SK hynix i-TAP
반도체 Data Scientist를 위한
ML/DL 심화 커리큘럼

5강 GANs

Ernest K. Ryu (류경석)

2020.12.9

서울대학교 · 자연과학대학

5강. GANs

- Minimax optimization

Introduction to Minimax Optimization

In supervised learning, we solve the minimization problem

$$\underset{\theta}{\text{minimize}} \quad \mathbb{E}_X[L(\theta; X)]$$

with

$$\theta^{k+1} = \theta^k - \alpha_k \nabla L\left(\theta^k; X^k\right)$$

where X^k is a randomly selected or randomly generated data.

(We need expectations instead of finite sums to accommodate expectations with respect to generated data.)

Introduction to Minimax Optimization

In adversarial training, we solve

$$\underset{\theta_1}{\text{minimize}}\ \underset{\theta_2}{\text{maximize}} \quad \mathbb{E}_X[L(\theta_1, \theta_2; X)]$$

Simultaneous gradient "descent"

$$\theta_2^{k+1} = \theta_2^k + \alpha_k \nabla L\left(\theta_1^k, \theta_2^k; X^k\right)$$
$$\theta_1^{k+1} = \theta_1^k - \alpha_k \nabla L\left(\theta_1^k, \theta_2^k; X^k\right)$$

(Also called simultaneous gradient "descent-ascent".)

SCIENCE 서울대학교·자연과학대학

Introduction to Minimax Optimization

Alternating gradient descent

$$\theta_2^{k+1} = \theta_2^k + \alpha_k \nabla L\left(\theta_1^k, \theta_2^k; X^k\right)$$
$$\theta_1^{k+1} = \theta_1^k - \alpha_k \nabla L\left(\theta_1^k, \theta_2^{k+1}; X^k\right)$$

Slightly better than simultaneous update empirically.

Introduction to Minimax Optimization

Alternating multi ascent-single descent

$$\theta_2^{k+1,(1)} = \theta_2^{k,(N_{\text{dis}})} + \alpha_k \nabla L\left(\theta_1^k, \theta_2^{k,(N_{\text{dis}})}; X^k\right)$$

$$\theta_2^{k+1,(2)} = \theta_2^{k+1,(1)} + \alpha_k \nabla L\left(\theta_1^k, \theta_2^{k,(1)}; X^k\right)$$

$$\vdots$$

$$\theta_2^{k+1,(N_{\text{dis}})} = \theta_2^{k+1,(N_{\text{dis}}-1)} + \alpha_k \nabla L\left(\theta_1^k, \theta_2^{k,(N_{\text{dis}}-1)}; X^k\right)$$

$$\theta_1^{k+1} = \theta_1^k - \alpha_k \nabla L\left(\theta_1^k, \theta_2^{k+1,N_{\text{dis}}}; X^k\right)$$

Common value: $N_{\text{dis}} = 5$

Introduction to Minimax Optimization

Minimax optimization is much more difficult than minimization.

- No good way to quantify and ensure progress. Proxy measures such as inception score or Fréchet Inception Distance are used.
- Tuning stepsize and optimization parameters is much more tricky.

Generative Adversarial Networks

Goal: Given data $X_1, \dots, X_N \sim p^{\mathrm{true}}$ learn "generative distribution" $p^{\mathrm{gen}} \approx p^{\mathrm{true}}$.

A "generative distribution" is a distribution from which you can efficiently generate data from. (Being able to evaluate the density function is not necessary.)

Implicitly represent p^{gen} with $G_{\theta_G}(Z)$, where Z is a standard Gaussian vector.

Goodfellow, Pouget-Abadie, Mirza, Xu, Warde-Farley, Ozair, Courville, Bengio, Generative Adversarial Networks, NeurIPS, 2014.

Generative Adversarial Networks

Naïve idea: solve

$$\underset{\theta_G}{\text{minimize}} \quad d\left(p_{\theta_G}^{\text{gen}}, p^{\text{true}}\right)$$

where $d(\cdot,\cdot)$ is a distance measure for probability distributions.

Problem: How to we compute d? How do we backpropagate on d?

Goodfellow, Pouget-Abadie, Mirza, Xu, Warde-Farley, Ozair, Courville, Bengio, Generative Adversarial Networks, NeurIPS, 2014.

Generative Adversarial Networks

For many distance measures (Jensen–Shannon, Wasserstein distance)

$$d\left(p^{\text{gen}}_{\theta_G}, p^{\text{true}}\right) = \underset{D}{\text{maximize}} \quad \mathbb{E}_{X \sim p^{\text{true}}}[\text{something with } D(X)] + \mathbb{E}_{X \sim p^{\text{gen}}_{\theta_G}}[\text{something with } D(X)]$$

Problem: This is a maximization over all functions $D: \mathcal{X} \to [0,1]$. Not practical.

Goodfellow, Pouget-Abadie, Mirza, Xu, Warde-Farley, Ozair, Courville, Bengio, Generative Adversarial Networks, NeurIPS, 2014.

Generative Adversarial Networks

Solution: make D a neural network

$$\begin{aligned} &d\left(p^{\text{gen}}_{\theta_G}, p^{\text{true}}\right) \\ &= \underset{\theta_D}{\text{maximize}} \quad \mathbb{E}_{X\sim p^{\text{true}}}\left[\text{something with } D_{\theta_D}(X)\right] + \mathbb{E}_{X\sim p^{\text{gen}}_{\theta_G}}\left[\text{something with } D_{\theta_D}(X)\right] \\ &= \underset{\theta_D}{\text{maximize}} \quad \mathbb{E}_{X\sim p^{\text{true}}}\left[\text{something with } D_{\theta_D}(X)\right] + \mathbb{E}_{Z\sim\mathcal{N}}\left[\text{something with } D_{\theta_D}\left(G_{\theta_G}(Z)\right)\right] \end{aligned}$$

Finally, solve the minimax problem

$$\underset{\theta_G}{\text{minimize}}\,\underset{\theta_D}{\text{maximize}} \quad \mathbb{E}_{X\sim p^{\text{true}}}\left[\text{something with } D_{\theta_D}(X)\right] + \mathbb{E}_{Z\sim\mathcal{N}}\left[\text{something with } D_{\theta_D}\left(G_{\theta_G}(Z)\right)\right]$$

Generative Adversarial Networks

Original GAN formulation

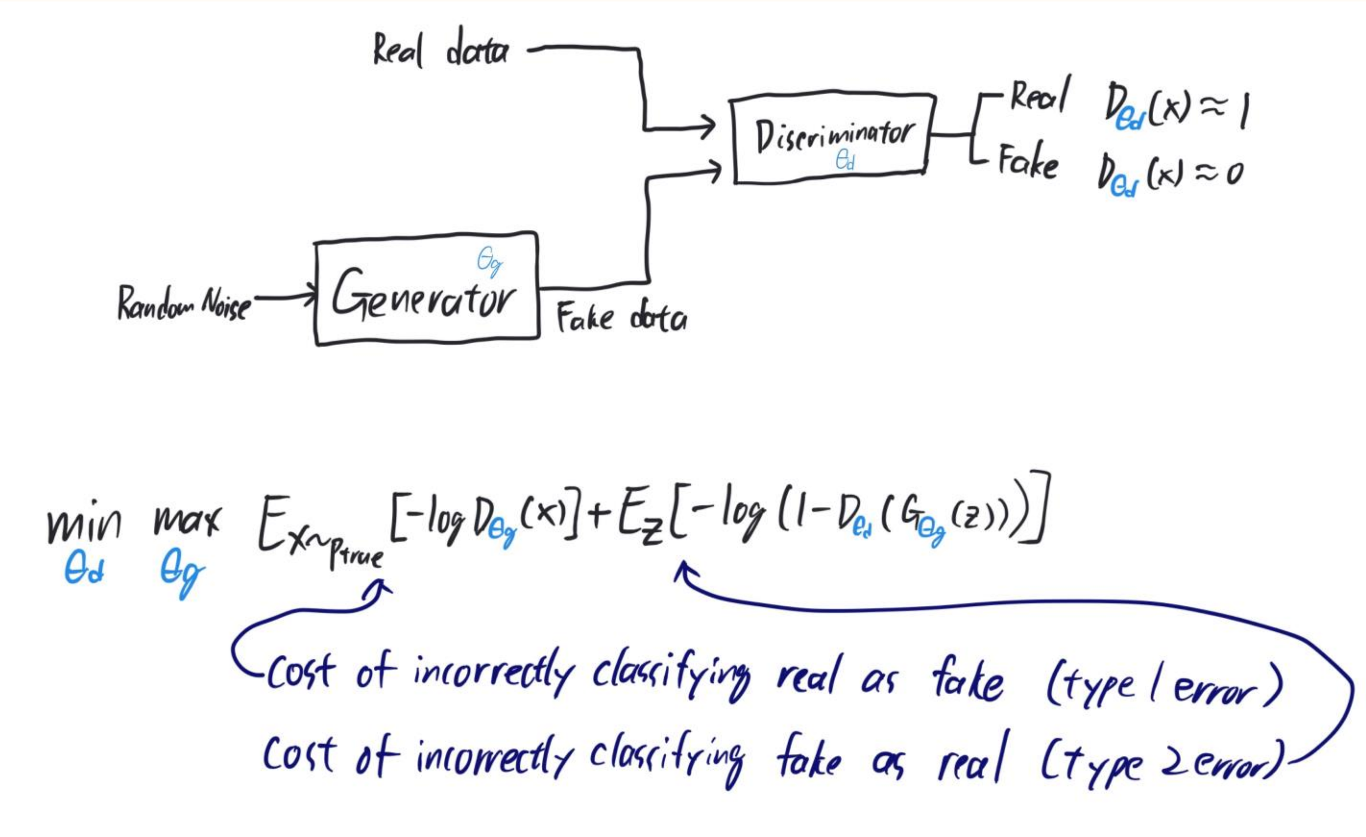
$$\underset{\theta_G}{\text{minimize}}\,\underset{\theta_D}{\text{maximize}}\quad \mathbb{E}_{X\sim p^{\text{true}}}\left[\log D_{\theta_D}(X)\right] + \mathbb{E}_{Z\sim\mathcal{N}}\left[\log\left(1 - D_{\theta_D}\left(G_{\theta_G}(Z)\right)\right)\right]$$

- Interpretation 1: minimize the Jensen–Shannon divergence between $p_{\theta_G}^{\text{gen}}$ and p^{true}.

Goodfellow, Pouget-Abadie, Mirza, Xu, Warde-Farley, Ozair, Courville, Bengio, Generative Adversarial Networks, NeurIPS, 2014.

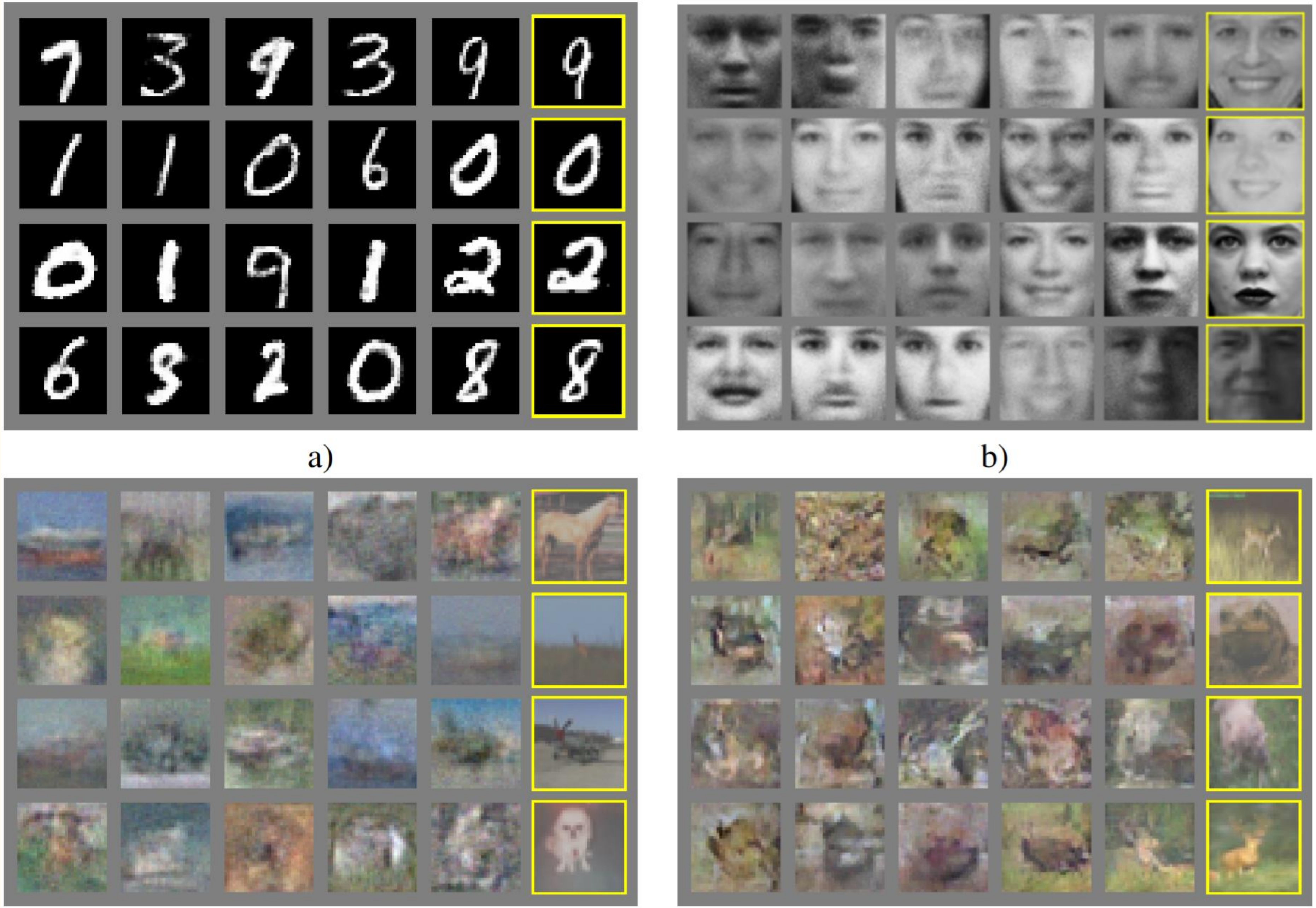
Generative Adversarial Networks

- Interpretation 2: The two networks are adversarially competing and improve together.



Results

Not very strong, but interesting starting point.



Goodfellow, Pouget-Abadie, Mirza, Xu, Warde-Farley, Ozair, Courville, Bengio, Generative Adversarial Networks, NeurIPS, 2014.

Code Demo

Convolution and its Transpose

The convolutions in deep learning are linear operators.

Its “transpose” or “adjoint” operations have two important uses.

1. Implementation of backprop requires application of transposes.

2. The transpose convolution operations are used in architectures where the output is an image.

1D Convolution and its Transpose

Consider 1D convolution with $f=3$, $s=1$, $p=1$.

$y=$ Conv1D(in_channels=1, out_channels=1, kernel_size=3, stride=1, padding=1) (x)

can be represented as $y=Ax$, where

$$A = \begin{bmatrix} k_2 & k_3 & 0 & 0 & 0 & & \\ k_1 & k_2 & k_3 & 0 & 0 & & \\ 0 & k_1 & k_2 & k_3 & 0 & & \\ 0 & 0 & & \ddots & & & 0 \\ & & & 0 & k_1 & k_2 & k_3 \\ & & & 0 & 0 & k_1 & k_2 \end{bmatrix} \in \mathbb{R}^{n\times n}$$

The transpose of this matrix is

$$A^T = \begin{bmatrix} k_2 & k_1 & 0 & 0 & 0 & & \\ k_3 & k_2 & k_1 & 0 & & & \\ 0 & k_3 & k_2 & k_1 & & & \\ 0 & 0 & & \ddots & & & 0 \\ & & & & k_3 & k_2 & k_1 \\ & & & & 0 & k_3 & k_2 \end{bmatrix} \in \mathbb{R}^{n\times n}$$

1D Convolution and its Transpose

Consider 1D convolution with $f=3$, $s=2$, $p=0$.

$y=$ Conv1D(in_channels=1, out_channels=1, kernel_size=3, stride=2, padding=0) (x)

can be represented as $y=Ax$, where

$$A = \begin{bmatrix} k_1 & k_2 & k_3 & 0 & 0 & 0 & & & & \\ 0 & 0 & k_1 & k_2 & k_3 & 0 & & & & \\ & 0 & 0 & 0 & k_1 & k_2 & k_3 & & & \\ & & & & & & \ddots & & & \\ & & & & & & k_1 & k_2 & k_3 & 0 \\ & & & & & & 0 & 0 & k_1 & k_2 & k_3 \end{bmatrix} \in \mathbb{R}^{(\frac{n-3}{2}+1)\times n}$$

(for odd n)

The transpose of this matrix is

$$A^T = \begin{bmatrix} k_1 & 0 & 0 & & \\ k_2 & 0 & 0 & & \\ k_3 & k_1 & 0 & & \\ 0 & k_2 & 0 & & \\ 0 & k_3 & \ddots & & \\ & & & k_1 & 0 \\ & & & k_2 & 0 \\ & & & k_3 & k_1 \\ & & & 0 & k_2 \\ & & & 0 & k_3 \end{bmatrix} \in \mathbb{R}^{n\times(\frac{n-3}{2}+1)}$$

1D Convolution and its Transpose

For example, consider

$$\underset{x}{\text{minimize}} \quad \frac{1}{2}\|\text{conv}(x) - y\|^2$$

gradient descent is

$$x^{k+1} = x^k - \alpha_k \text{transpose\_conv}\left(\text{conv}\left(x^k\right) - y\right)$$

2D Convolution and its Transpose

2D or 3D convolution with multiple input and output channels are also linear operators.

Other names:

- Fractionally strided convolution (sounds sophisticated, but not very clear)
- Deconvolution (bad name because “deconvolution” means the inverse of a convolution operation in classical signal processing)
- Transpose convolution (now the standard terminology. Best name)

Deep convolutional generative adversarial networks (DCGAN)

- Key contribution: presented architecture guidelines and demonstrated a marked improvement in performance for GANs.

- Original GAN was also deep and some were convolutional. However, DCGAN presented improved architectures.

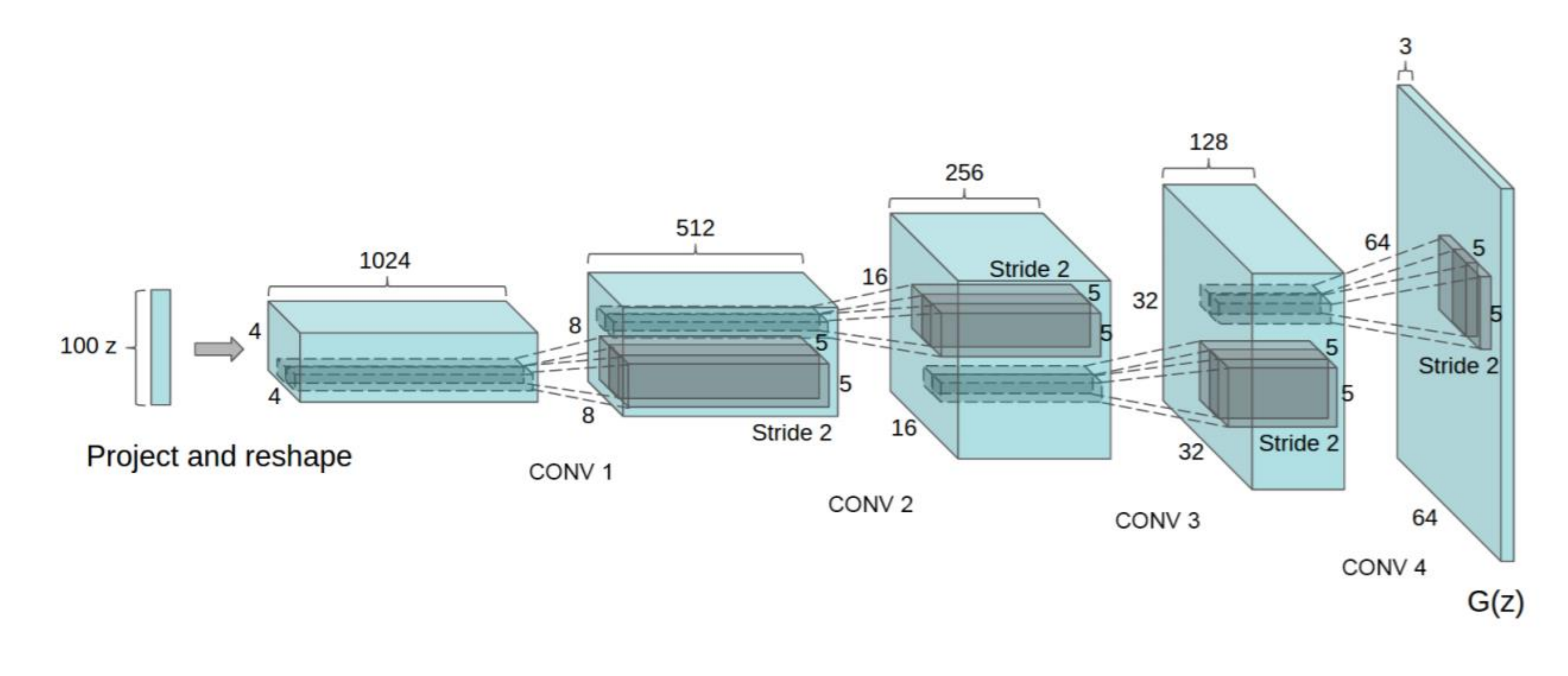
Radford, Metz, Chintala, Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, ICLR, 2016.

Deep convolutional generative adversarial networks (DCGAN)

- Don't use pooling layers. Use strided convolutions for discriminator and strided transpose convolutions for generator.
- Use batch norm.
- Remove (minimize) use of fully connected layers.
- Use ReLU in generator, except for the generator output which uses tanh. Image needs output between [0,1] so that it can be scaled and rounded to $\{0,1,\ldots,255\}$.
- Use LeakyReLU activation in the discriminator for all layers.

Radford, Metz, Chintala, Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, ICLR, 2016.

DCGAN Generator architecture



The “CONV X” is really a transpose convolution, which is why it increases the spatial dimension.

DCGAN Results

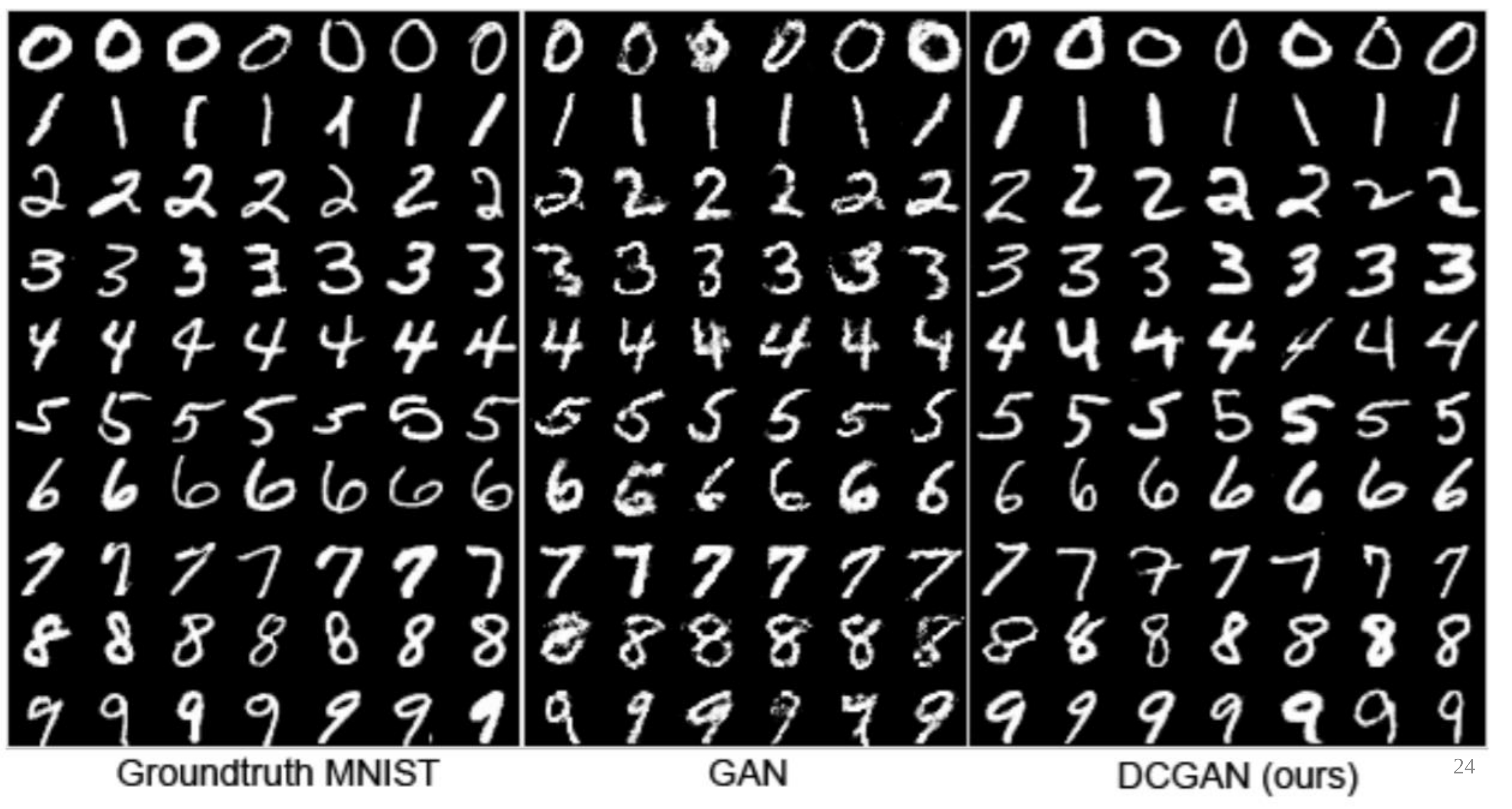


Figure 10: More face generations from our Face DCGAN.

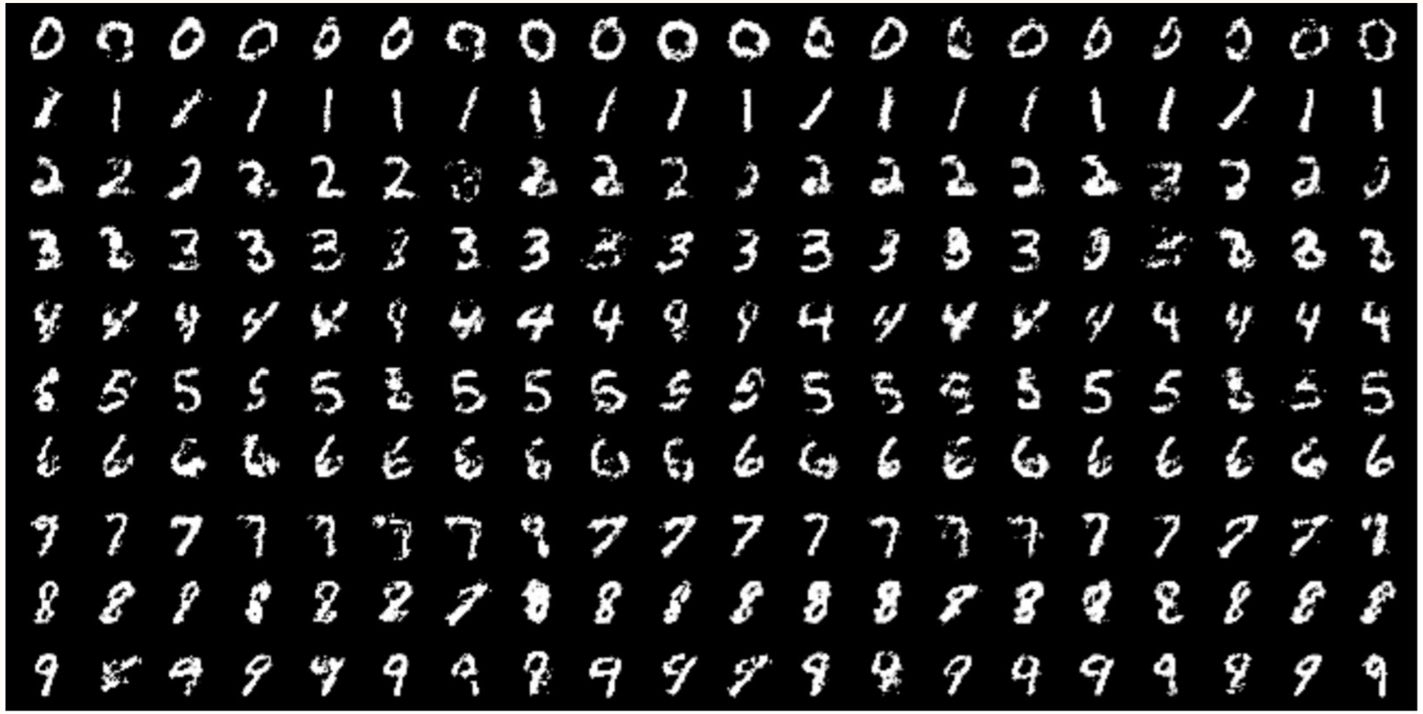
DCGAN Results

Figure 11: Generations of a DCGAN that was trained on the Imagenet-1k dataset.

Conditional GANs

Conditional GAN formulation

$$\underset{\theta_G}{\text{minimize}}\,\underset{\theta_D}{\text{maximize}}\quad \mathbb{E}_{(X,Y)\sim p^{\text{true}}}\left[\log D_{\theta_D}(X|Y)\right] + \mathbb{E}_{Z\sim\mathcal{N},Y}\left[\log\left(1 - D_{\theta_D}\left(G_{\theta_G}(Z|Y)|X\right)\right)\right]$$

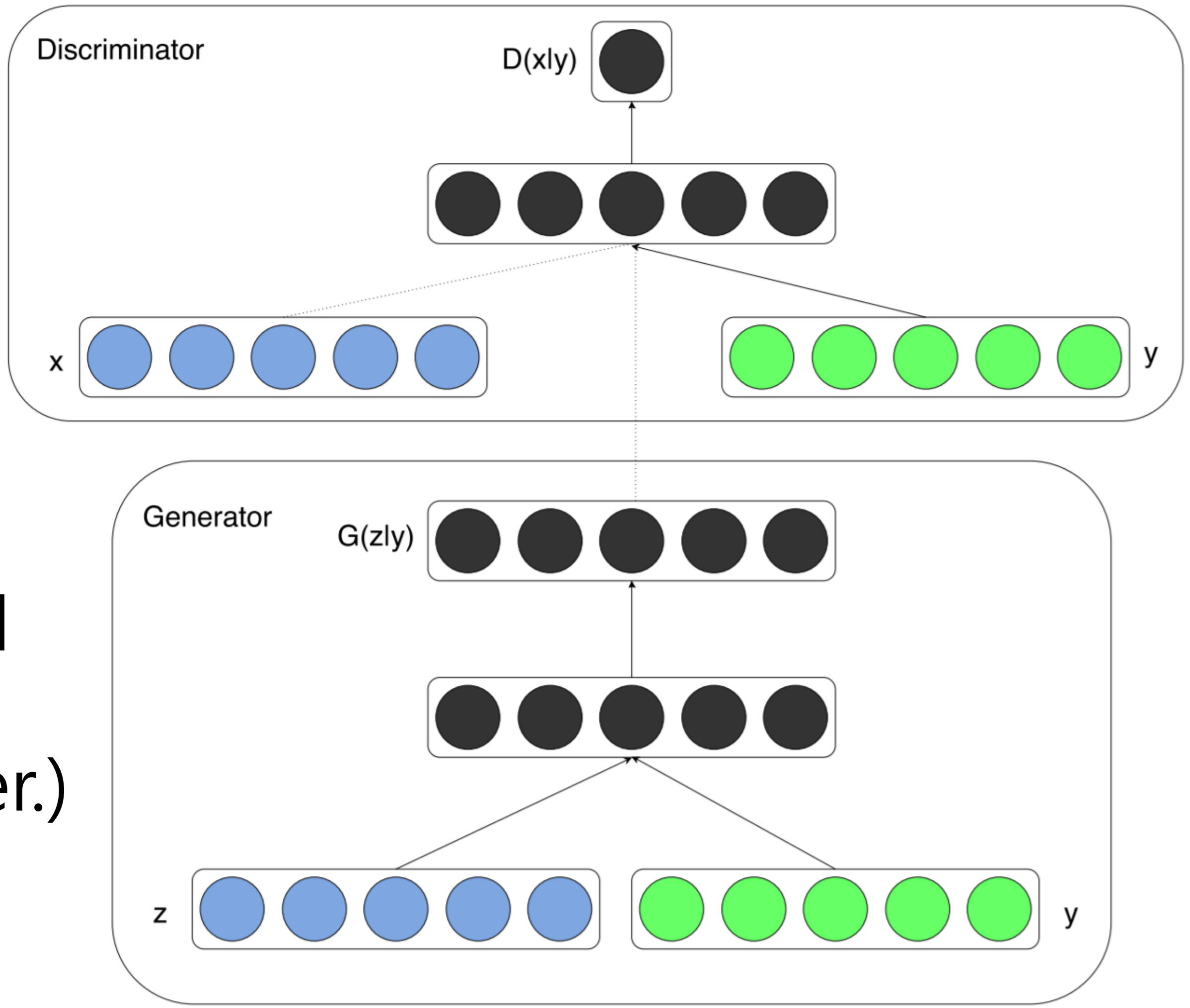


Mirza, Osindero, Conditional Generative Adversarial Nets, arXiv, 2014.

Conditional GANs

Architecture: Construct a single generator and discriminator that combines data X and label Y.

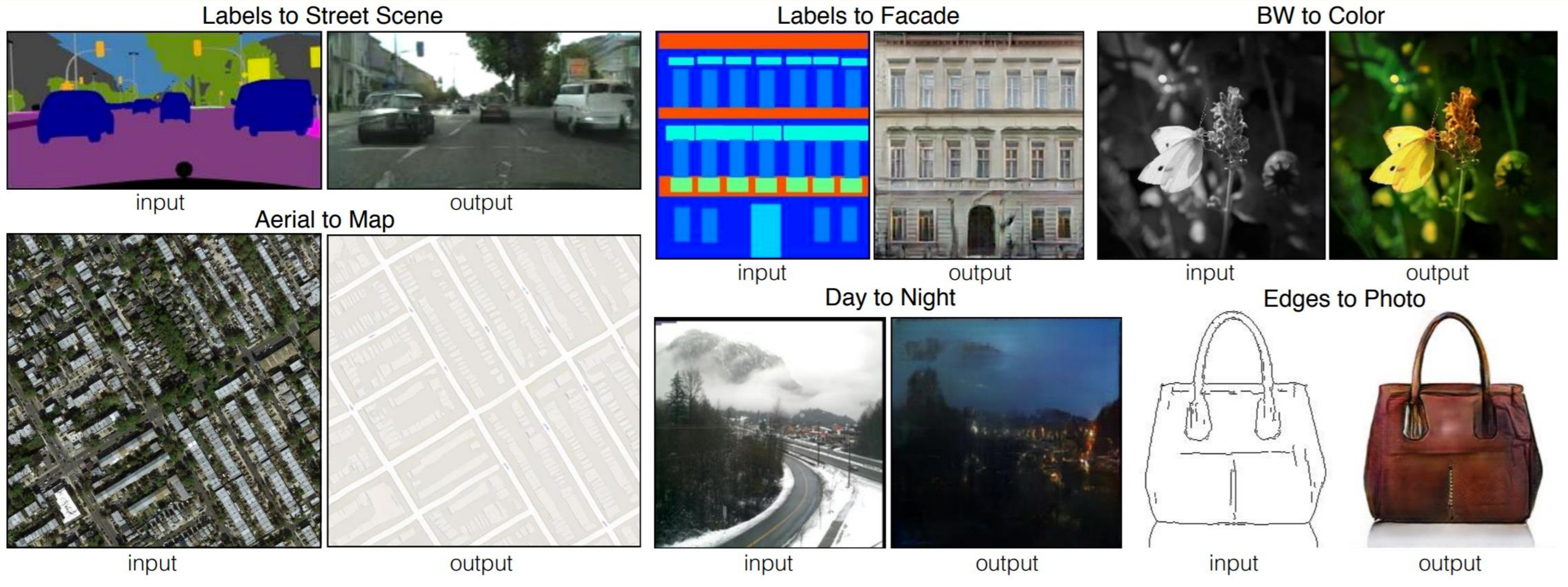
(One could have a separate D and G for each value of Y, but a combined network performs better.)



Mirza, Osindero, Conditional Generative Adversarial Nets, arXiv, 2014.

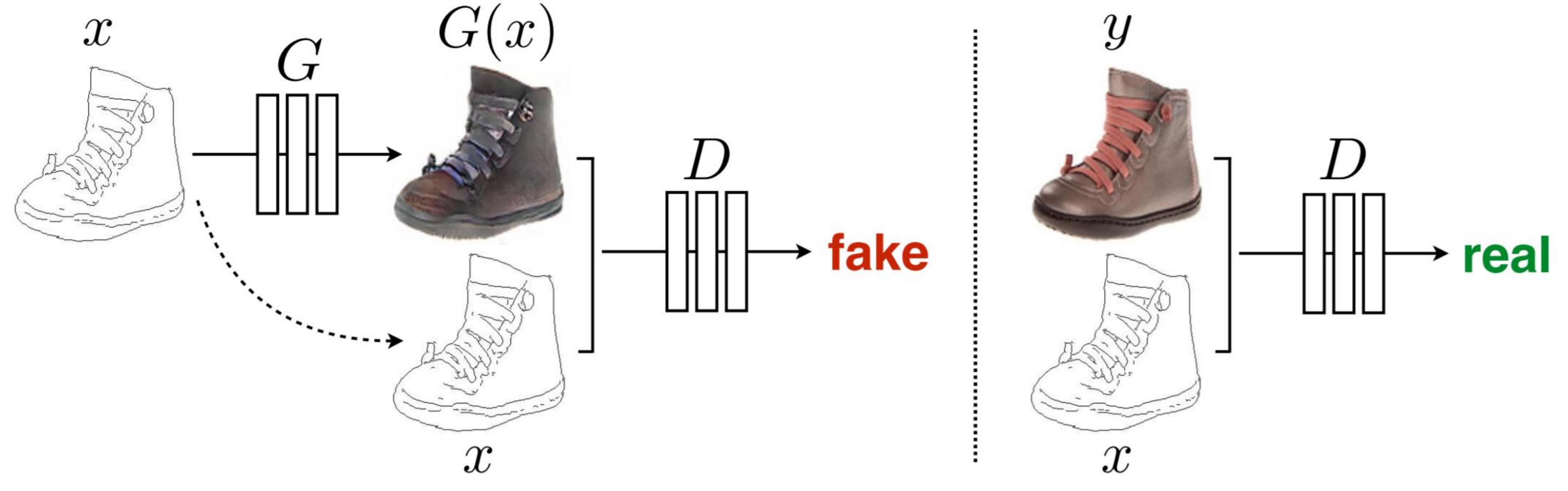
Pix2Pix

Goal: For (X, Y), we observe X and reconstruct Y.



Isola, Zhu, Zhou, Efros, Image-to-Image Translation with Conditional Adversarial Nets, CVPR, 2017.

Pix2Pix

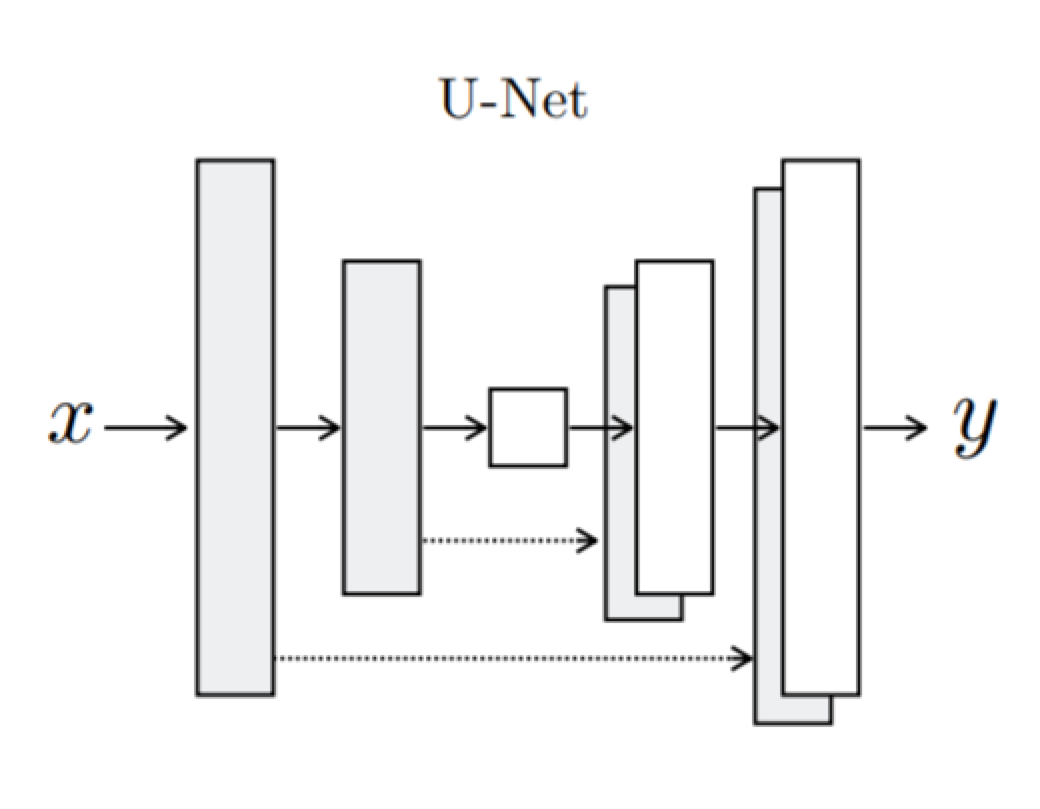


(In Mirza and Osindero's conditional GAN paper, image is X and discrete label is Y. In this paper, image is Y and "label" or auxiliary data is X.)

Pix2Pix: UNet architecture

Generator uses Unet architecture

- Reduce spatial dimension with conv.
- Increase spatial dimension with transpose conv and concatenate with previous layers.



Pix2Pix: Adversarial loss

$$\underset{\theta_G}{\text{minimize}}\,\underset{\theta_D}{\text{maximize}}\quad \mathcal{L}_1 + \mathcal{L}_2 + \mathcal{L}_3$$

- $\mathcal{L}_1 = \mathbb{E}_{(X,Y)\sim p^{\text{true}}}\left[\log D_{\theta_D}(Y|X)\right]$
- $\mathcal{L}_2 = \mathbb{E}_{(X,Y)\sim p^{\text{true}},\, Z\sim\mathcal{N}}\left[\log\left(1 - D_{\theta_D}\left(G_{\theta_G}(X|Z)|X\right)\right)\right]$
- $\mathcal{L}_3 = \lambda\mathbb{E}_{(X,Y)\sim p^{\text{true}}Z\sim\mathcal{N}}\left[\left\|y - G_{\theta_G}(X|Z)\right\|_1\right]$

- $\mathcal{L}_1$ represents loss for incorrectly classifying true image Y as not real.
- $\mathcal{L}_2$ represents loss for incorrectly classifying reconstruction image $G_{\theta_G}(X|Z)$ is real. (Y is not used in this loss, but the X is information for which a corresponding real information exists.)
- $\mathcal{L}_3$ represents different between reconstruction and original.

Pix2Pix: Source of randomness

- Interestingly, generator takes in no explicit randomness. Randomless injected only through dropout.
- Dropout is used during test time as well.
- When generator is offered an input Gaussian noise, it learns to ignore it.

Isola, Zhu, Zhou, Efros, Image-to-Image Translation with Conditional Adversarial Nets, CVPR, 2017.

BigGAN

Combination of many GAN techniques with very large computation.

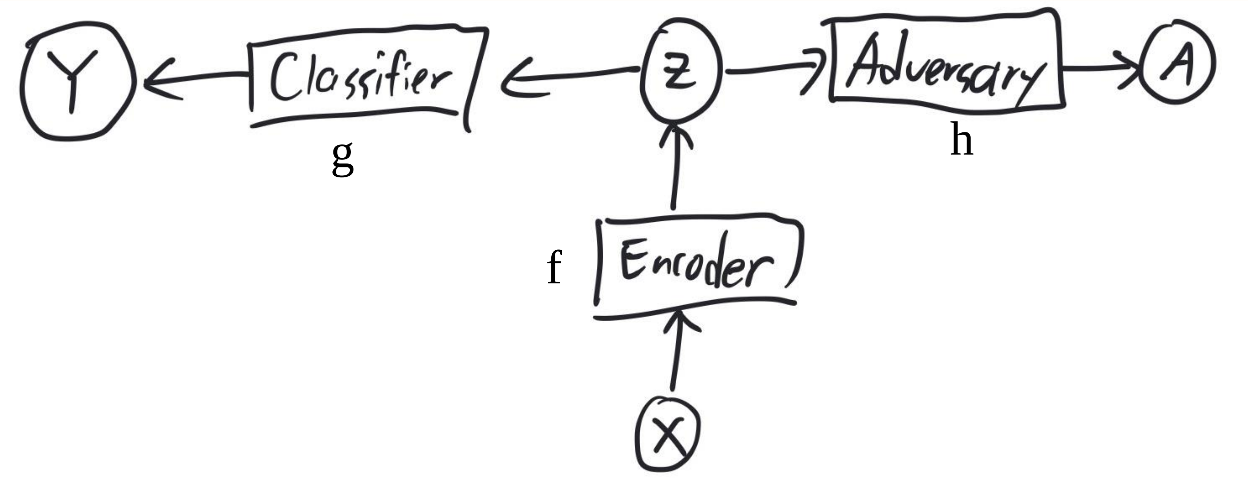
Brock, Donahue, Simonyan, Large Scale GAN Training for High Fidelity Natural Image Synthesis, ICLR, 2019.

Adversarially Fair Representations

Classification setting

$$\mathcal{L} = \mathcal{L}_{\text{classification}} + \mathcal{L}_{\text{adversarial}}$$

$$\mathcal{L}_{\text{adversarial}} = \sum_{(x,a)} |h(f(x)) - a|$$



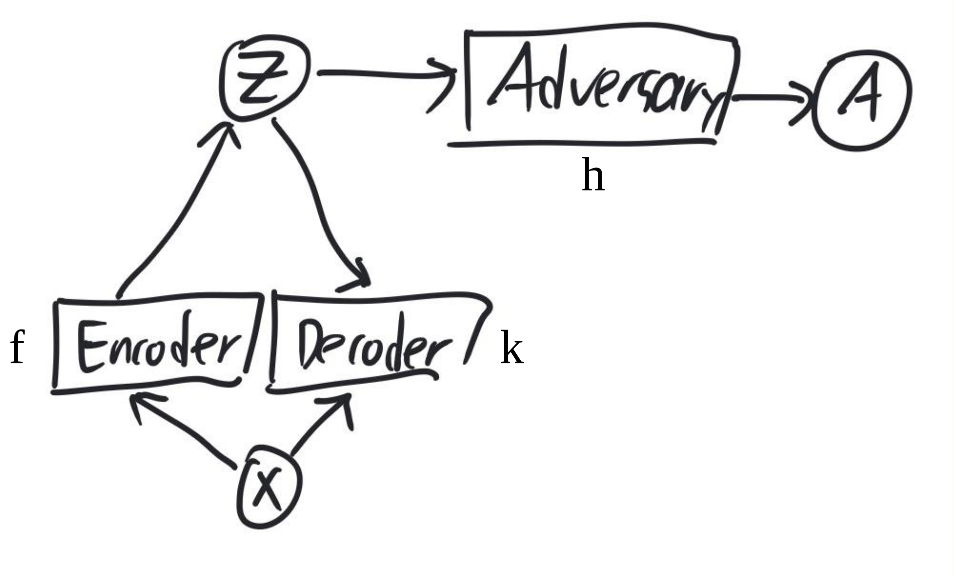
Madras, Creager, Pitassi, Zemel, Learning Adversarially Fair and Transferable Representations, ICML, 2018.

Adversarially Fair Representations

Transfer learning setting

$$\mathcal{L} = \mathcal{L}_{\text{reconstruction}} + \mathcal{L}_{\text{adversarial}}$$

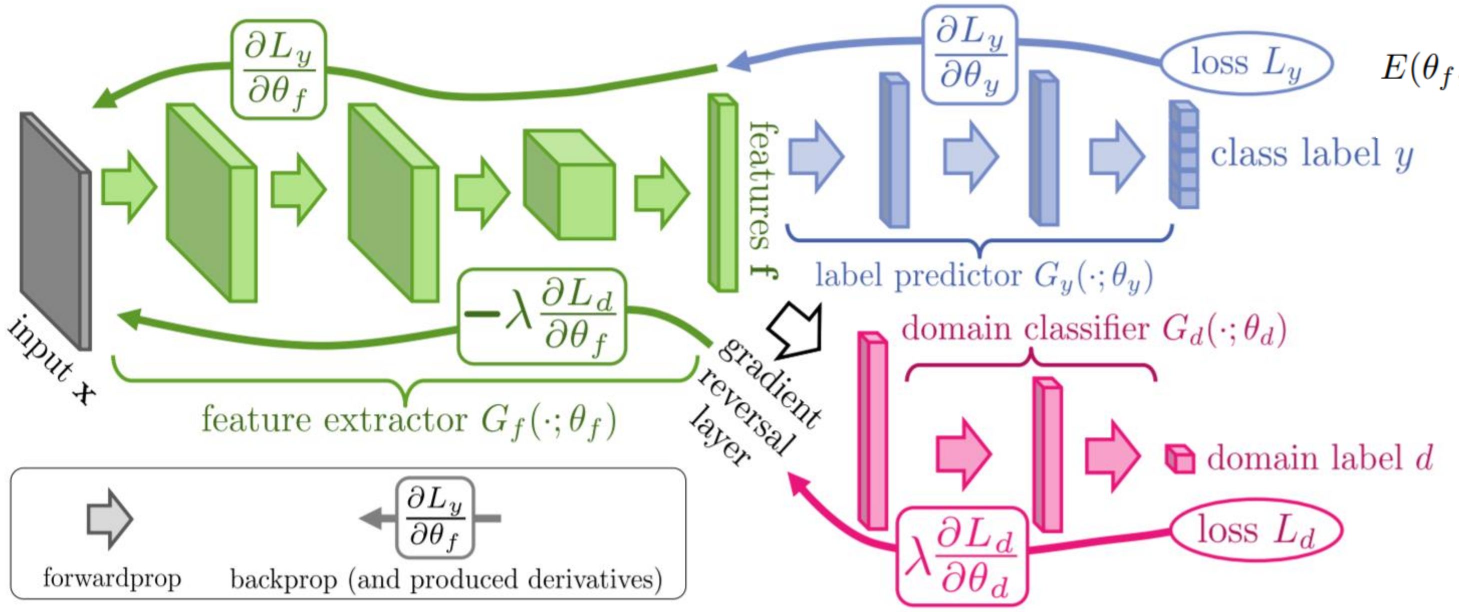
$$\mathcal{L}_{\text{reconstruction}} = \sum_{x} \left\| k\big(f(x)\big) - x \right\|^2$$



Domain-Adversarial Networks

Use adversarial training to extract features f so that a discriminator cannot distinguish $\mathcal{P}$ from $\mathcal{Q}$.Train classifier to make decision only on these extracted features

$$\mathcal{L}_y^i(\theta_f,\theta_y) = \mathcal{L}_y\big(G_y(G_f(\mathbf{x}_i;\theta_f);\theta_y),y_i\big)$$
$$\mathcal{L}_d^i(\theta_f,\theta_d) = \mathcal{L}_d\big(G_d(G_f(\mathbf{x}_i;\theta_f);\theta_d),d_i\big)$$



$$E(\theta_f,\theta_y,\theta_d) = \frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_y^i(\theta_f,\theta_y) - \lambda\Big(\frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_d^i(\theta_f,\theta_d) + \frac{1}{n'}\sum_{i=n+1}^{N}\mathcal{L}_d^i(\theta_f,\theta_d)\Big)$$

$$(\hat{\theta}_f,\hat{\theta}_y) = \underset{\theta_f,\theta_y}{\operatorname{argmin}}\, E(\theta_f,\theta_y,\hat{\theta}_d)$$
$$\hat{\theta}_d = \underset{\theta_d}{\operatorname{argmax}}\, E(\hat{\theta}_f,\hat{\theta}_y,\theta_d)$$

$$\theta_f \longleftarrow \theta_f - \mu\left(\frac{\partial\mathcal{L}_y^i}{\partial\theta_f} - \lambda\frac{\partial\mathcal{L}_d^i}{\partial\theta_f}\right),$$
$$\theta_y \longleftarrow \theta_y - \mu\frac{\partial\mathcal{L}_y^i}{\partial\theta_y},$$
$$\theta_d \longleftarrow \theta_d - \mu\lambda\frac{\partial\mathcal{L}_d^i}{\partial\theta_d},$$

Ganin, Ustinova, Ajakan, Germain, Larochelle, Laviolette, Marchand, Lempitsky, Domain-Adversarial Training of Neural Networks, JMLR, 2016.