Pro GAN & Cycle GAN

20181896 조유빈

Pro GAN (Progressive Growing of GANs)

- 일반적인 GAN으로 고해상도 이미지를 만든다면 다음과 같은 문제들이 발생

  - High resolution일수록 discriminator는 generator가 생성한 이미지가 Fake이미지인지 아닌지를 구분하기 쉬워진다.

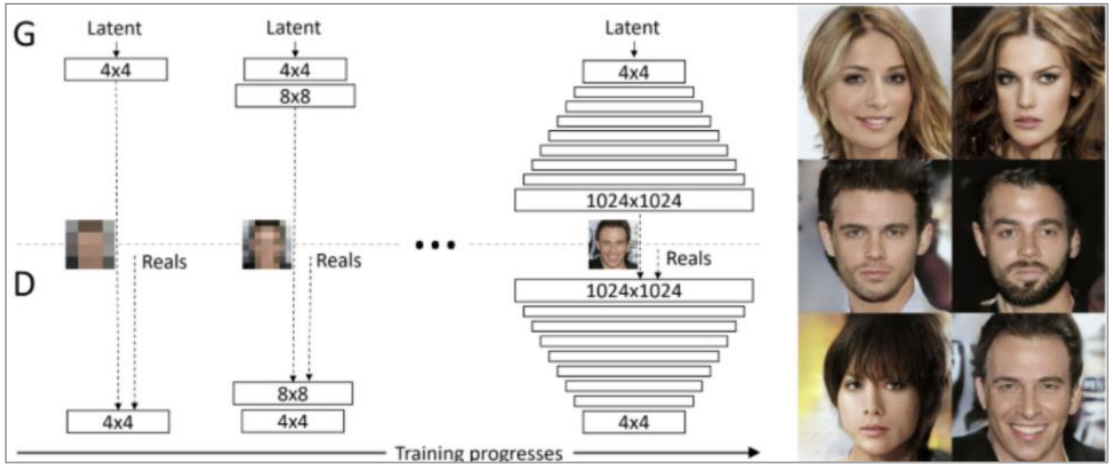
  - 메모리 제약 조건때문에 Batch size를 줄여야 하는데 그렇게 되면 학습이 불안정해지는 현상 발생

Pro GAN (Progressive Growing of GANs) - Primary Contribution

- Start with low-resolution images, and then progressively increase the resolution by adding layers to the networks

- This incremental nature allows the training to first discover large-scale structure of the image distribution and then shift attention to increasingly finer scale detail, instead of having to learn all scales simultaneously

- When new layers are added to the networks, fade them in smoothly



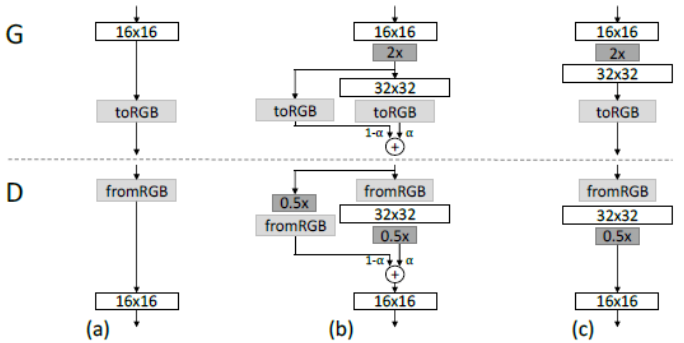
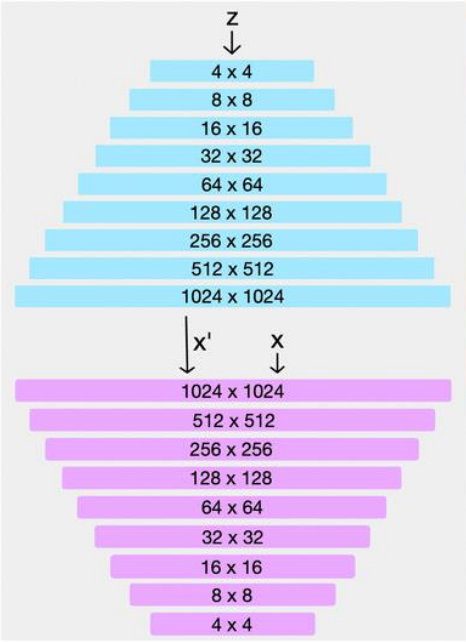


Figure 2: When doubling the resolution of the generator (G) and discriminator (D) we fade in the new layers smoothly. This example illustrates the transition from 16×16 images (a) to 32×32 images (c). During the transition (b) we treat the layers that operate on the higher resolution like a residual block, whose weight α increases linearly from 0 to 1. Here $2\times$ and $0.5\times$ refer to doubling and halving the image resolution using nearest neighbor filtering and average pooling, respectively. The toRGB represents a layer that projects feature vectors to RGB colors and fromRGB does the reverse; both use 1×1 convolutions. When training the discriminator, we feed in real images that are downscaled to match the current resolution of the network. During a resolution transition, we interpolate between two resolutions of the real images, similarly to how the generator output combines two resolutions.

Pro GAN (Progressive Growing of GANs) - Benefit

- Stabilize the training
  - Smaller images have less class information and fewer modes
  - A mapping from latent vector to images by increasing the resolution little by little
- Reduce the training time
  - Most of the iterations are done at lower resolutions



Training time: 14 days
1024x1024 resolution

Generator
Discriminator

z = random code
x = real image
x' = generated image

Pro GAN (Progressive Growing of GANs)

- Increasing variation using mini batch standard deviation
  - GANs have a tendency to capture only a subset of the variation found in training data
  - Compute feature statistics not only from individual images but also across the minibatch,

    thus encouraging the minibatches of generated and training images to show similar statistics
  - Insert it towards the end in the discriminator

Pro GAN (Progressive Growing of GANs) - Normalization

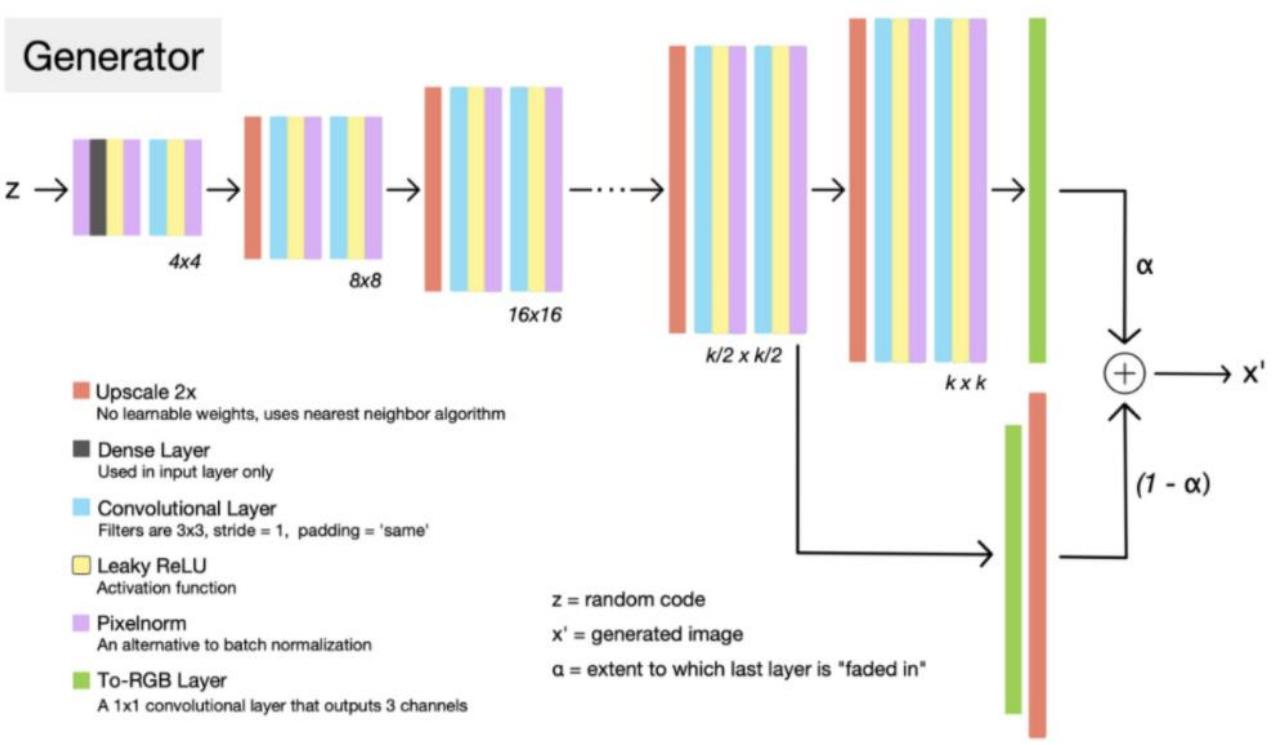
- Equalized learning rate
  - Use a trivial N (0, 1) initialization and then explicitly scale the weights at runtime
  - The dynamic range, and thus the learning speed, is the same for all weights

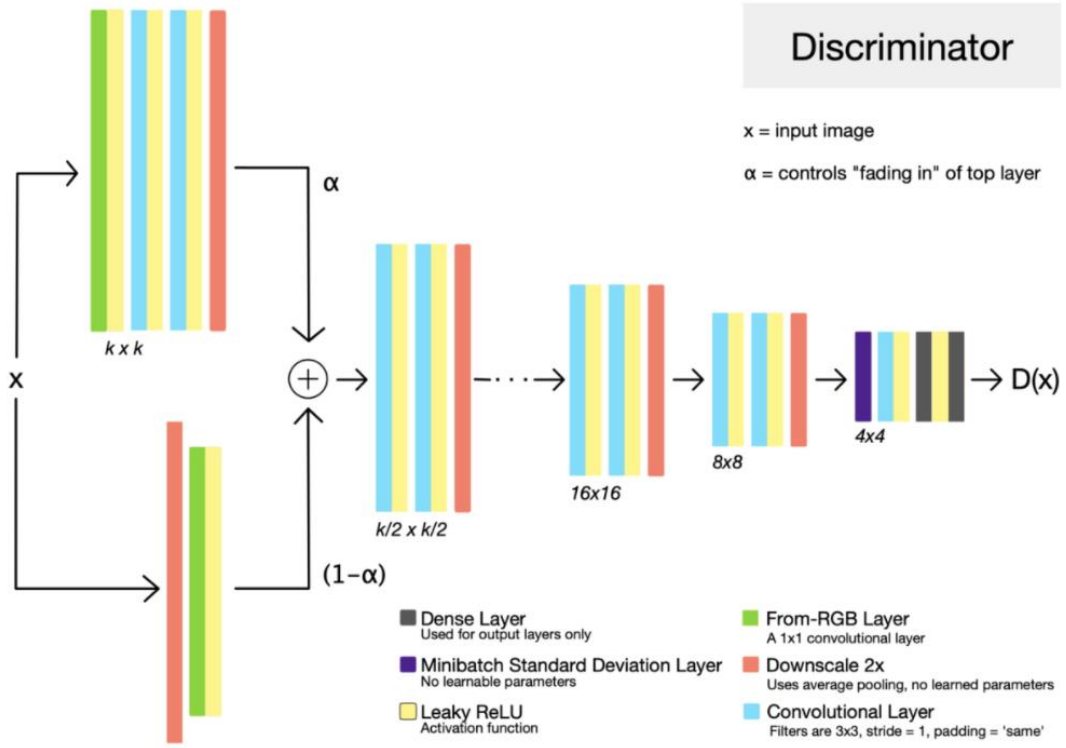
- Pixelwise feature vector normalization in generator
  - Normalize the feature vector in each pixel to unit length in the generator after each convolutional layer

$$b_{x,y} = a_{x,y} / \sqrt{\frac{1}{N} \sum_{j=0}^{N-1} (a_{x,y}^{j})^2 + \epsilon}, \text{ where } \epsilon = 10^{-8}$$

- N - feature map의 크기
- a_x_y : pixel (x, y)에서의 original vector
- b_x_y :pixel (x, y)에서의 normalized vector
- 이 방법은 실험결과에 대한 변화가 없지만, 위에서 언급한 시그널의 크기가 갑자기 커지는 현상를 막아줌.

Pro GAN (Progressive Growing of GANs)





Pro GAN (Progressive Growing of GANs)

- Multi-Scale Statistical Similarity for GAN results
  - Find large-scale mode collapses reliably but fail to react to smaller effects such as loss of variation in colors or textures, and they also do not directly assess image quality in terms of similarity to the training set
  - A successful generator will produce samples whose local image structure is similar to the training set over all scales
  - Considering the multiscale statistical similarity between distributions of local image patches drawn from Laplacian pyramid representations of generated and target images

1) Obtain the patches from the training set and generated set, respectively
2) Normalize them with regard to the mean and standard deviation of each color channel
3) Estimate the statistical similarity by computing their sliced Wasserstein distance (SWD)

- A small Wasserstein distance indicates that the distribution of the patches is similar, meaning that the training images and generator samples appear similar in both appearance and variation at this spatial resolution

Pro GAN (Progressive Growing of GANs)

Mao et al. (2016b) (128 × 128) Gulrajani et al. (2017) (128 × 128) Our (256 × 256)

Cycle GAN

- Obtaining paired training data can be difficult and expensive.
- 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.
- Seek an algorithm that can learn to translate between domains without paired input-output examples

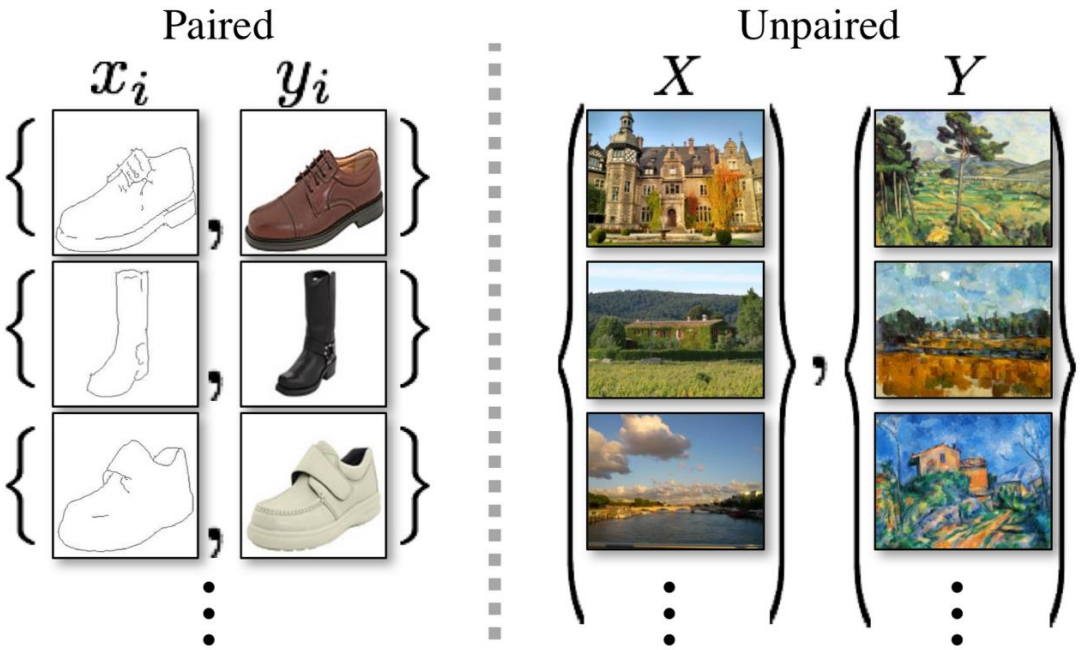


Figure 2: *Paired* training data (left) consists of training examples $\{x_i, y_i\}_{i=1}^N$, where the correspondence between x_i and y_i exists [22]. We instead consider *unpaired* training data (right), consisting of a source set $\{x_i\}_{i=1}^N$ ($x_i \in X$) and a target set $\{y_j\}_{j=1}$ ($y_j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

Cycle GAN

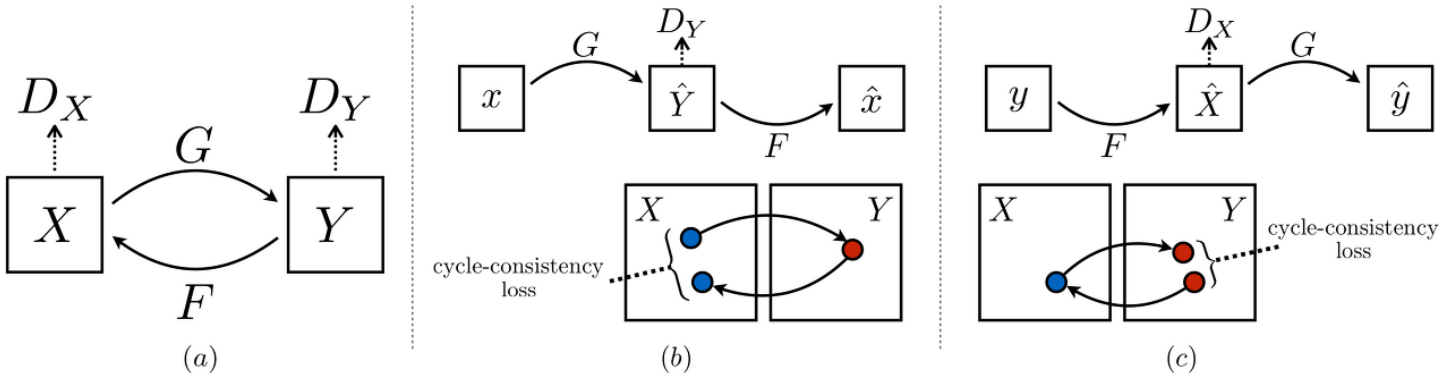


Figure 3: (a) Our model contains two mapping functions $G : X \rightarrow Y$ and $F : Y \rightarrow X$, and associated adversarial discriminators D_Y and D_X. D_Y encourages G to translate X into outputs indistinguishable from domain Y, and vice versa for D_X and F. To further regularize the mappings,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

- Translators G : X → Y and F : Y → X should be inverses of each other, and both mappings should be bijections.
- Dx aims to distinguish between images {x} and translated images {F(y)}; in the same way, Dy aims to discriminate between {y} and {G(x)}
- Add a cycle consistency loss that encourages F(G(x)) ≈ x and G(F(y)) ≈ y. Combining this loss with adversarial losses on domains X and Y yields our full objective for unpaired image-to-image translation.

Cycle GAN

- **Adversarial Loss**

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

- G tries to generate images G(x) that look similar to images from domain Y , while Dy aims to distinguish between translated samples G(x) and real samples y.

$$\min_G \max_{D_Y} \mathcal{L}_{GAN}(G, D_Y, X, Y)$$

- G aims to minimize this objective against an adversary D that tries to maximize it.

$$\min_F \max_{D_X} \mathcal{L}_{GAN}(F, D_X, Y, X)$$

- A similar adversarial loss for the mapping function F : Y → X and its discriminator Dx as well.

Cycle GAN

- **Cycle Consistency Loss**
  - 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.
  - Forward cycle consistency : $x \rightarrow G(x) \rightarrow F(G(x)) \approx x$
  - Backward cycle consistency : $y \rightarrow F(y) \rightarrow G(F(y)) \approx y$

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

Cycle GAN

- **Full Objective Function**

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

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

  - λ controls the relative importance of the two objectives

Cycle GAN

- **Image reconstruction quality**

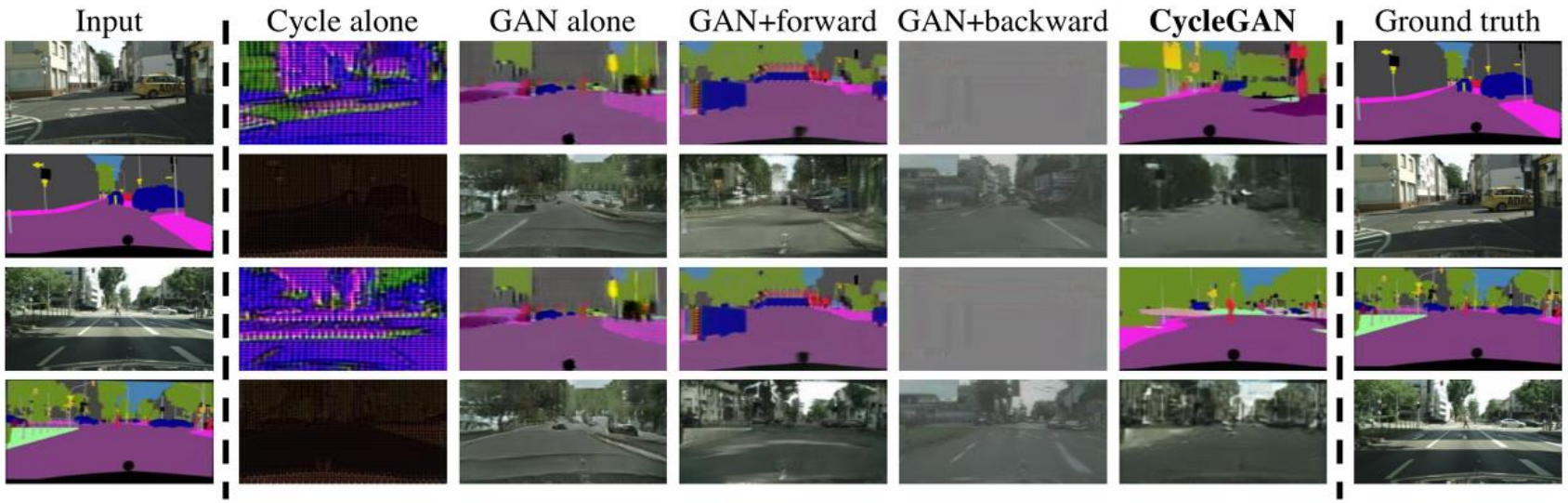
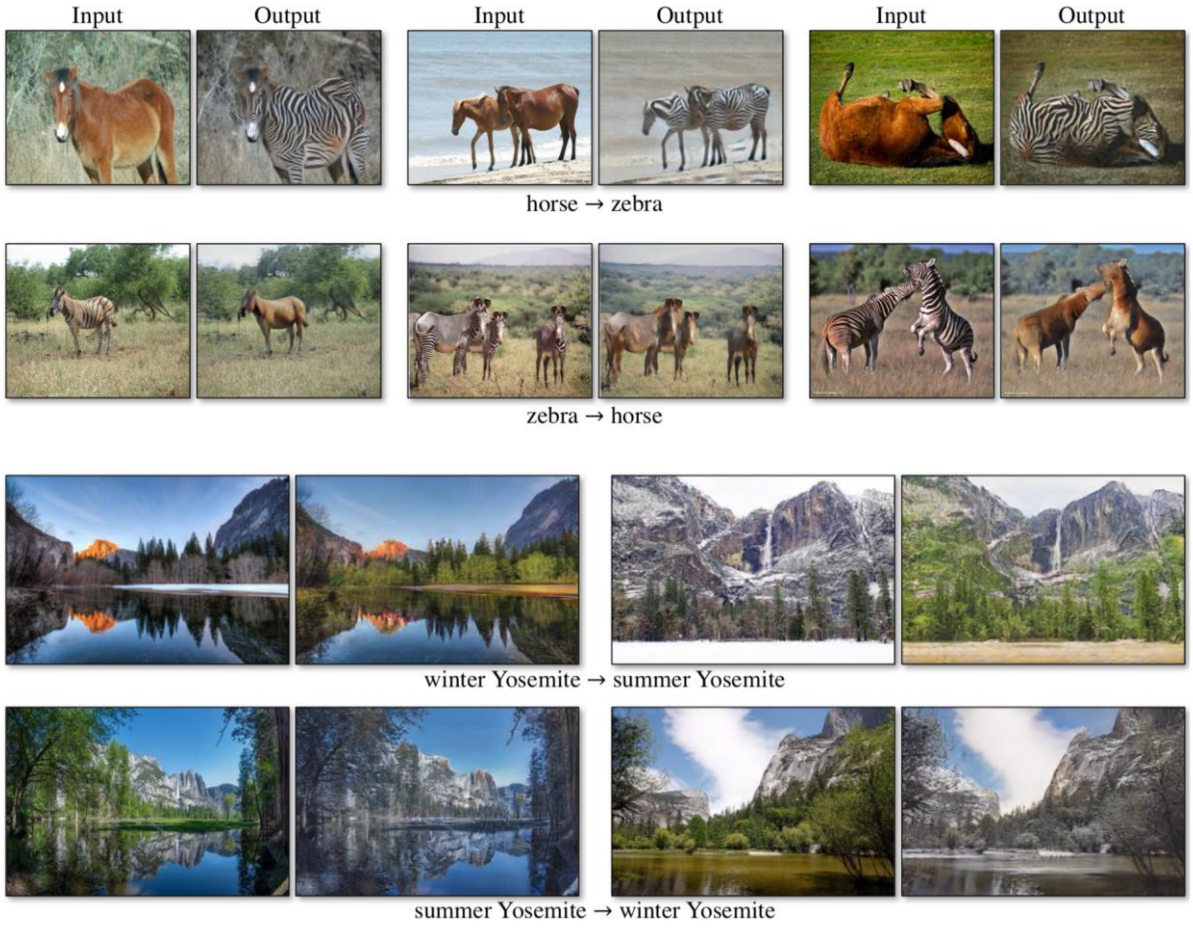
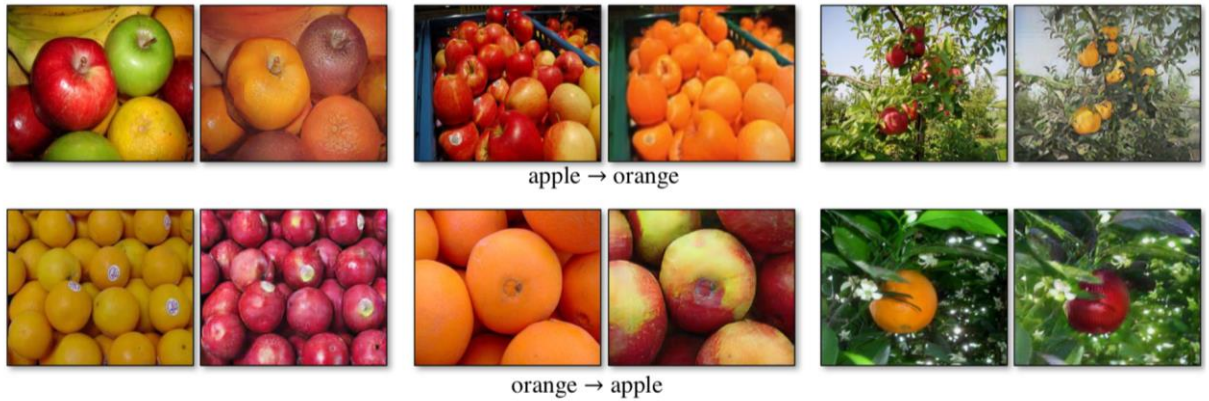


Figure 7: Different variants of our method for mapping labels↔photos trained on cityscapes. From left to right: input, cycle-consistency loss alone, adversarial loss alone, GAN + forward cycle-consistency loss ($F(G(x)) \approx x$), GAN + backward cycle-consistency loss ($G(F(y)) \approx y$), CycleGAN (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.

Cycle GAN

- **Object Transfiguration / Season Transfer**





Cycle GAN

- **Photo generation from paintings**

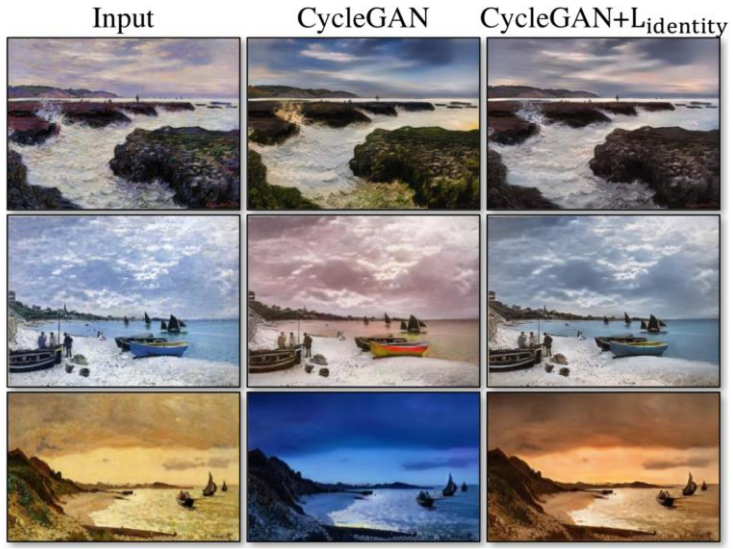


Figure 9: The effect of the *identity mapping loss* on Monet's painting→ photos. From left to right: input paintings, CycleGAN without identity mapping loss, CycleGAN with identity mapping loss. The identity mapping loss helps preserve the color of the input paintings.

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

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

Cycle GAN

- **Photo enhancement**

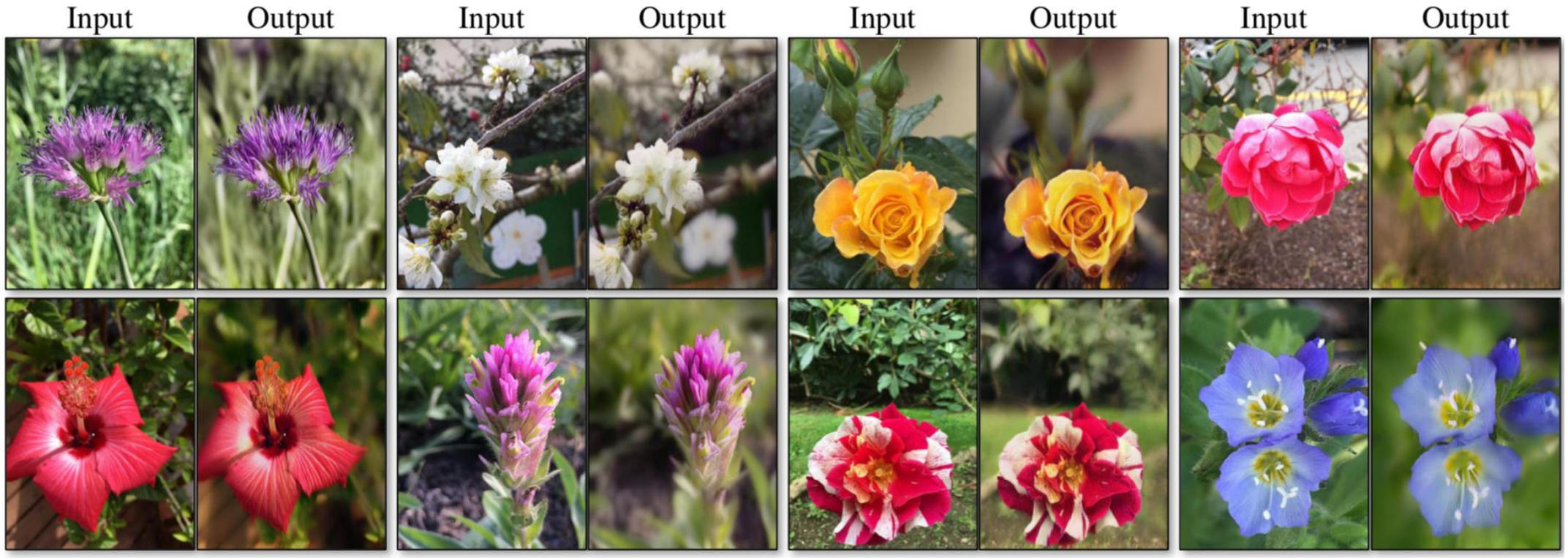


Figure 14: Photo enhancement: mapping from a set of smartphone snaps to professional DSLR photographs, the system often learns to produce shallow focus. Here we show some of the most successful results in our test set – average performance is considerably worse. Please see our website for more comprehensive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