flower, dof*. The images were scaled to 360 pixels by width. The identity mapping loss of weight 0.5λ was used. The training set size of the smartphone and DSLR dataset were 1813 and 3326, respectively. We set $\lambda = 10$.

7.2. Network architectures

We provide both PyTorch and Torch implementations.

Generator architectures We adopt our architectures from Johnson et al. [23]. We use 6 residual blocks for 128×128 training images, and 9 residual blocks for 256×256 or higher-resolution training images. Below, we follow the naming convention used in the Johnson et al.'s Github repository.

Let `c7s1-k` denote a 7×7 Convolution-InstanceNorm-ReLU layer with k filters and stride 1. `dk` denotes a 3×3 Convolution-InstanceNorm-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-InstanceNorm-ReLU layer with k filters and stride $\frac{1}{2}$.

The network with 6 residual blocks consists of:

```
c7s1-64,d128,d256,R256,R256,R256,
R256,R256,R256,u128,u64,c7s1-3
```

The network with 9 residual blocks consists of:

```
c7s1-64,d128,d256,R256,R256,R256,
R256,R256,R256,R256,R256,R256,u128
u64,c7s1-3
```

Discriminator architectures For discriminator networks, we use 70×70 PatchGAN [22]. Let `Ck` denote a 4×4 Convolution-InstanceNorm-LeakyReLU layer with k filters and stride 2. After the last layer, we apply a convolution to produce a 1-dimensional output. We do not use InstanceNorm for the first `C64` layer. We use leaky ReLUs with a slope of 0.2. The discriminator architecture is:

```
C64-C128-C256-C512
```