CycleGAN

- Cycle-Consistent Adversarial Networks

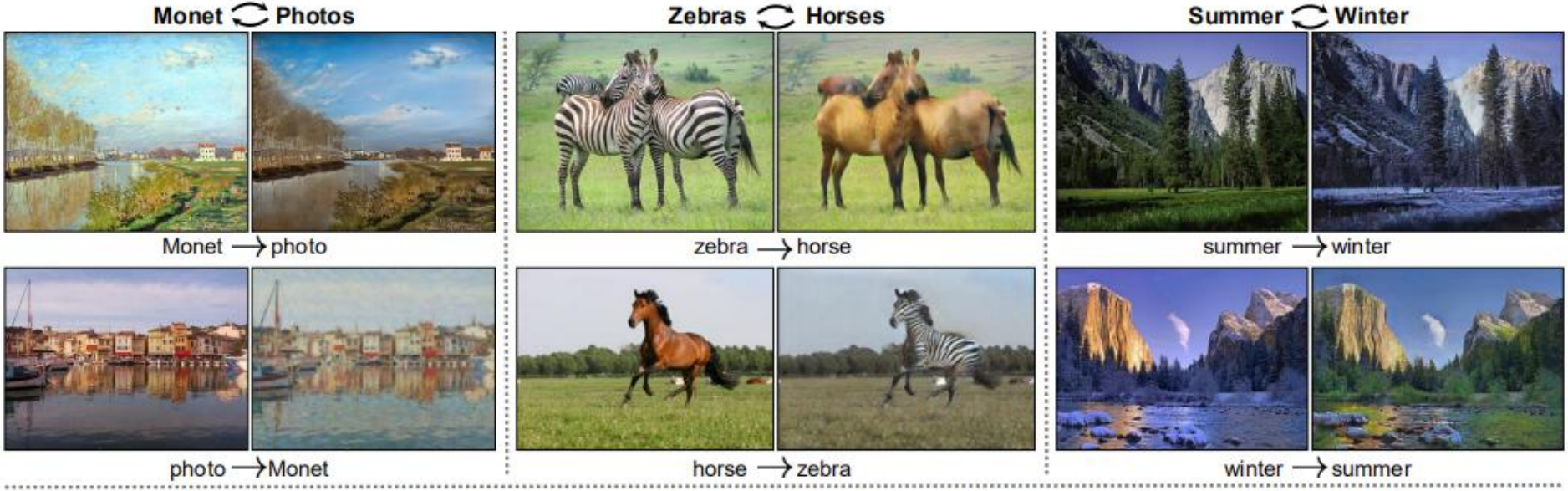
Overview

- CycleGAN
- CycleGAN-VC
- StarGan-VC

CycleGAN

- The goal is to learn a mapping G : X → 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 → X and introduce a cycle consistency loss to enforce F(G(X)) ≈ X (and vice versa).

CycleGAN


Monet ⇄ Photos
Zebras ⇄ Horses
Summer ⇄ Winter
Monet → photo
zebra → horse
summer → winter
photo → Monet
horse → zebra
winter → summer
Photograph
Monet
Van Gogh
Cezanne
Ukiyo-e

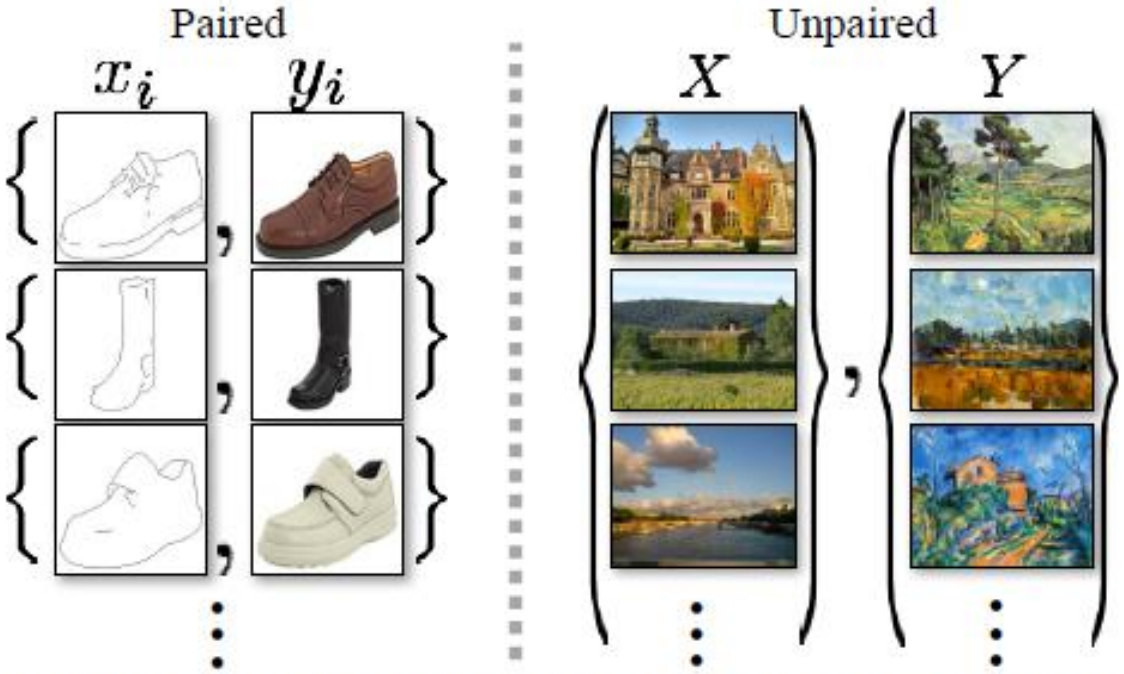


Figure 2: *Paired* training data (left) consists of training examples $\{x_i, y_i\}_{i=1}^N$, where the correspondence between x_i and y_i exists [22]. We instead consider *unpaired* training data (right), consisting of a source set $\{x_i\}_{i=1}^N$ ($x_i \in X$) and a target set $\{y_j\}_{j=1}$ ($y_j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

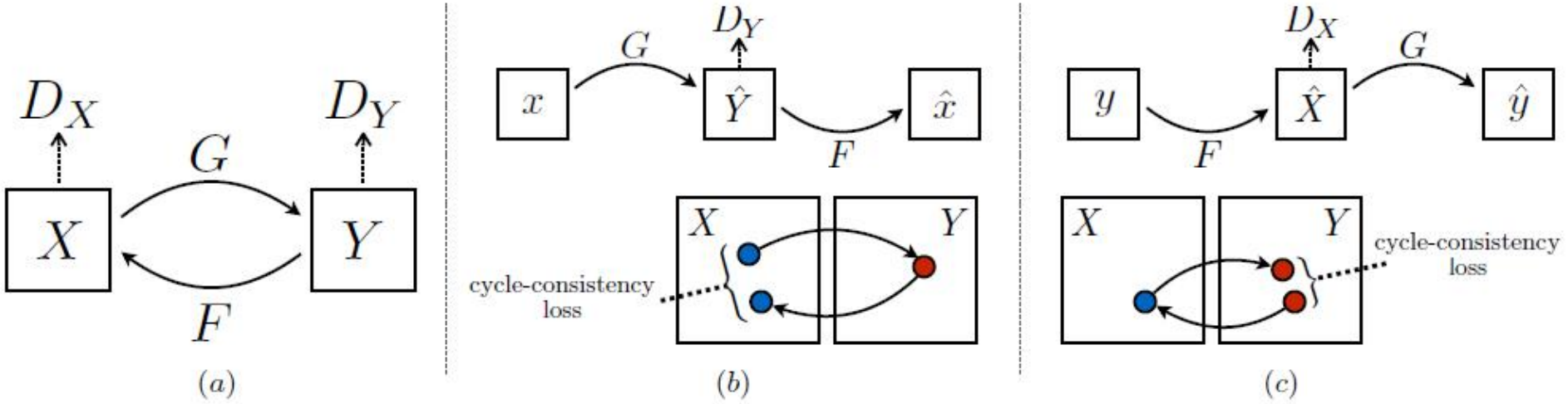
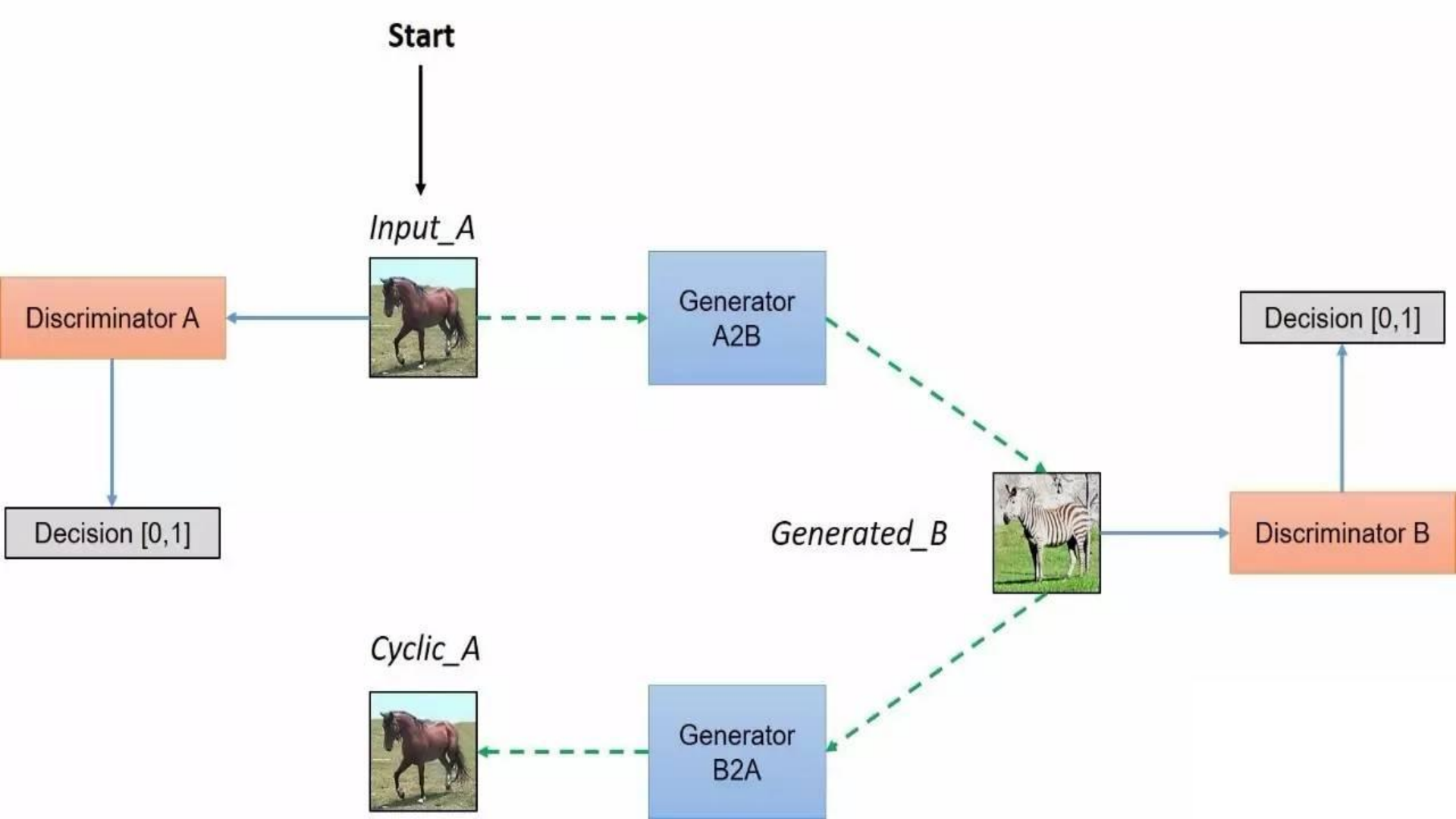
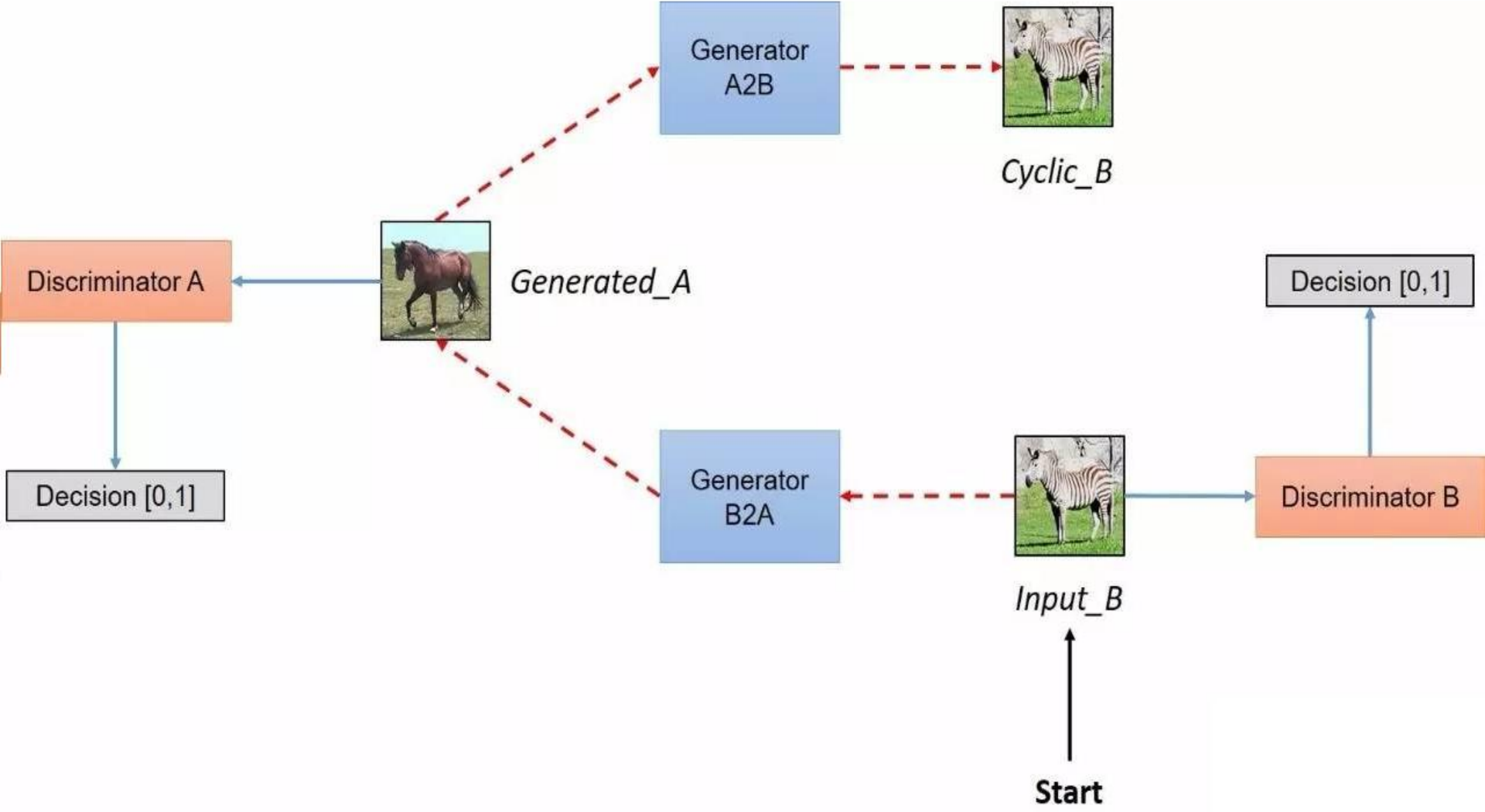


Figure 3: (a) Our model contains two mapping functions $G : X \rightarrow Y$ and $F : Y \rightarrow X$, and associated adversarial discriminators D_Y and D_X. D_Y encourages G to translate X into outputs indistinguishable from domain Y, and vice versa for D_X and F. To further regularize the mappings, 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: $y \rightarrow F(y) \rightarrow G(F(y)) \approx y$


Start
Input_A
Discriminator A
Generator A2B
Decision [0,1]
Decision [0,1]
Generated_B
Discriminator B
Cyclic_A
Generator B2A


Generator
A2B
Cyclic_B
Discriminator A
Generated_A
Decision [0,1]
Decision [0,1]
Generator
B2A
Discriminator B
Input_B
Start

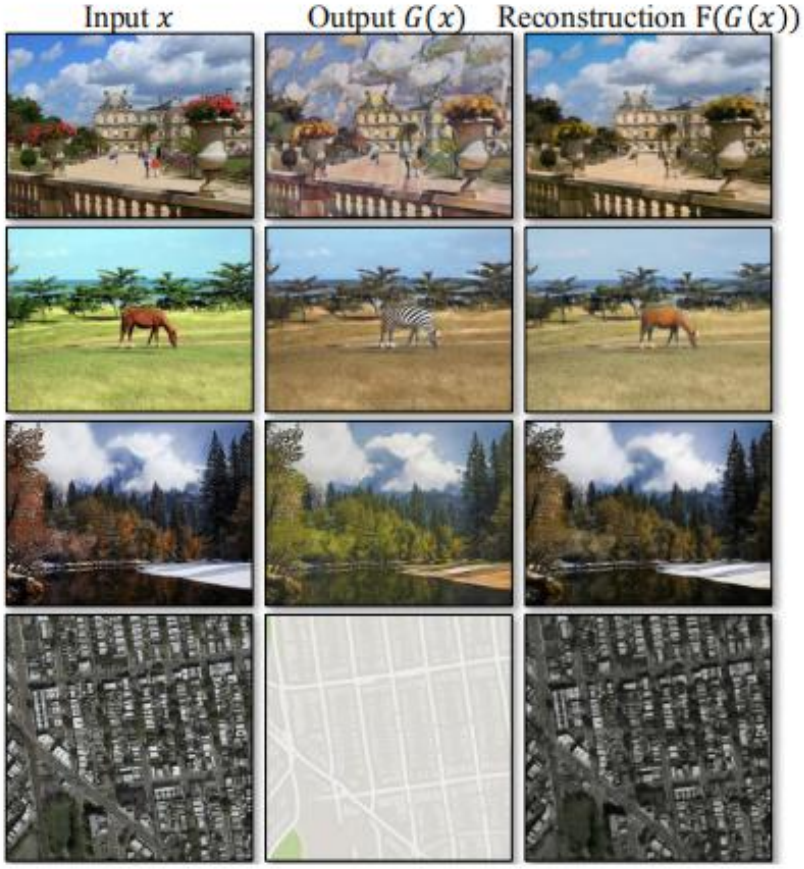


Figure 4: The input images x, output images $G(x)$ and the reconstructed images $F(G(x))$ from various experiments. From top to bottom: photo↔Cezanne, horses↔zebras, winter→summer Yosemite, aerial photos↔Google maps.

losses

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

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

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

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

losses

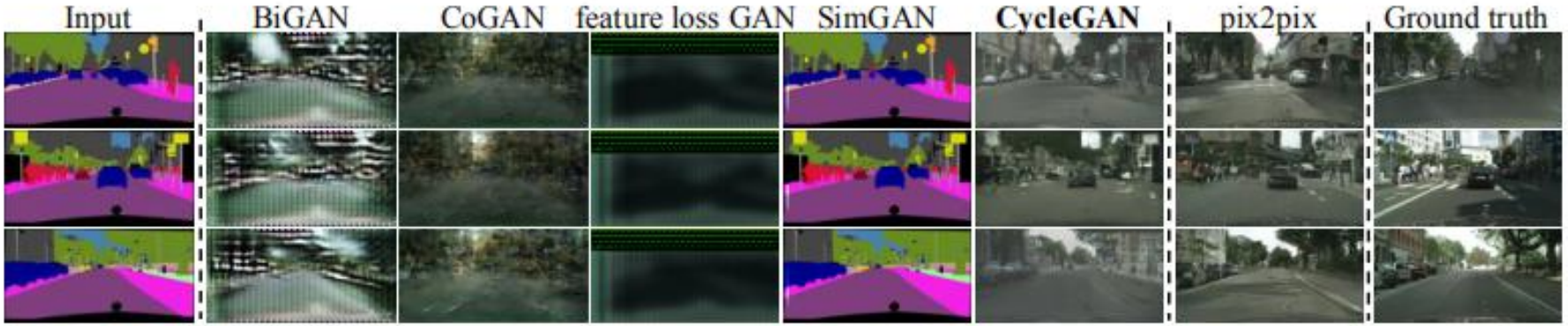


Figure 5: Different methods for mapping labels↔photos trained on Cityscapes images. From left to right: input, BiGAN/ALI [7, 9], CoGAN [32], feature loss + GAN, SimGAN [46], CycleGAN (ours), pix2pix [22] trained on paired data, and ground truth.

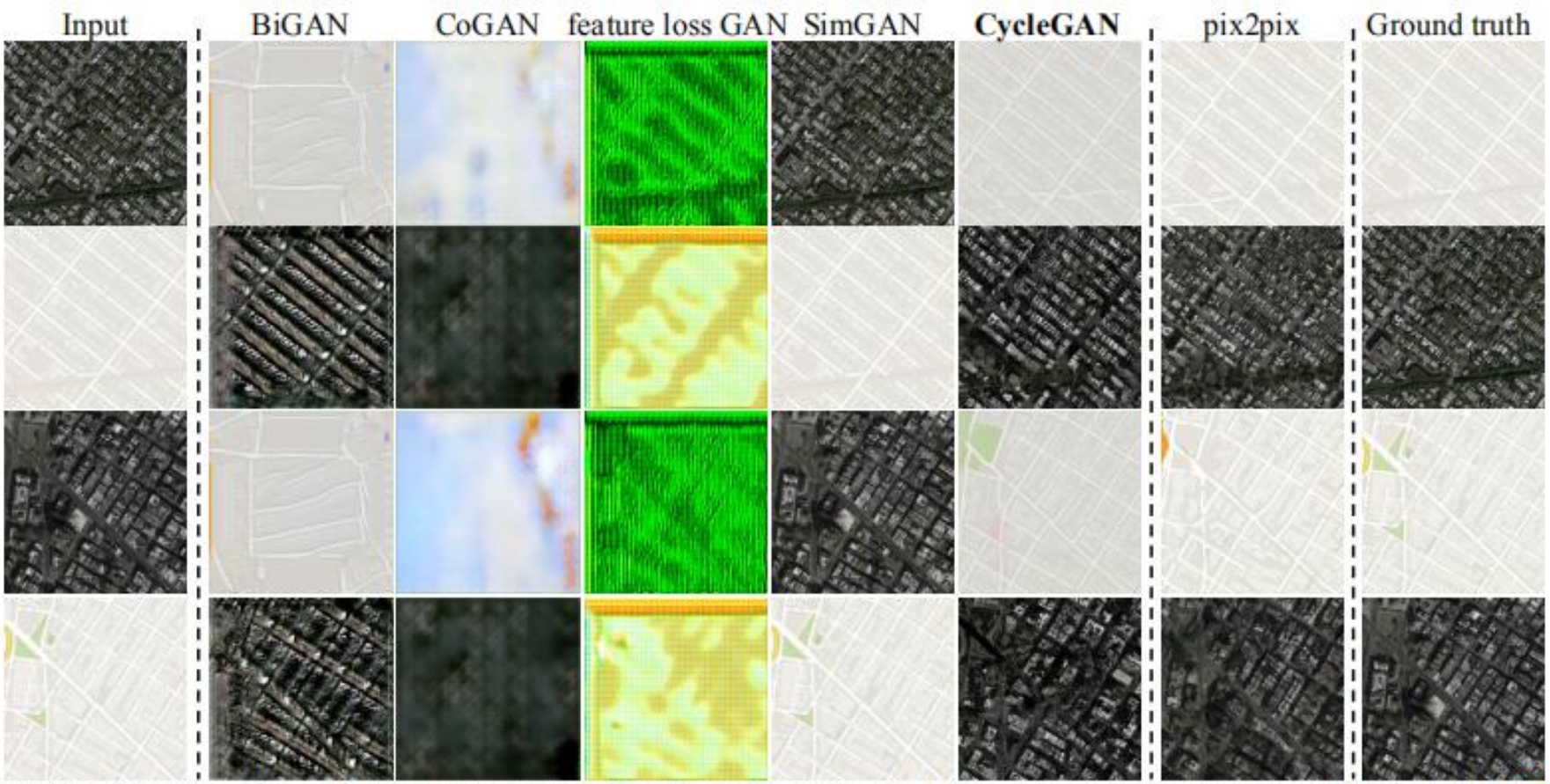


Figure 6: Different methods for mapping aerial photos↔maps on Google Maps. From left to right: input, BiGAN/ALI [7, 9], CoGAN [32], feature loss + GAN, SimGAN [46], CycleGAN (ours), pix2pix [22] trained on paired data, and ground truth.

losses

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.

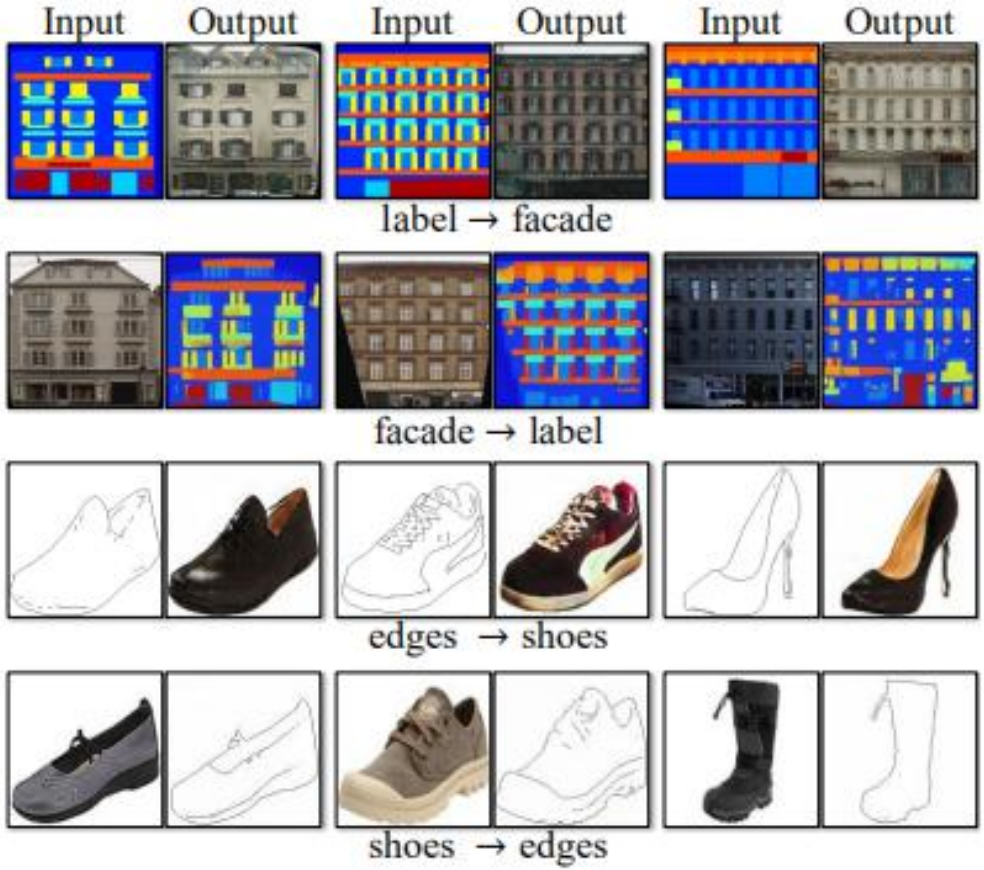


Figure 8: Example results of CycleGAN on paired datasets used in "pix2pix" [22] such as architectural labels↔photos and edges↔shoes.

losses

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., $\mathcal{L}_{\text{identity}}(G, F) = \mathbb{E}_{y \sim p_{\text{data}}(y)}[\|G(y) - y\|_1] + \mathbb{E}_{x \sim p_{\text{data}}(x)}[\|F(x) - x\|_1]$.

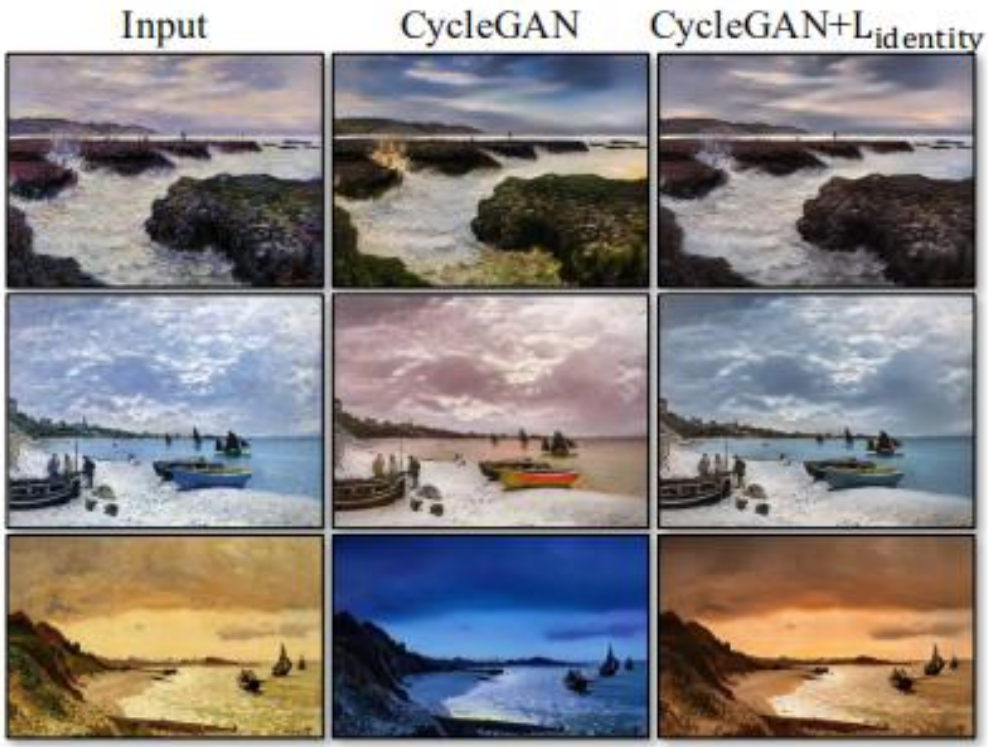
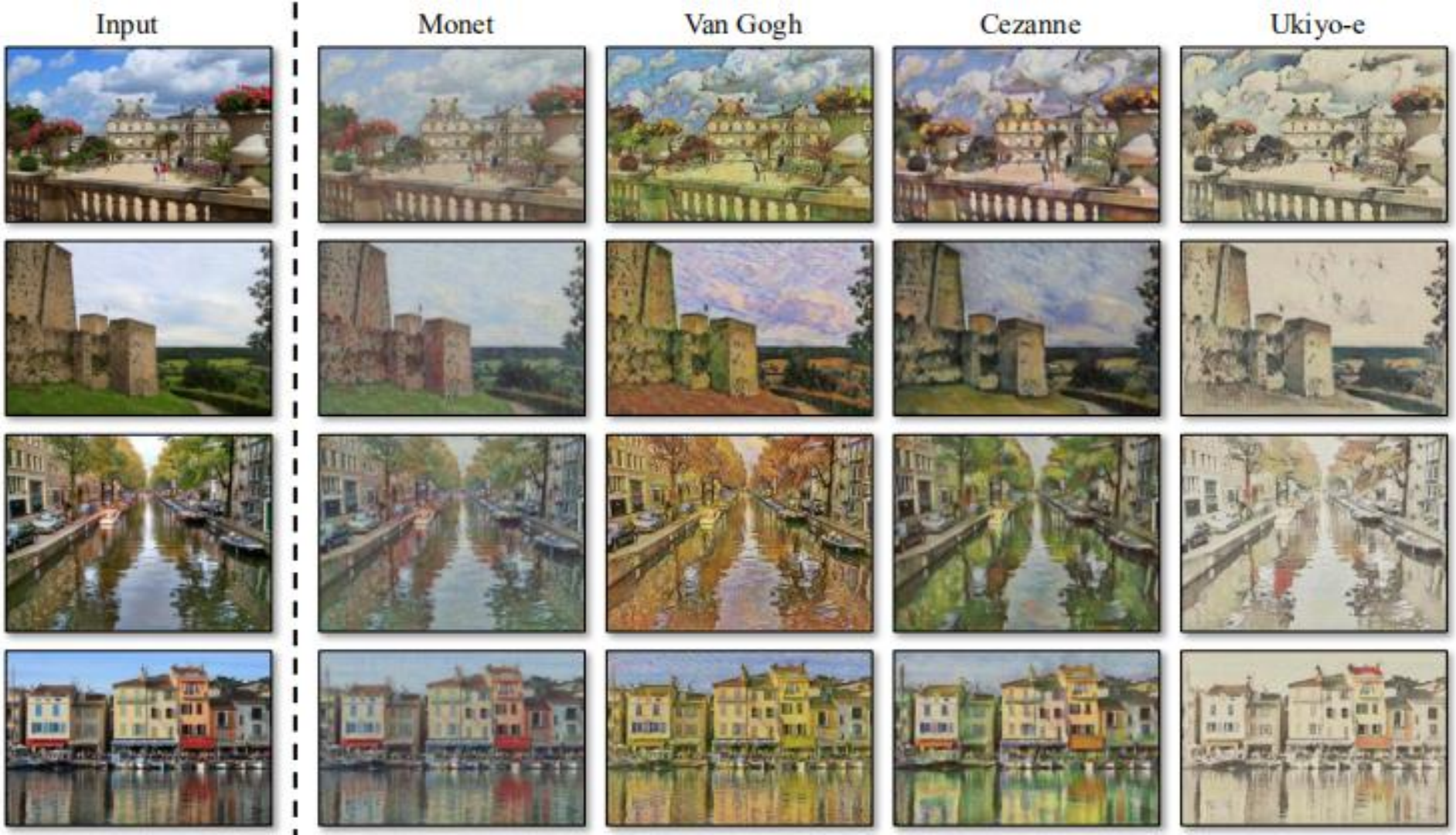
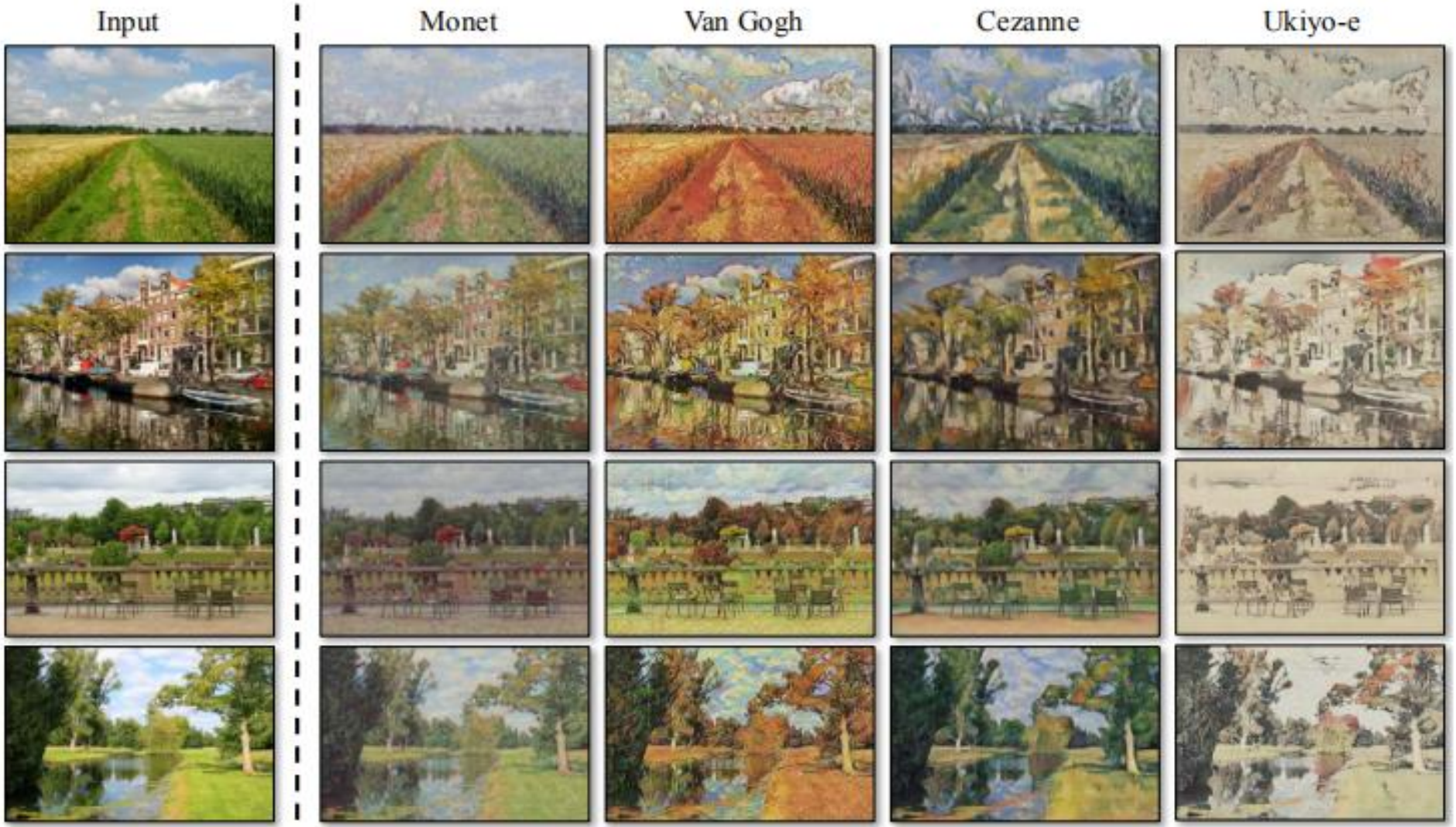


Figure 9: The effect of the *identity mapping loss* on Monet's painting→ photos. From left to right: input paintings, CycleGAN without identity mapping loss, CycleGAN with identity mapping loss. The identity mapping loss helps preserve the color of the input paintings.


Input
Monet
Van Gogh
Cezanne
Ukiyo-e


Input
Monet
Van Gogh
Cezanne
Ukiyo-e


Input
Output
Input
Output
Input
Output
horse → zebra
zebra → horse

winter Yosemite → summer Yosemite

summer Yosemite → winter Yosemite

apple → orange

orange → apple

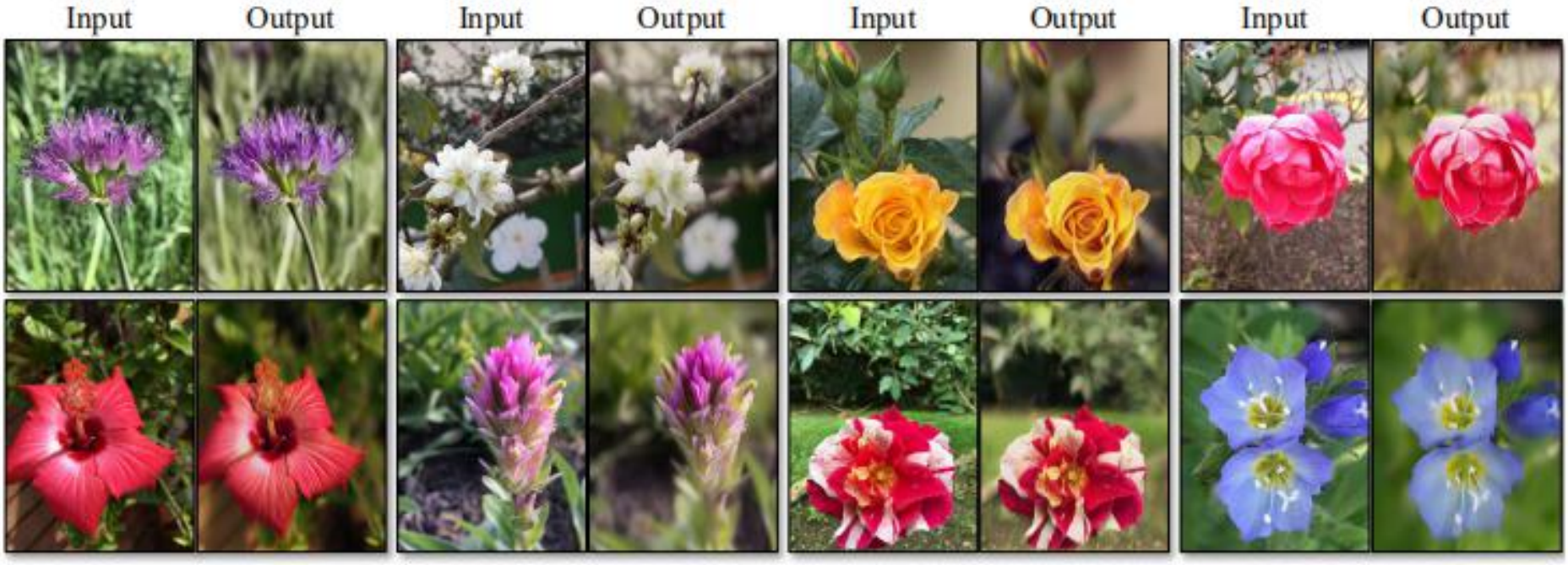


Figure 14: Photo enhancement: mapping from a set of smartphone snaps to professional DSLR photographs, the system often learns to produce shallow focus. Here we show some of the most successful results in our test set – average performance is considerably worse. Please see our website for more comprehensive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

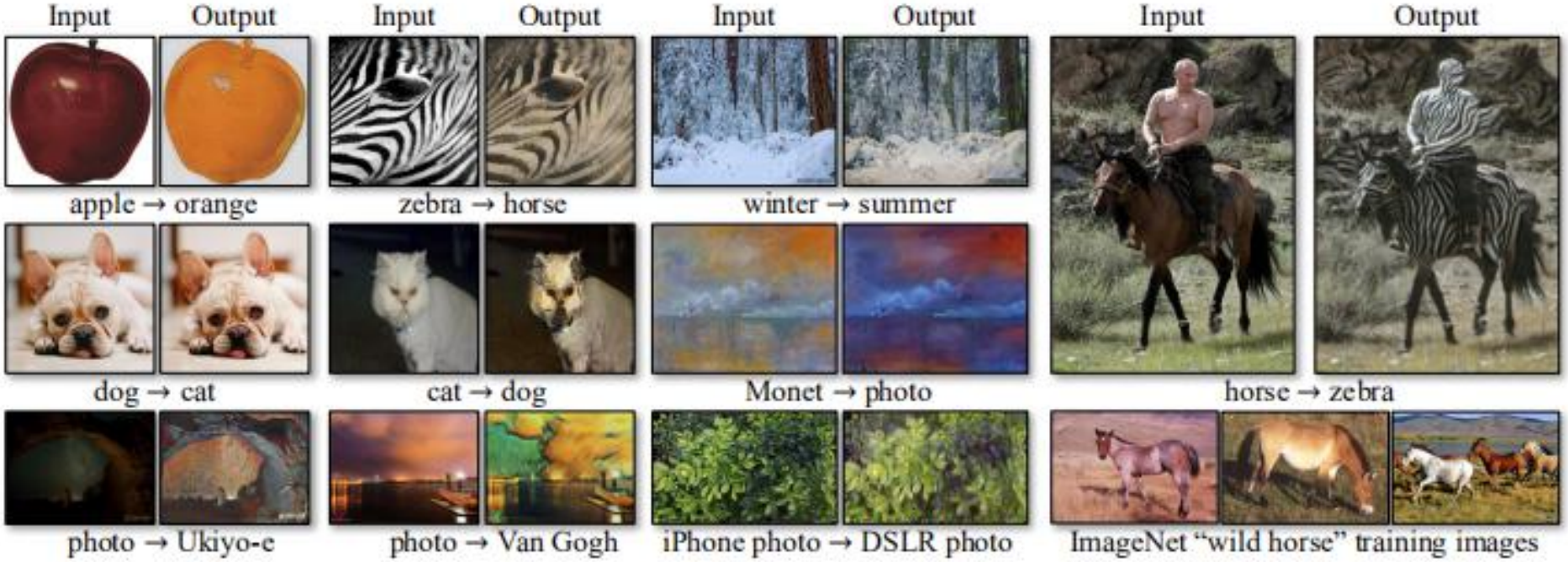


Figure 17: Typical failure cases of our method. Left: in the task of dog→cat transfiguration, CycleGAN can only make minimal changes to the input. Right: CycleGAN also fails in this horse → zebra example as our model has not seen images of horseback riding during training. Please see our website for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

run demo code

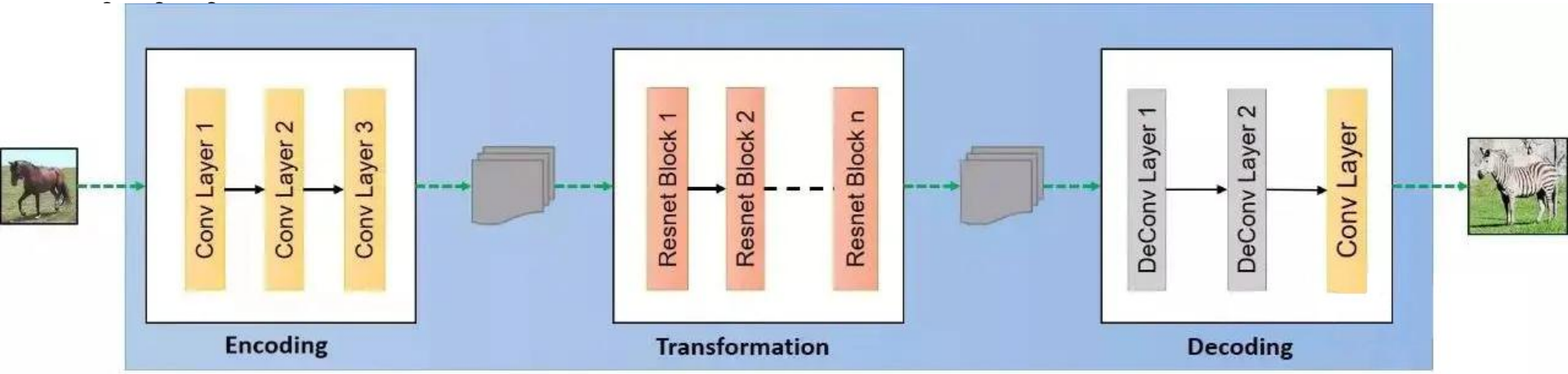
```
python -m visdom.server
```

```
python train.py --dataroot ./datasets/maps
--name maps_cyclegan --model cycle_gan
```

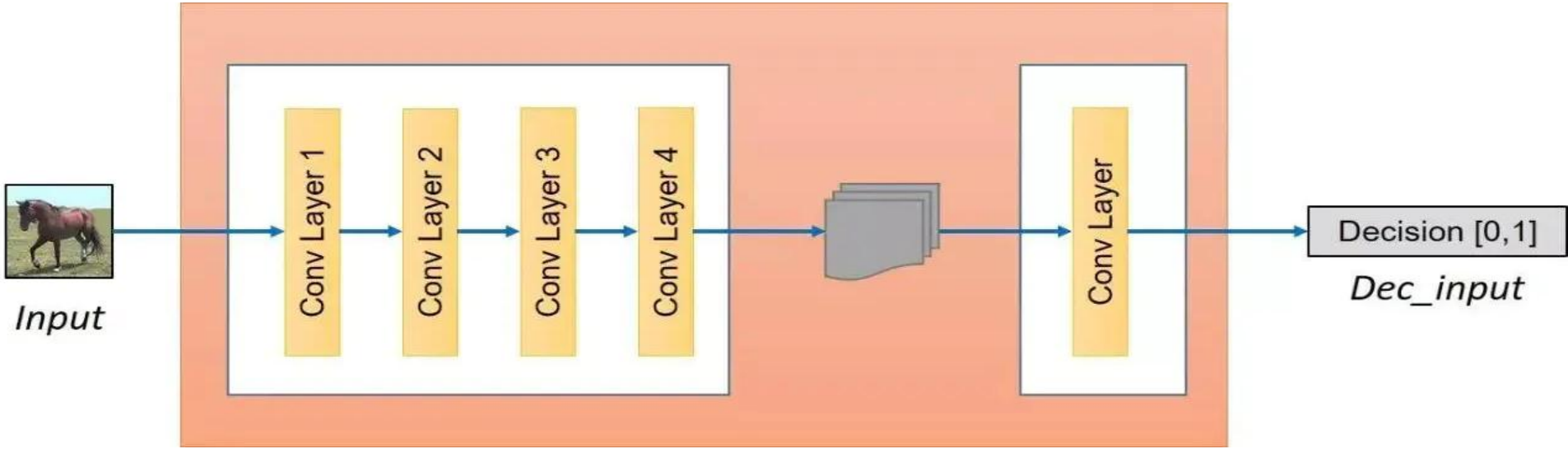
datasets

apple2orange, summer2winter_yosemite, horse2zebra, monet2photo, cezanne2photo, ukiyoe2photo, vangogh2photo, maps, cityscapes, facades, iphone2dslr_flower, ae_photos

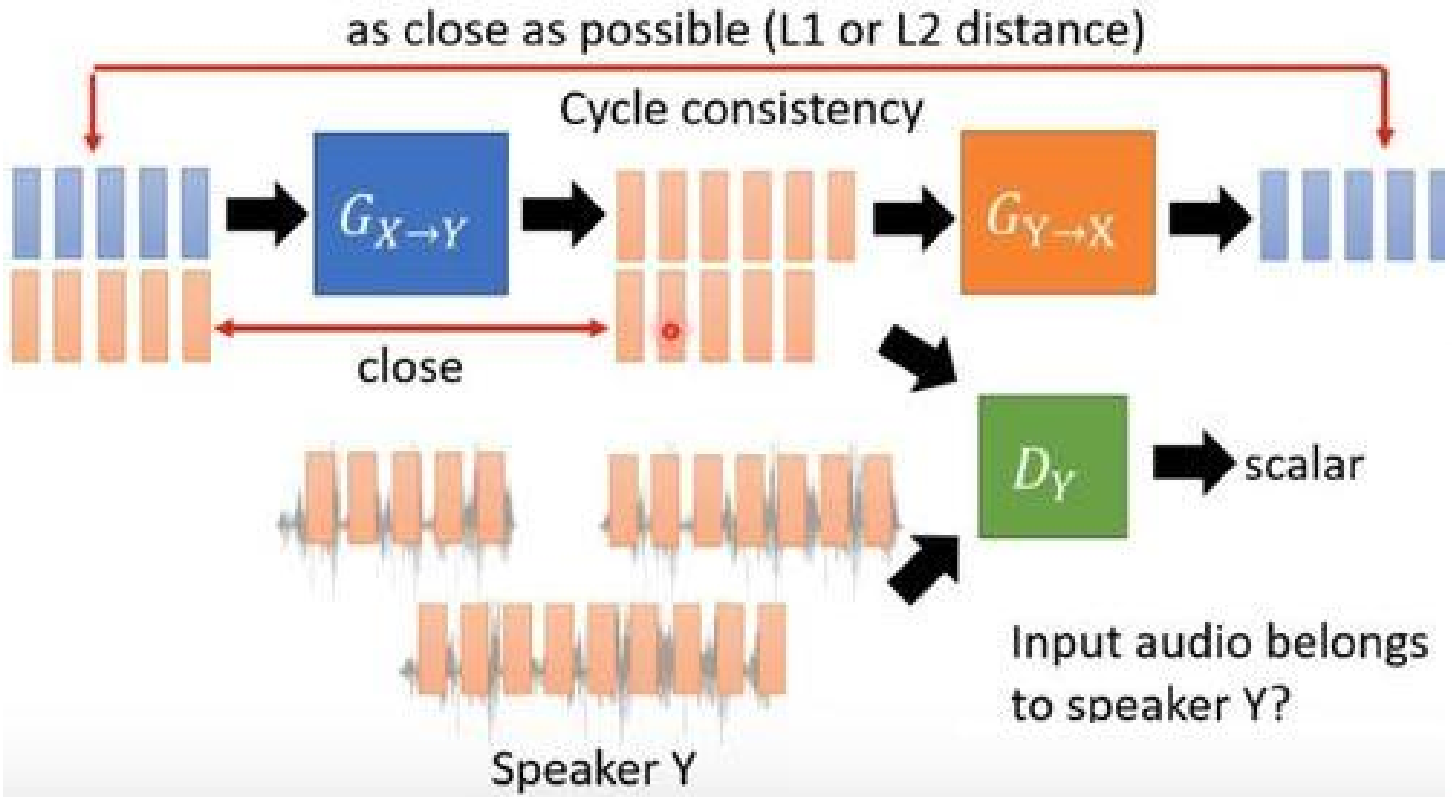
generator


Conv Layer 1
Conv Layer 2
Conv Layer 3
Encoding
Resnet Block 1
Resnet Block 2
Resnet Block n
Transformation
DeConv Layer 1
DeConv Layer 2
Conv Layer
Decoding

discriminator



CycleGAN-VC



Stargan-VC

- requires no parallel utterances, transcriptions, or time alignment procedures for speech generator training,
- simultaneously learns many-to-many mappings across different attribute domains using a single generator network,
- is able to generate converted speech signals quickly enough to allow real-time implementations and
- requires only several minutes of training examples to generate reasonably realistic sounding speech.
- Subjective evaluation experiments on a non-parallel many-to-many speaker identity conversion task revealed that the proposed method obtained higher sound quality and speaker similarity than a state-of-the-art method based on variational autoencoding GANs

CYCLEGAN VOICE CONVERSION

$$
\begin{aligned}
\mathcal{L}_{\text{adv}}^{D_Y}(D_Y) = &- \mathbb{E}_{\mathbf{y}\sim p_Y(\mathbf{y})}[\log D_Y(\mathbf{y})] \\
&- \mathbb{E}_{\mathbf{x}\sim p_X(\mathbf{x})}[\log(1 - D_Y(G(\mathbf{x})))], \\
\mathcal{L}_{\text{adv}}^{G}(G) = &\mathbb{E}_{\mathbf{x}\sim p_X(\mathbf{x})}[\log(1 - D_Y(G(\mathbf{x})))], \\
\mathcal{L}_{\text{adv}}^{D_X}(D_X) = &- \mathbb{E}_{\mathbf{x}\sim p_X(\mathbf{x})}[\log D_X(\mathbf{x})] \\
&- \mathbb{E}_{\mathbf{y}\sim p_Y(\mathbf{y})}[\log(1 - D_X(F(\mathbf{y})))], \\
\mathcal{L}_{\text{adv}}^{F}(F) = &\mathbb{E}_{\mathbf{y}\sim p_Y(\mathbf{y})}[\log(1 - D_X(F(\mathbf{y})))],
\end{aligned}
$$

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

$$
\begin{aligned}
\mathcal{L}_{\text{id}}(G,F) = &\mathbb{E}_{\mathbf{x}\sim p_X(\mathbf{x})}[\|F(\mathbf{x}) - \mathbf{x}\|_1] \\
&+ \mathbb{E}_{\mathbf{y}\sim p_Y(\mathbf{y})}[\|G(\mathbf{y}) - \mathbf{y}\|_1],
\end{aligned}
$$

$$
\begin{aligned}
\mathcal{I}_{G,F}(G,F) = &\mathcal{L}_{\text{adv}}^{G}(G) + \mathcal{L}_{\text{adv}}^{F}(F) \\
&+ \lambda_{\text{cyc}}\mathcal{L}_{\text{cyc}}(G,F) + \lambda_{\text{id}}\mathcal{L}_{\text{id}}(G,F), \\
\mathcal{I}_{D}(D_X,D_Y) = &\mathcal{L}_{\text{adv}}^{D_X}(D_X) + \mathcal{L}_{\text{adv}}^{D_Y}(D_Y),
\end{aligned}
$$

STARGAN VOICE CONVERSION

$$\mathcal{L}_{\text{adv}}^{D}(D) = -\mathbb{E}_{c\sim p(c),\mathbf{y}\sim p(\mathbf{y}|c)}[\log D(\mathbf{y},c)] - \mathbb{E}_{\mathbf{x}\sim p(\mathbf{x}),c\sim p(c)}[\log(1-D(G(\mathbf{x},c),c))],$$

$$\mathcal{L}_{\text{adv}}^{G}(G) = -\mathbb{E}_{\mathbf{x}\sim p(\mathbf{x}),c\sim p(c)}[\log D(G(\mathbf{x},c),c)],$$

$$\mathcal{L}_{\text{cls}}^{C}(C) = -\mathbb{E}_{c\sim p(c),\mathbf{y}\sim p(\mathbf{y}|c)}[\log p_C(c|\mathbf{y})],$$

$$\mathcal{L}_{\text{cls}}^{G}(G) = -\mathbb{E}_{\mathbf{x}\sim p(\mathbf{x}),c\sim p(c)}[\log p_C(c|G(\mathbf{x},c))],$$

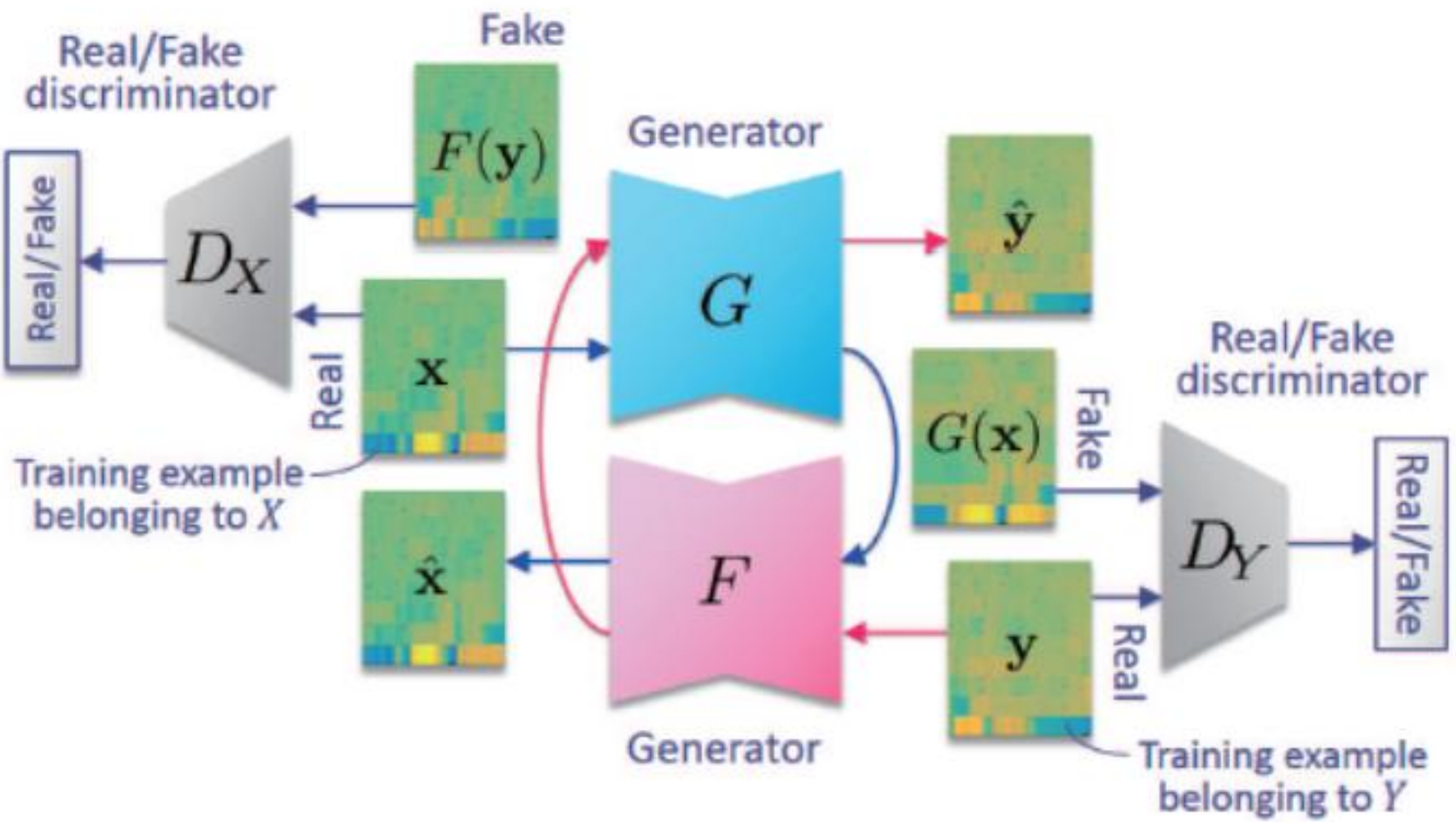
$$\mathcal{L}_{\text{cyc}}(G) = \mathbb{E}_{c'\sim p(c),\mathbf{x}\sim p(\mathbf{x}|c'),c\sim p(c)}[\|G(G(\mathbf{x},c),c')-\mathbf{x}\|_\rho],$$

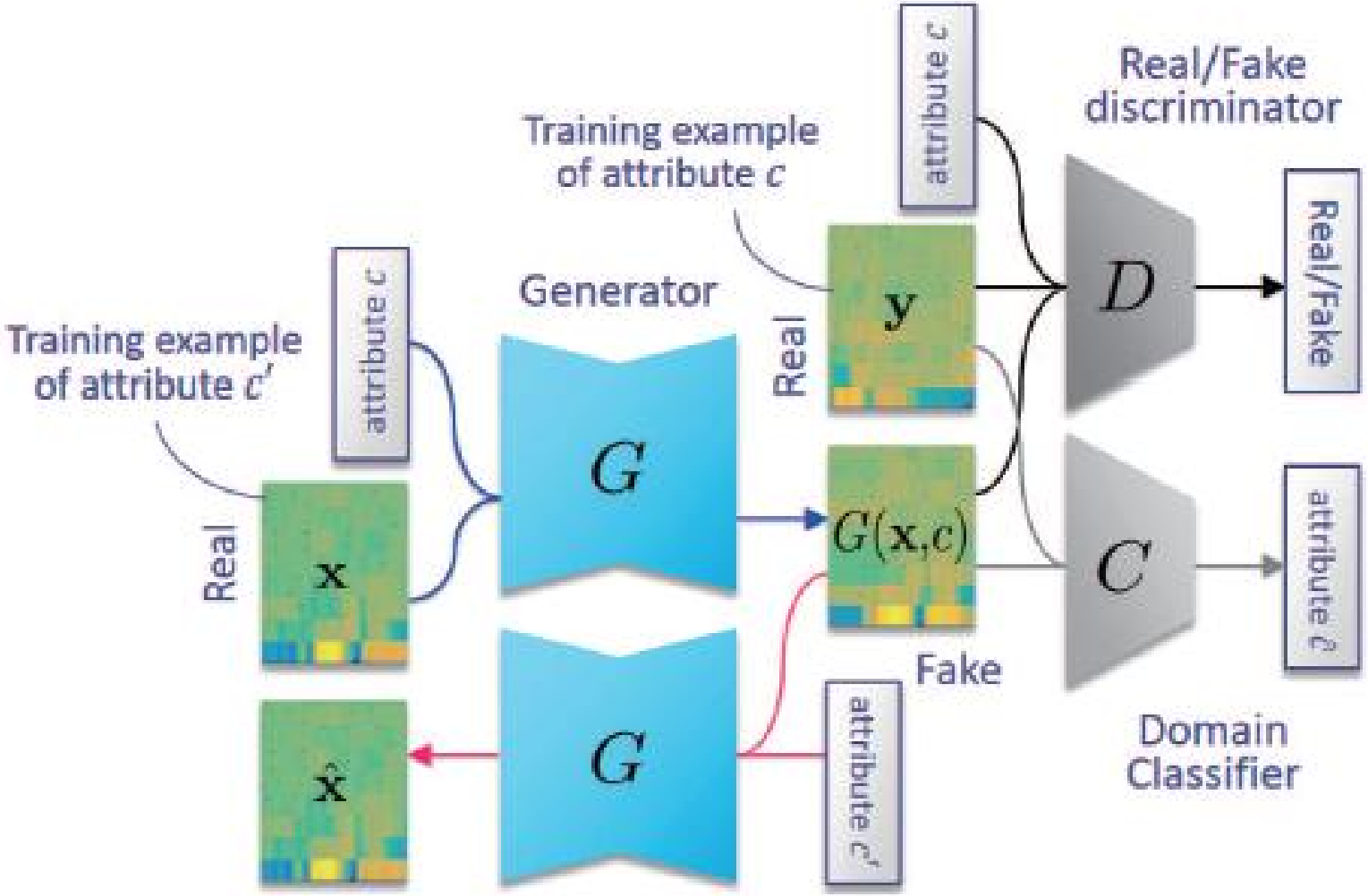
$$\mathcal{L}_{\text{id}}(G) = \mathbb{E}_{c'\sim p(c),\mathbf{x}\sim p(\mathbf{x}|c')}[\|G(\mathbf{x},c')-\mathbf{x}\|_\rho],$$

$$\mathcal{I}_G(G) = \mathcal{L}_{\text{adv}}^{G}(G) + \lambda_{\text{cls}}\mathcal{L}_{\text{cls}}^{G}(G) + \lambda_{\text{cyc}}\mathcal{L}_{\text{cyc}}(G) + \lambda_{\text{id}}\mathcal{L}_{\text{id}}(G),$$

$$\mathcal{I}_D(D) = \mathcal{L}_{\text{adv}}^{D}(D),$$

$$\mathcal{I}_C(C) = \mathcal{L}_{\text{cls}}^{C}(C),$$


Real/Fake discriminator
Fake
$F(\mathbf{y})$
Generator
$\hat{\mathbf{y}}$
Real/Fake
D_X
G
Real
$\mathbf{x}$
Real/Fake discriminator
$G(\mathbf{x})$
Fake
Training example belonging to X
$\hat{\mathbf{x}}$
F
D_Y
Real/Fake
$\mathbf{y}$
Real
Generator
Training example belonging to Y


Training example of attribute c'
attribute c
Generator
Training example of attribute c
attribute c
Real/Fake discriminator
Real
x
G
Real
y
D
Real/Fake
$G(x,c)$
C
attribute c
Fake
Domain Classifier
x'
G
attribute c'