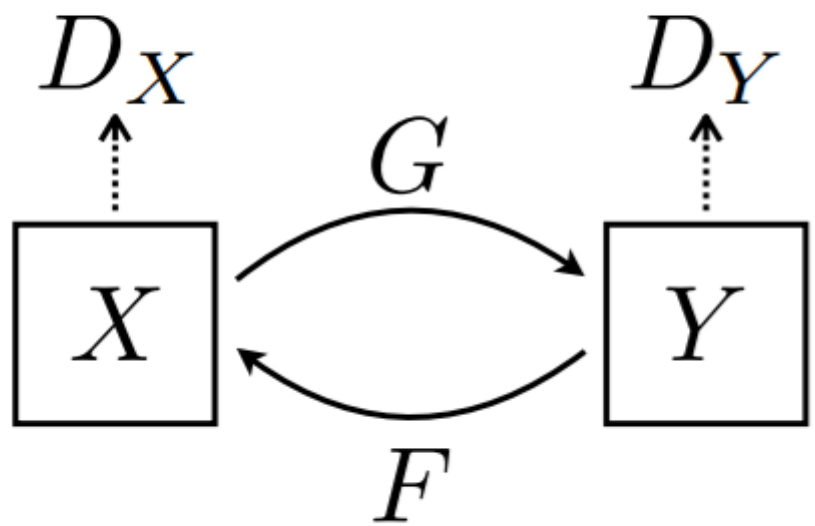
Cycle GAN



(a)

Adversarial Loss G: X -> Y

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

G x to y loss

D_y loss

Ls GAN

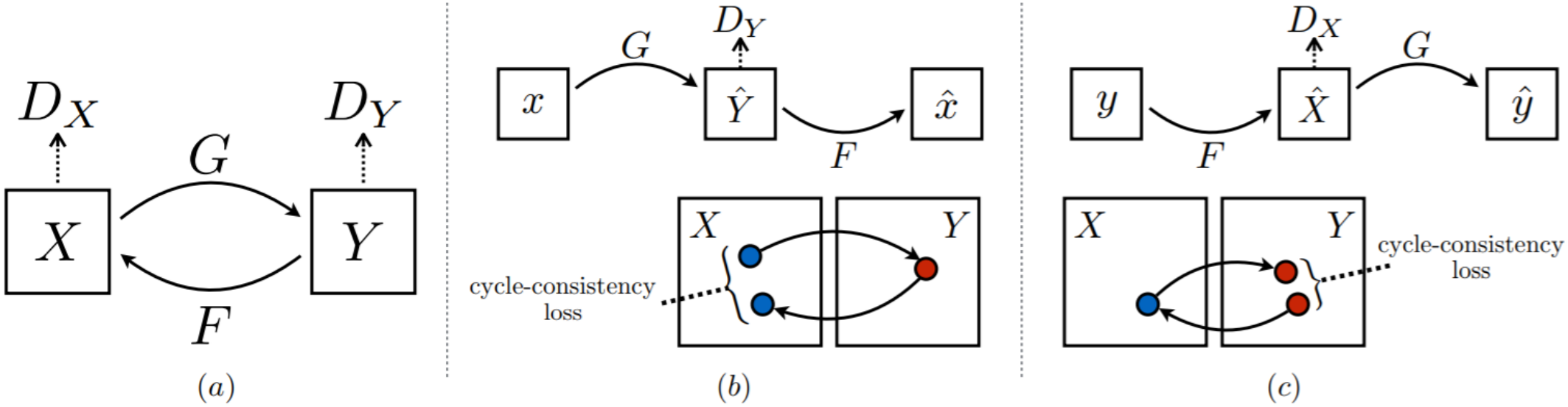
$$Loss_{x \to y} = \mathbb{E}_y[(D_y(y) - 1)^2] + \mathbb{E}_x[(D_y(G(x)))^2]$$

D minimize

$$\mathbb{E}_{x \sim p_{\text{data}}(x)}[(D(G(x)) - 1)^2]$$

G minimize

Cycle-consistency loss



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

L1 norm

Full Objective

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

code

```
# Generator arguments : input_dim, num_filter, output_dim, num_resnet
G_A = Generator(3, params.ngf, 3, params.num_resnet)
G_B = Generator(3, params.ngf, 3, params.num_resnet)
# Discriminator arguments : input_dim, num_filter, output_dim
D_A = Discriminator(3, params.ndf, 1)
D_B = Discriminator(3, params.ndf, 1)
```

G,D 2개씩 사용

https://github.com/ratsgo/ratsgo.github.io/blob/master/_posts/2018-01-30-genmodels.md

G_loss

```
# A -> B
fake_B = G_A(real_A)
D_B_fake_decision = D_B(fake_B)
G_A_loss = MSE_loss(D_B_fake_decision, Variable(torch.ones(D_B_fake_decision.size())))

# forward cycle loss
recon_A = G_B(fake_B)
cycle_A_loss = L1_loss(recon_A, real_A) * params.lambdaA

# B -> A
fake_A = G_B(real_B)
D_A_fake_decision = D_A(fake_A)
G_B_loss = MSE_loss(D_A_fake_decision, Variable(torch.ones(D_A_fake_decision.size())))

# backward cycle loss
recon_B = G_A(fake_A)
cycle_B_loss = L1_loss(recon_B, real_B) * params.lambdaB

# Back propagation
G_loss = G_A_loss + G_B_loss + cycle_A_loss + cycle_B_loss
```

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

D_loss

```
# Train discriminator D_A
D_A_real_decision = D_A(real_A)
D_A_real_loss = MSE_loss(D_A_real_decision, Variable(torch.ones(D_A_real_decision.size())))
fake_A = fake_A_pool.query(fake_A)
D_A_fake_decision = D_A(fake_A)
D_A_fake_loss = MSE_loss(D_A_fake_decision, Variable(torch.zeros(D_A_fake_decision.size())))
D_A_loss = (D_A_real_loss + D_A_fake_loss) * 0.5

# Train discriminator D_B
D_B_real_decision = D_B(real_B)
D_B_real_loss = MSE_loss(D_B_real_decision, Variable(torch.ones(D_B_real_decision.size())))
fake_B = fake_B_pool.query(fake_B)
D_B_fake_decision = D_B(fake_B)
D_B_fake_loss = MSE_loss(D_B_fake_decision, Variable(torch.zeros(D_B_fake_decision.size())))
D_B_loss = (D_B_real_loss + D_B_fake_loss) * 0.5
```