

A Hybrid DL and GM (cont'd)

+

Applications in Computer Vision

Kayhan Batmanghelich

Mingming Gong

Outlines

- Joint deep feature extraction and learning
- Variational Auto Encoder (VAE)
- Generative Adversarial Networks (GANs)

Recap: Variational Auto Encoder (VAE)

$$\max_{\phi, \theta} \mathcal{F}(x, q_\phi) = \mathbb{E}_{q_\phi(z)} [\log p_\theta(x|z)] - KL [q_\phi(z)||p(z)]$$

Stochastic encoder Data code-length Hypothesis code

$$q_\phi(\mathbf{z}|\mathbf{x}) = q_\phi(\mathbf{z}_1, \dots, \mathbf{z}_M|\mathbf{x}) = \prod_{j=1}^M q_\phi(\mathbf{z}_j|Pa(\mathbf{z}_j), \mathbf{x})$$
$$z \sim q_\phi(z|x)$$

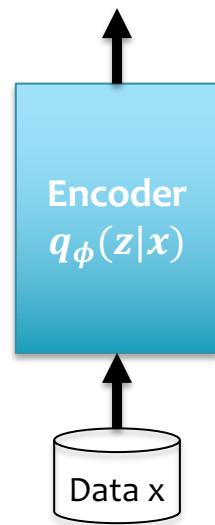


Amortization
(sharing parameters)

Example:

$$(\mu, \log \sigma) = \text{EncoderNeuralNet}_\phi(\mathbf{x})$$

$$q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}; \mu, \text{diag}(\sigma))$$



Recap: Mapping the Distributions

- VAE learns a mapping that transforms a simple distribution, $q(z)$, to a complicate distribution, $q(z|x)$ so that the empirical distribution (in the x -space) are matched.

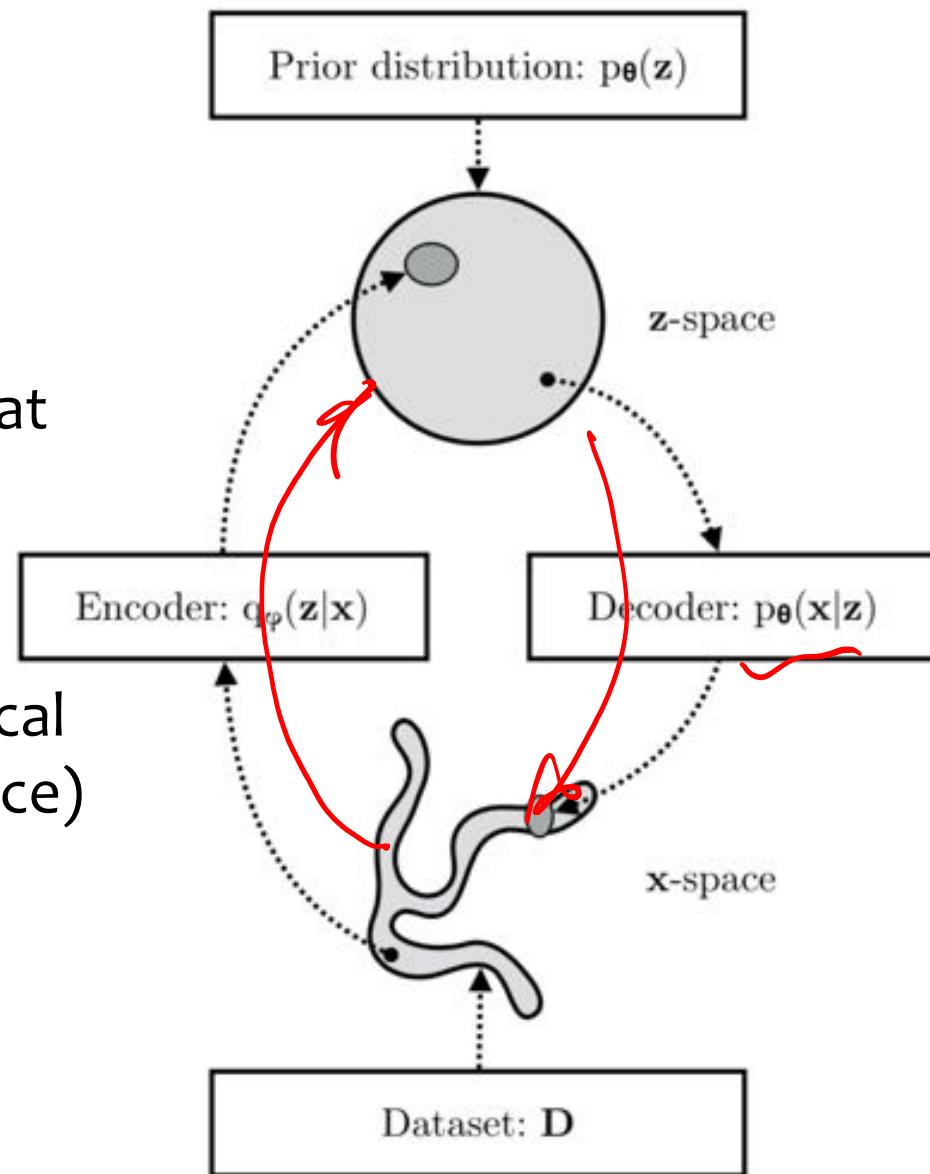


Fig Credit: Kingma's thesis

Recap: Reparametrization Trick

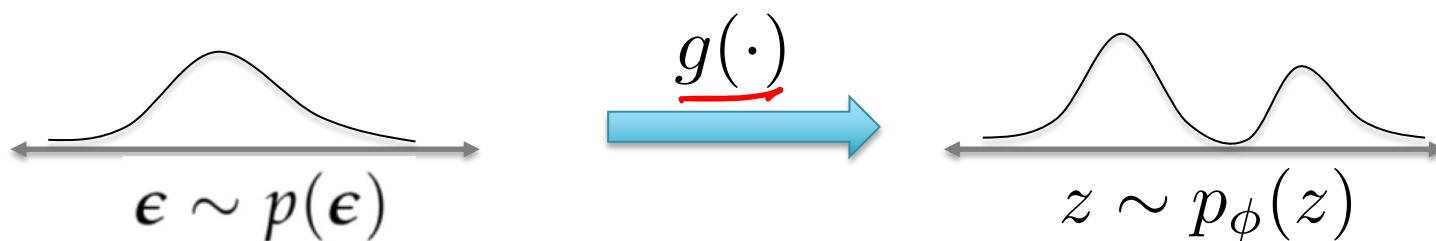
$$\mathcal{L}_{\theta, \phi}(\mathbf{x}) = \mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})} [\log p_{\theta}(\mathbf{x}, \mathbf{z}) - \log q_{\phi}(\mathbf{z}|\mathbf{x})]$$

$\mathbf{z} = g(\epsilon, \phi, \mathbf{x})$

$$= \mathbb{E}_{p(\epsilon)} [\log p_{\theta}(\mathbf{x}, \mathbf{z}) - \log q_{\phi}(\mathbf{z}|\mathbf{x})]$$

Can we improve the estimation of
gradient?
(reduce the variance)

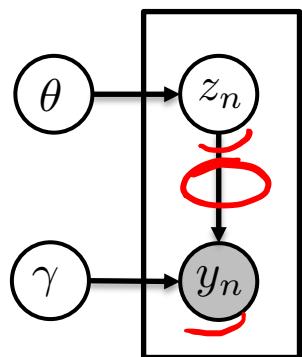
Let's revisit the change of variables:



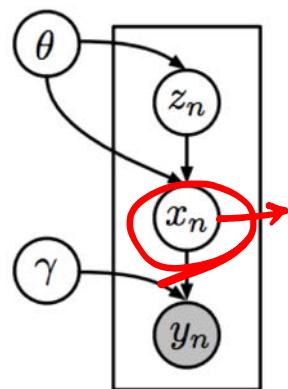
$$\log q_{\phi}(\mathbf{z}|\mathbf{x}) = \log p(\epsilon) - \log \left| \det \left(\frac{\partial \mathbf{z}}{\partial \epsilon} \right) \right|$$

Recap: How to use this technique in a Graphical Model?

Example: Mixture Model



$$\begin{aligned} \pi &\sim \text{Dir}(\alpha), \\ (\mu_k, \Sigma_k) &\stackrel{\text{iid}}{\sim} \text{NIW}(\lambda), \\ z_n | \pi &\stackrel{\text{iid}}{\sim} \pi \\ y_n | z_n, \{(\mu_k, \Sigma_k)\}_{k=1}^K &\stackrel{\text{iid}}{\sim} \mathcal{N}(\mu_{z_n}, \Sigma_{z_n}). \end{aligned}$$



$$\begin{aligned} \pi &\sim \text{Dir}(\alpha), \\ (\mu_k, \Sigma_k) &\stackrel{\text{iid}}{\sim} \text{NIW}(\lambda), \\ z_n | \pi &\stackrel{\text{iid}}{\sim} \pi \\ x_n &\stackrel{\text{iid}}{\sim} \mathcal{N}(\mu^{(z_n)}, \Sigma^{(z_n)}), \\ y_n | x_n, \gamma &\stackrel{\text{iid}}{\sim} \mathcal{N}(\mu(x_n; \gamma), \Sigma(x_n; \gamma)). \end{aligned}$$



(PCE)

$$z \sim N(\mu(x), \sigma(x))$$

Limitations of VAE



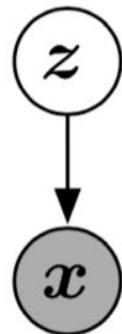
- When applied to image data, it results into blurry images.
- For images, it is sensitive to irrelevant variance, e.g., translations.
- Not applicable to discrete latent variables.
- Limited flexibility: converting the data distribution to fixed, single-mode prior distribution

GAN: GENERATIVE ADVERSARIAL NETWORK

GANs

- Introduced by Goodfellow et al., 2014
- Assumes implicit generative model
- Asymptotically consistent (unlike variational methods)
- No Markov chains needed
- Often regarded as producing the best samples (but,... no good way to quantify it)

Generator Network

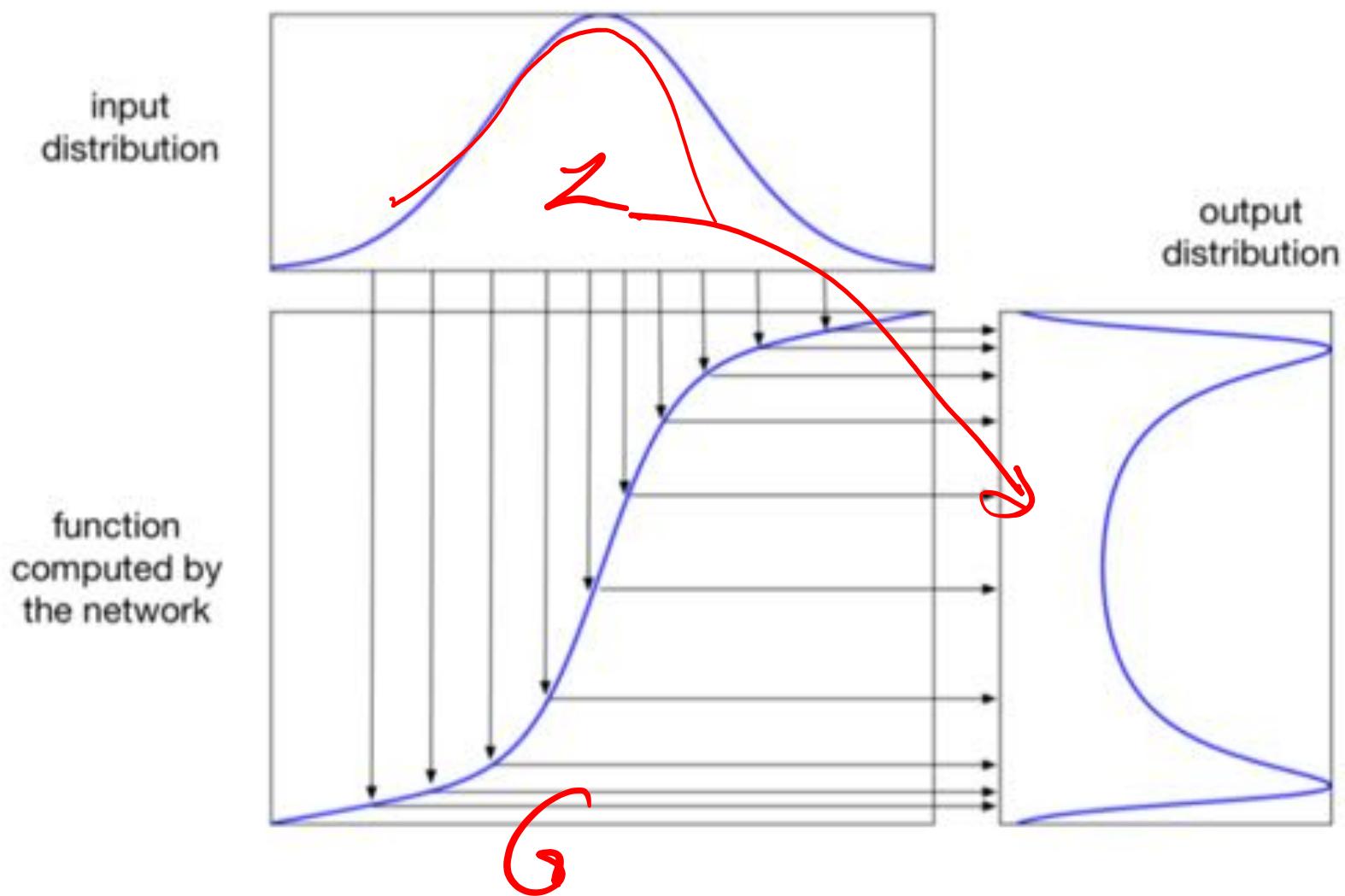


$$\underline{z \sim \mathcal{N}(0, I)}$$

$$\underline{x} = G(z; \underline{\theta^{(G)}})$$

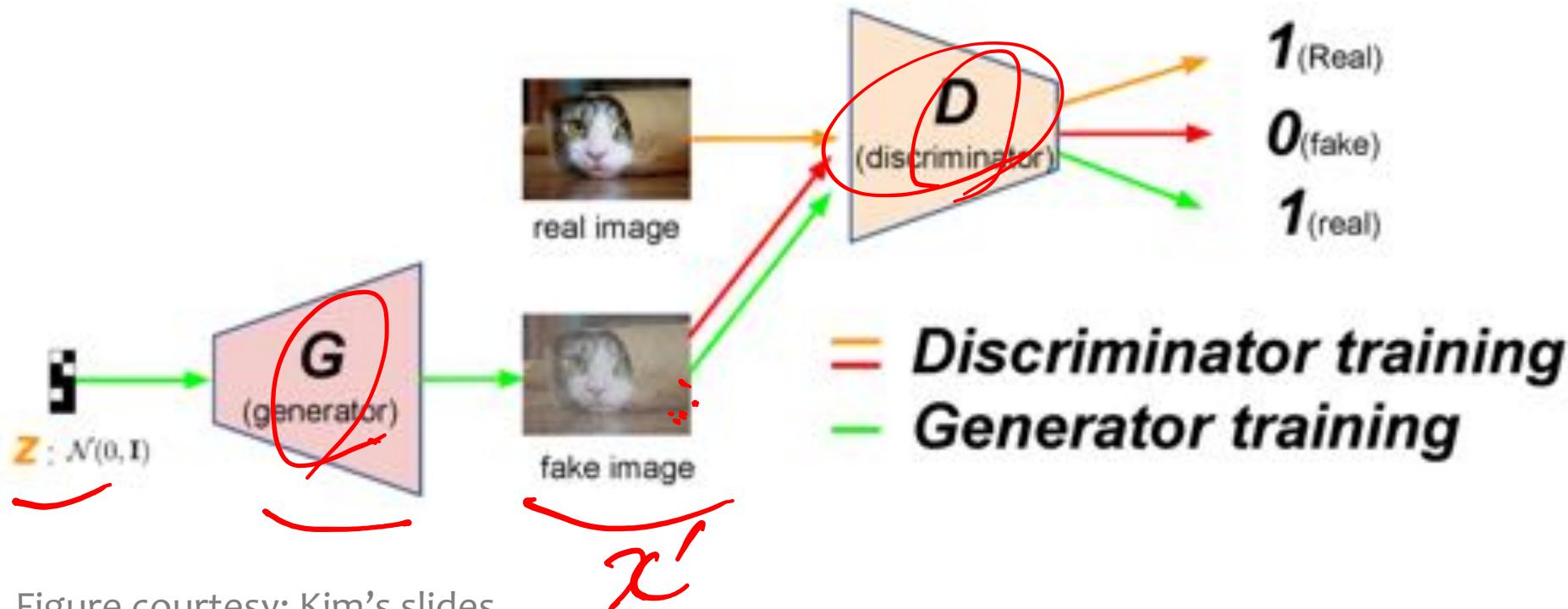
- Maps noise variable to the data space
- Must be differentiable but no invertibility is required

1-D Example



Learning

- Train D to discriminate between training examples and generated samples
- Train G to fool the discriminator



$$P(x) = D(x)^{\text{x is real}} \cdot (1 - D(x))^{\text{x is unreal}}$$

Minimax Game

$\min_G \max_D \mathbb{E}_{x \sim p_{\text{data}}(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))]$

Fixing G

$$\int_x p_{\text{data}}(x) \log(D(x)) + p_g(x) \log(1 - D(x)) dx$$

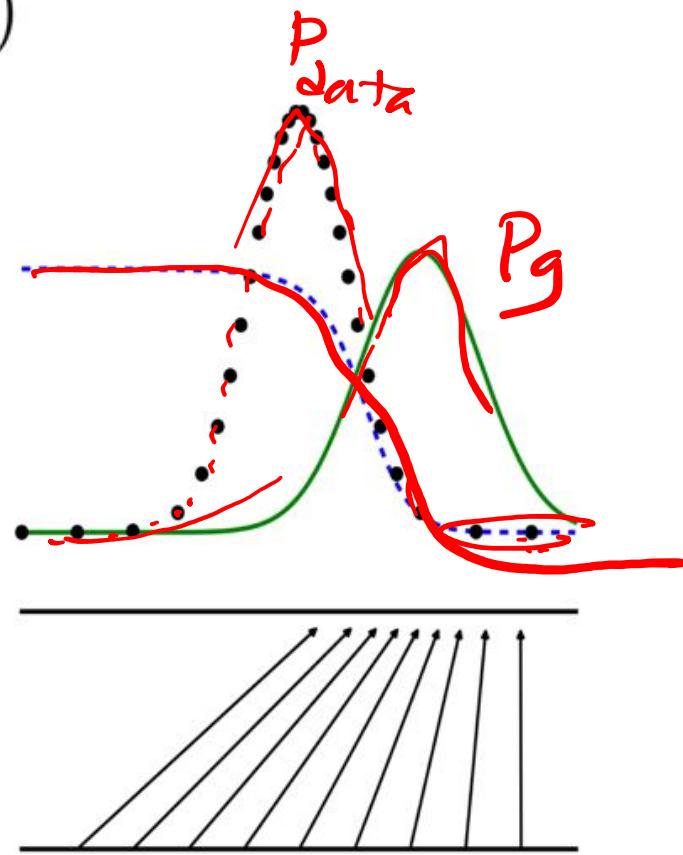
Taking derivative of
 $a \log(y) + b \log(1 - y)$

Optimal D:

$$D(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_{\text{model}}(x)}$$

Discriminator Strategy

$$D(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_{\text{model}}(x)}$$



$\mathbb{E}[\log]$
 $x \sim p$

Minimax Game

$$\min_G \max_D \mathbb{E}_{x \sim p_{\text{data}}(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))]$$

Substituting

$$D(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_{\text{model}}(x)}$$

$$\mathbb{E}_{x \sim p_{\text{data}}} \left[\log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_g(x)} \right] + \mathbb{E}_{x \sim p_g} \left[\log \frac{p_g(x)}{p_{\text{data}}(x) + p_g(x)} \right]$$

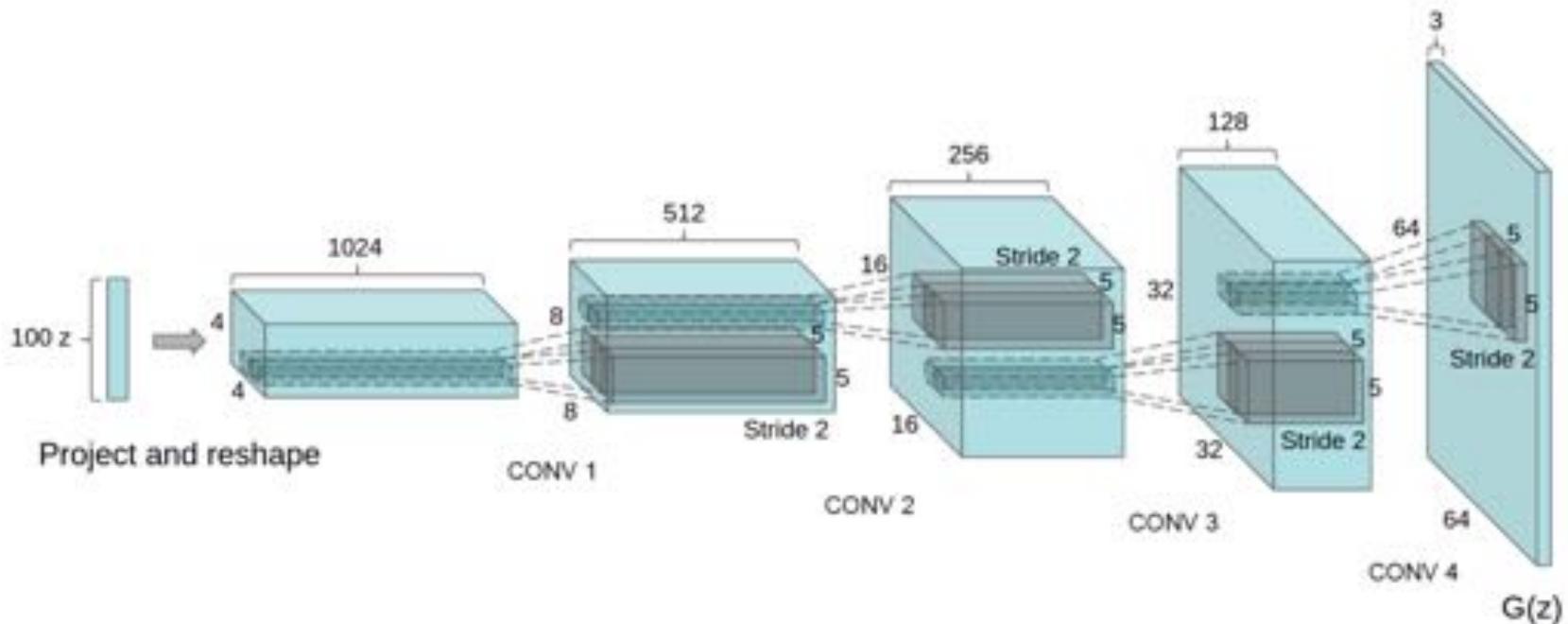
$KL(p_{\text{data}} \parallel \frac{p_{\text{data}} + p_g}{2})$ TCE
 $KL(p_g \parallel \frac{p_{\text{data}} + p_g}{2})$ TCE

$JSD(p_g, P)$

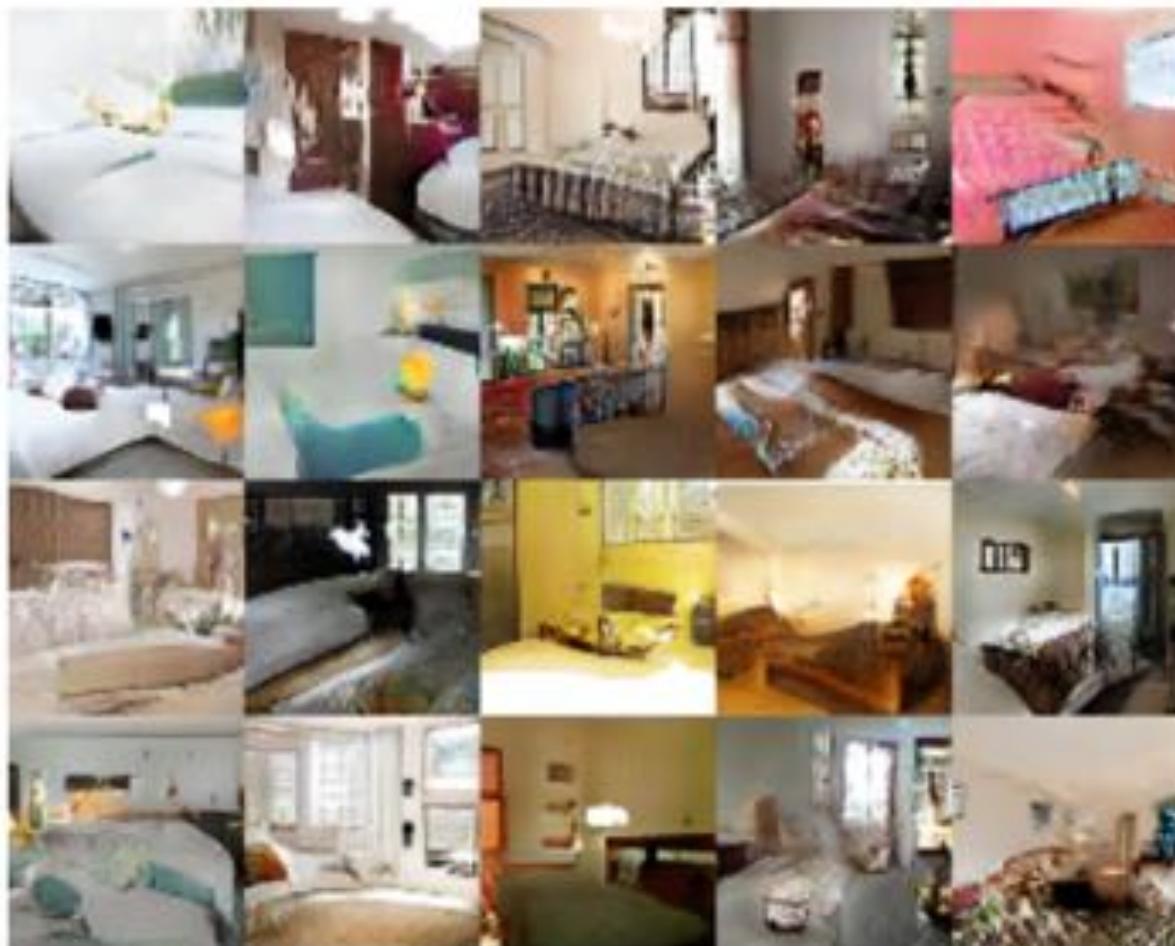
GANs minimizes the Jensen-Shannon divergence between two distributions.

More Stable Version

- DCGAN: Most “deconvs” are batch normalized



DCGANs for LSUN Bedrooms

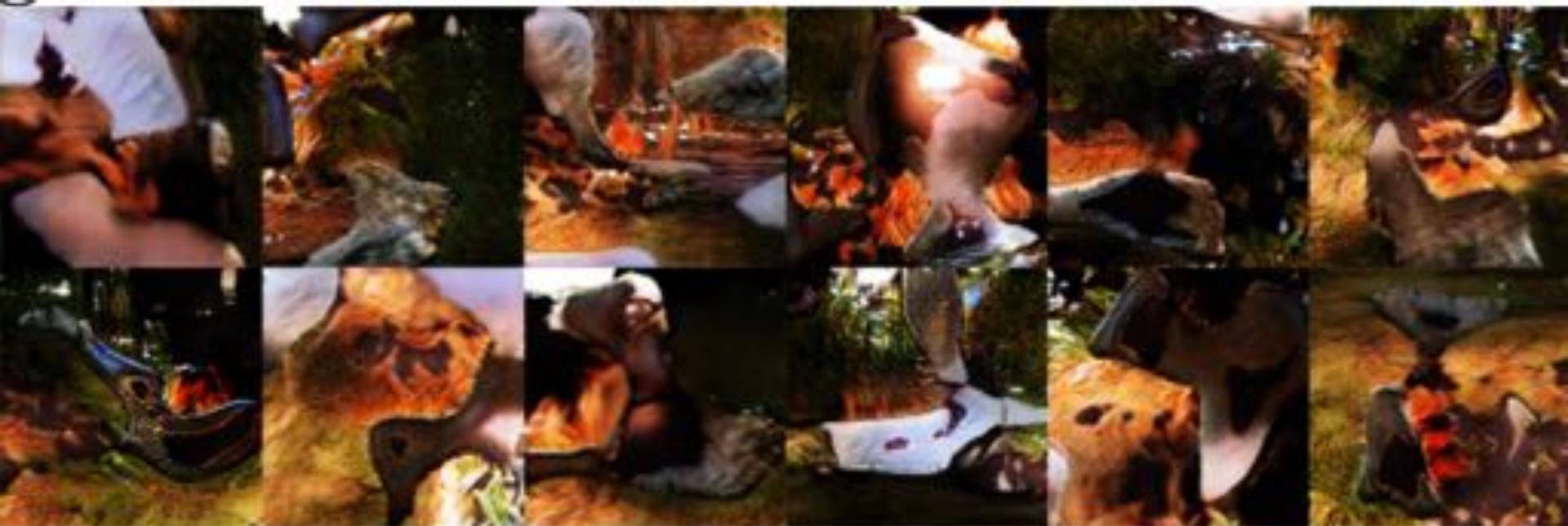


(Radford et al 2015)

Redford et al., “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks”

Practical Issues

Strong intra-batch correlation



Practical Issues

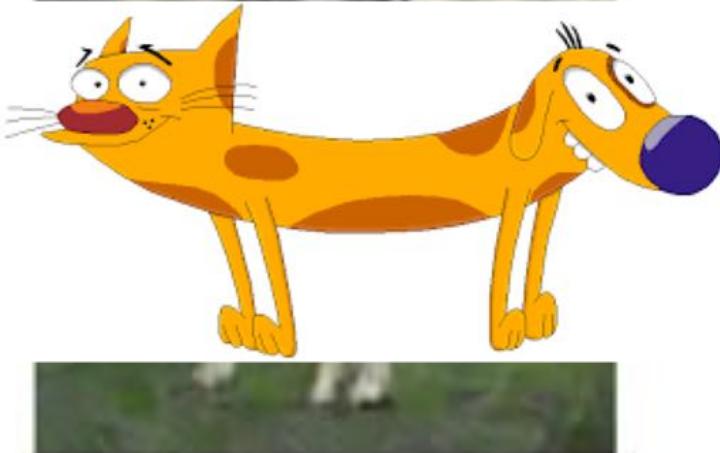
Mode Collapse



(Metz et al 2016)

(Goodfellow 2016)

Practical Issues



$$\begin{array}{c} z \rightarrow x \\ \nwarrow \uparrow \\ p(z|x) \end{array}$$

Summary

- Does not need a specific likelihood function at the last layer to train a latent-variable model
- Similar move from Exact inference to MCMC to var. inf: Don't restrict model to allow easy inference - just let a neural network clean up after.
- Topics of research:
 - How to compute $P(z|x)$, like VAE, without compromising the quality of samples?
 - How to make GANs more stable: various divergences, ... ?
 - How to handle discrete inputs ?
 - How to generalize to idea to GM. Notable works ?
 - “Averserial Variational Bayes”, Sebastian Nowozin, Andreas Geiger, 2017
 - “Learning in Implicit Generative Models,” Shakir Mohamed, Balaji Lakshminarayanan, 2016
 - “Variational Inference using Implicit Distributions,” Ferenc Huszar, 2017
 - “Deep and Hierarchical Implicit Models.” Dustin Tran, Rajesh Ranganath, David Blei, 2017

Applications in Vision

Mingming Gong

Outline

- Semantic Segmentation
- Image-to-Image Translation

Semantic Segmentation

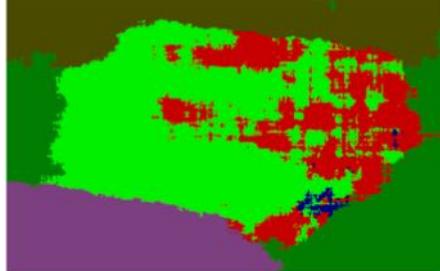
- Partition an image into semantically meaningful regions



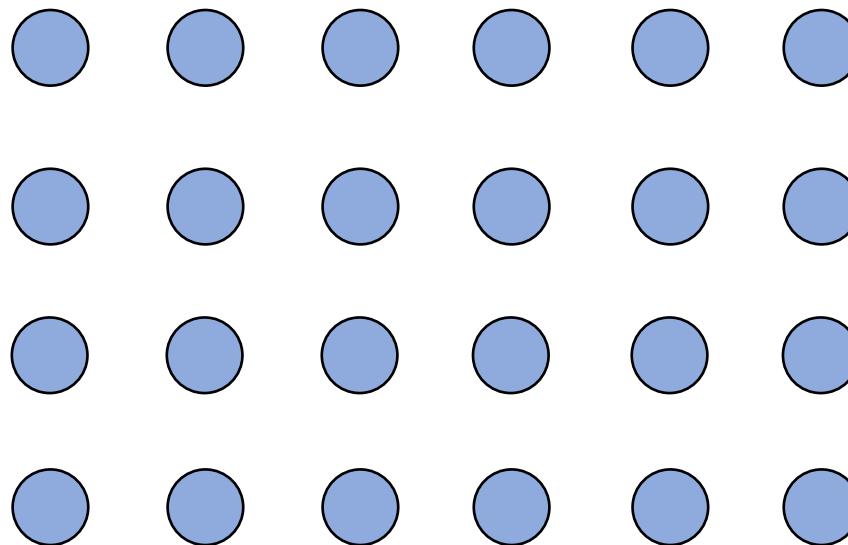
Pixel-wise Classification



(a) Image



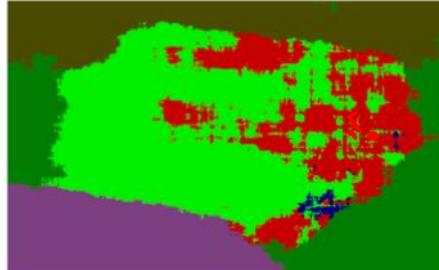
(b) Unary classifiers



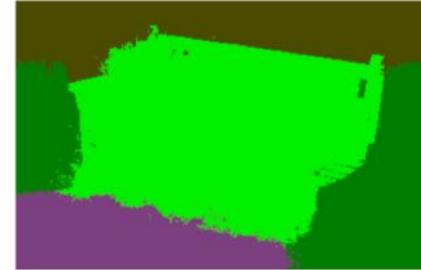
Gird CRF



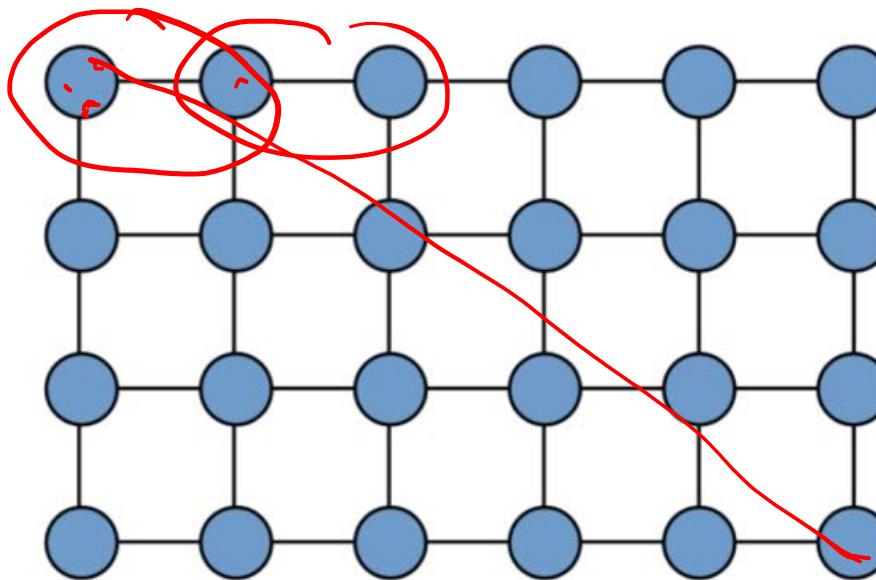
(a) Image



(b) Unary classifiers



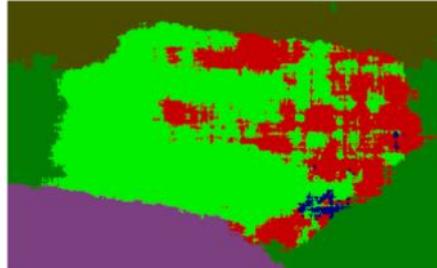
(c) Robust P^n CRF



Fully Connected CRF



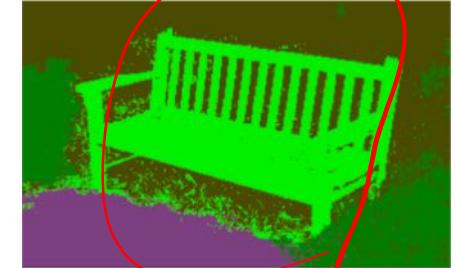
(a) Image



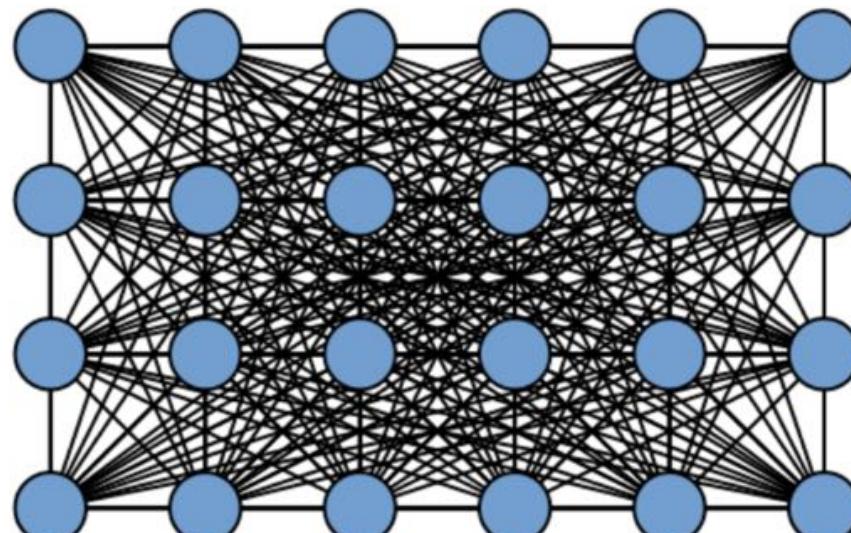
(b) Unary classifiers



(c) Robust P^n CRF



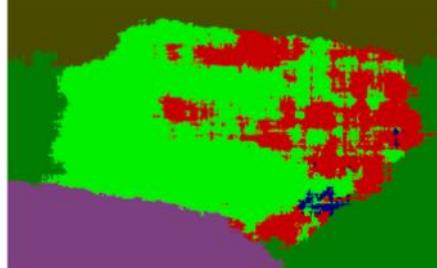
(d) Fully connected CRF,
MCMC inference, 36 hrs



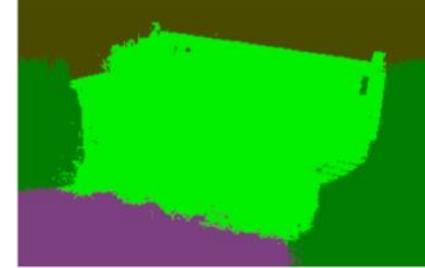
Fully Connected CRF



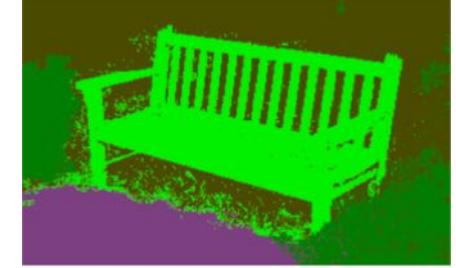
(a) Image



(b) Unary classifiers

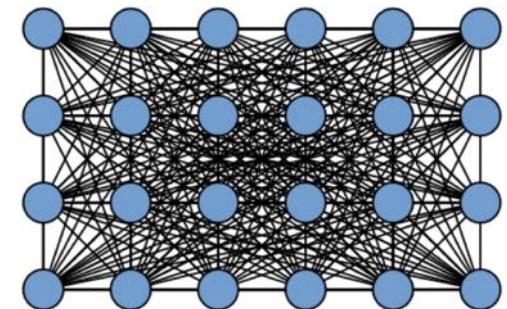


(c) Robust P^n CRF



(d) Fully connected CRF,
MCMC inference, 36 hrs

- Model long-range connections
- Inference is computationally hard



Fully Connected CRF

- Label $\mathbf{X} = \{X_1, \dots, X_N\}$

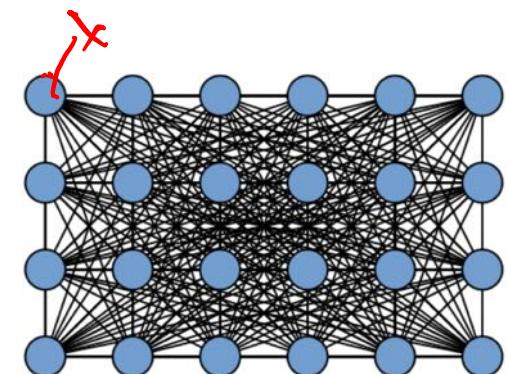
- Image $\mathbf{I} = \{I_1, \dots, I_N\}$

- Gibbs distribution

$$P(\mathbf{X}|\mathbf{I}) = \frac{1}{Z(\mathbf{I})} \exp \left(- \sum_{c \in \mathcal{C}_G} \phi_c(\mathbf{x}_c | \mathbf{I}) \right)$$

- Energy function

$$E(\mathbf{x}|\mathbf{I}) = \sum_{c \in \mathcal{C}_G} \phi_c(\mathbf{x}_c | \mathbf{I})$$



Fully Connected CRF

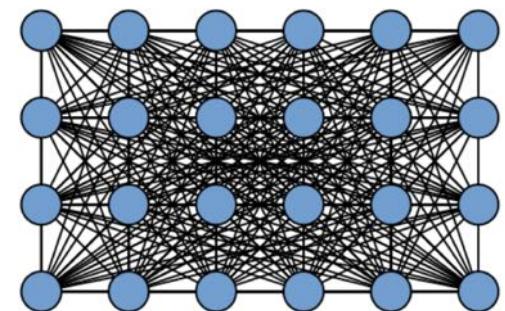
- Label $\mathbf{X} = \{X_1, \dots, X_N\}$
- Image $\mathbf{I} = \{I_1, \dots, I_N\}$
- Gibbs distribution

$$P(\mathbf{X}|\mathbf{I}) = \frac{1}{Z(\mathbf{I})} \exp\left(- \sum_{c \in \mathcal{C}_G} \phi_c(\mathbf{X}_c | \mathbf{I})\right)$$

- Energy function

$$E(\mathbf{x}) = \sum_i \psi_u(x_i) + \sum_{i < j} \psi_p(x_i, x_j)$$

Unary Pairwise



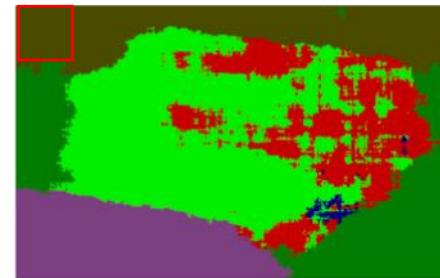
Unary potential

$$E(\mathbf{x}) = \sum_i \psi_u(x_i) + \sum_{i < j} \psi_p(x_i, x_j)$$

- Unary $\psi_u(x_i)$
 - Computed independently for each pixel
 - Multinomial logistic regression, boosting



(a) Image



(b) Unary classifiers

Pairwise Potential

$$E(\mathbf{x}) = \sum_i \psi_u(x_i) + \boxed{\sum_{i < j} \psi_p(x_i, x_j)}$$

- Pairwise potential

$$\psi_p(x_i, x_j) = \mu(x_i, x_j) \sum_{m=1}^K w^{(m)} k^{(m)}(f_i, f_j)$$

- μ - label compatibility function
- k – linear combination of Gaussian kernels

$$k^{(m)}(f_i, f_j) = \exp\left(-\frac{1}{2}(f_i - f_j)\Sigma^{(m)}(f_i - f_j)\right)$$

Detailed model

$$\psi_p(x_i, x_j) = \mu(x_i, x_j) \sum_{m=1}^K w^{(m)} k^{(m)}(\mathbf{f}_i, \mathbf{f}_j)$$



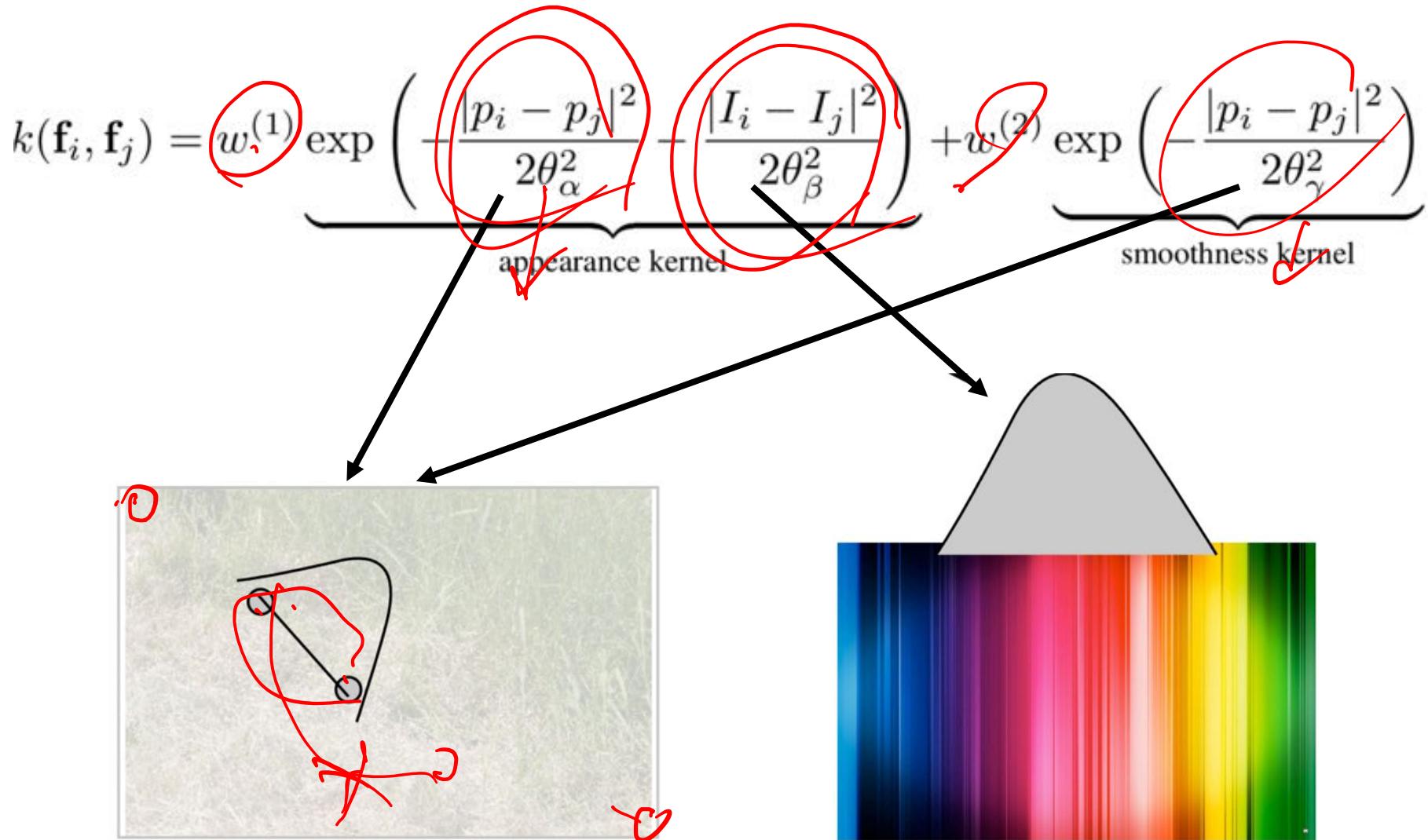
- μ - label compatibility function

Potts Model: $\mu(x_i, x_j) = [x_i \neq x_j]$

- k – Appearance and local smoothness

$$k(\mathbf{f}_i, \mathbf{f}_j) = \underbrace{w^{(1)} \exp \left(-\frac{|p_i - p_j|^2}{2\theta_\alpha^2} - \frac{|I_i - I_j|^2}{2\theta_\beta^2} \right)}_{\text{appearance kernel}} + w^{(2)} \underbrace{\exp \left(-\frac{|p_i - p_j|^2}{2\theta_\gamma^2} \right)}_{\text{smoothness kernel}}$$

Detailed model



Message Passing

- Mean field approximation

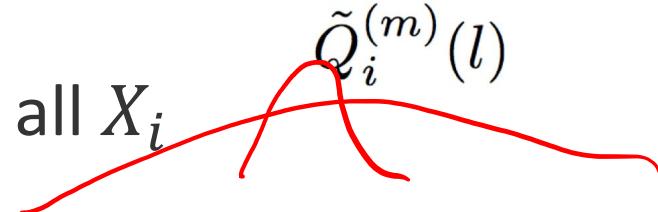
- Minimize KL divergence $D(Q||P)$

$$\text{s.t. } Q(X) = \prod_i Q_i(X_i)$$

- Iterative update equation

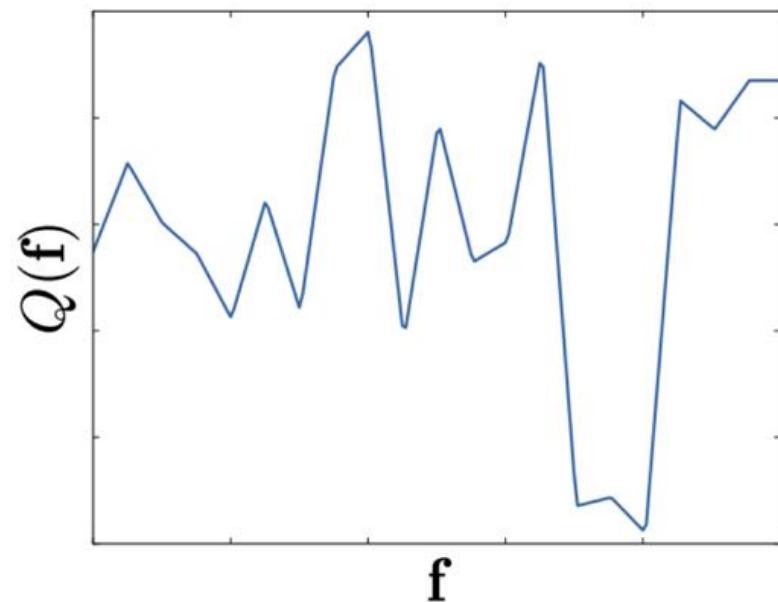
$$Q_i(x_i = l) = \frac{1}{Z_i} \exp \left\{ -\psi_u(x_i) - \sum_{l' \in \mathcal{L}} \mu(l, l') \sum_{m=1}^K w^{(m)} \sum_{j \neq i} k^{(m)}(\mathbf{f}_i, \mathbf{f}_j) Q_j(l') \right\}$$

- Message passing from all X_j to all X_i



High-dimensional Filtering

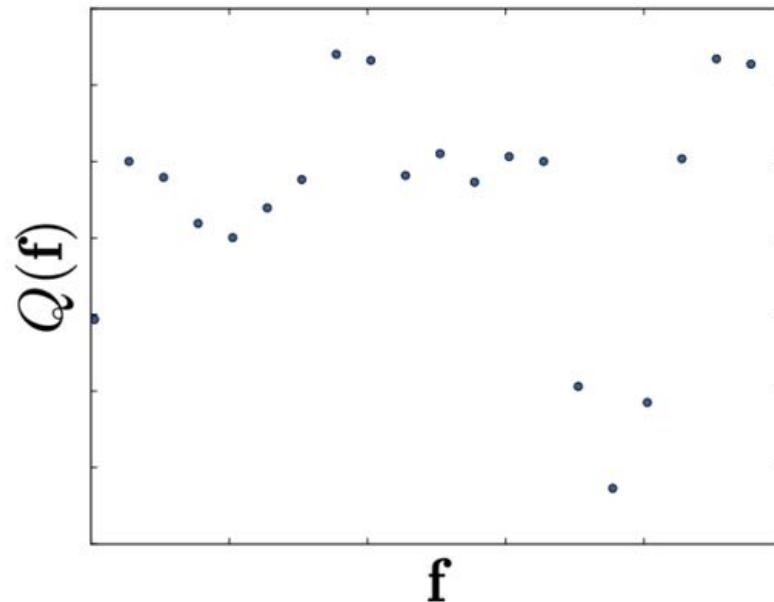
$$\tilde{Q}_i^{(m)}(l) = \underbrace{\sum_{j \in \mathcal{V}} k^{(m)}(\mathbf{f}_i, \mathbf{f}_j) Q_j(l) - Q_i(l)}_{\text{message passing}} = \underbrace{[G_{\Lambda^{(m)}} \otimes Q(l)](\mathbf{f}_i)}_{\overline{Q}_i^{(m)}(l)} - Q_i(l).$$



High-dimensional Filtering

$$\tilde{Q}_i^{(m)}(l) = \underbrace{\sum_{j \in \mathcal{V}} k^{(m)}(\mathbf{f}_i, \mathbf{f}_j) Q_j(l) - Q_i(l)}_{\text{message passing}} = \underbrace{[G_{\Lambda^{(m)}} \otimes Q(l)](\mathbf{f}_i)}_{\overline{Q}_i^{(m)}(l)} - Q_i(l).$$

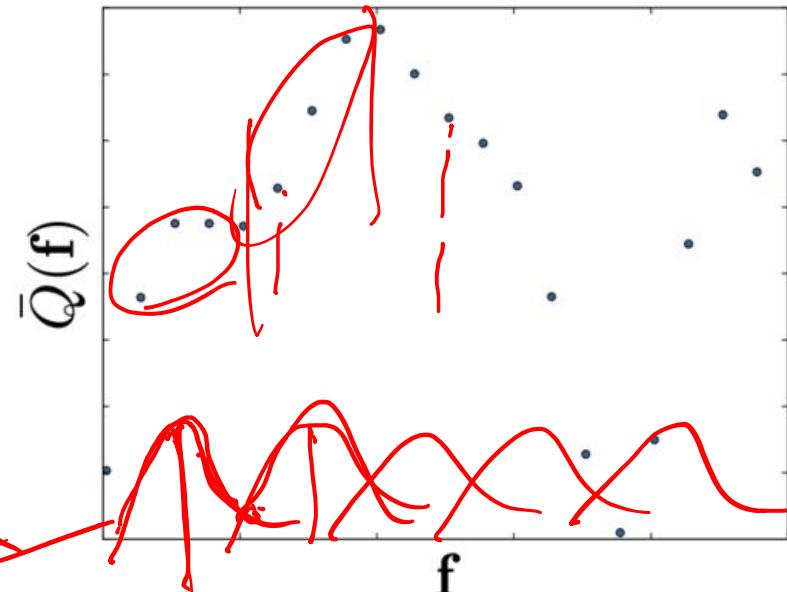
- Downsample $Q(l)$



High-dimensional Filtering

$$\tilde{Q}_i^{(m)}(l) = \underbrace{\sum_{j \in \mathcal{V}} k^{(m)}(\mathbf{f}_i, \mathbf{f}_j) Q_j(l) - Q_i(l)}_{\text{message passing}} = \underbrace{[G_{\Lambda^{(m)}} \otimes Q(l)](\mathbf{f}_i) - Q_i(l)}_{\overline{Q}_i^{(m)}(l)}.$$

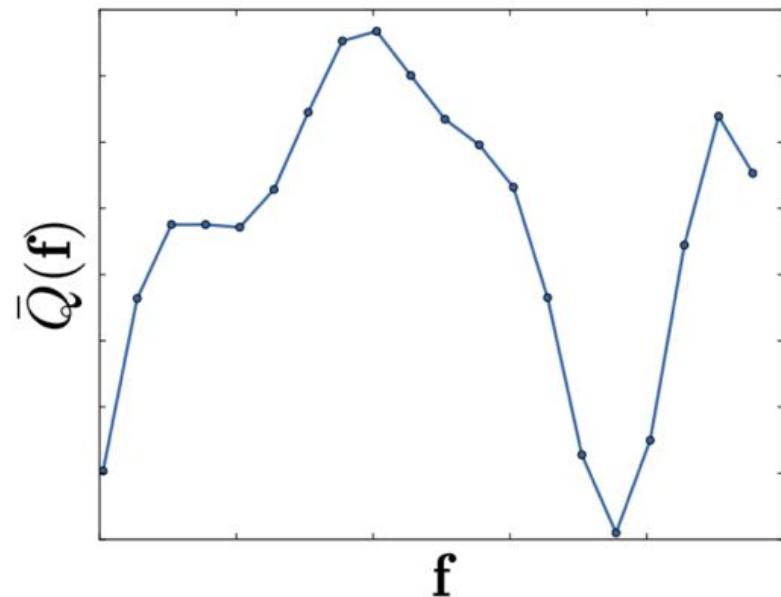
- Downsample $Q(l)$
- Blur the sampled signal



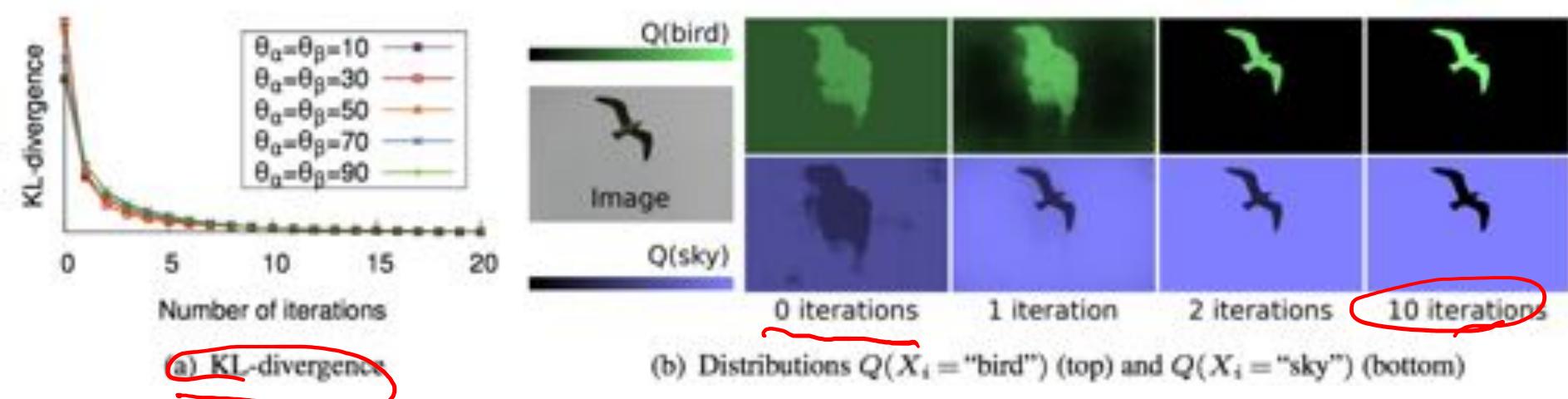
High-dimensional Filtering

$$\tilde{Q}_i^{(m)}(l) = \underbrace{\sum_{j \in \mathcal{V}} k^{(m)}(\mathbf{f}_i, \mathbf{f}_j) Q_j(l) - Q_i(l)}_{\text{message passing}} = \underbrace{[G_{\Lambda^{(m)}} \otimes Q(l)](\mathbf{f}_i)}_{\bar{Q}_i^{(m)}(l)} - Q_i(l).$$

- Downsample $Q(l)$
- Blur the sampled signal
- Upsample to reconstruct
 $\bar{Q}^{(m)}(l)$

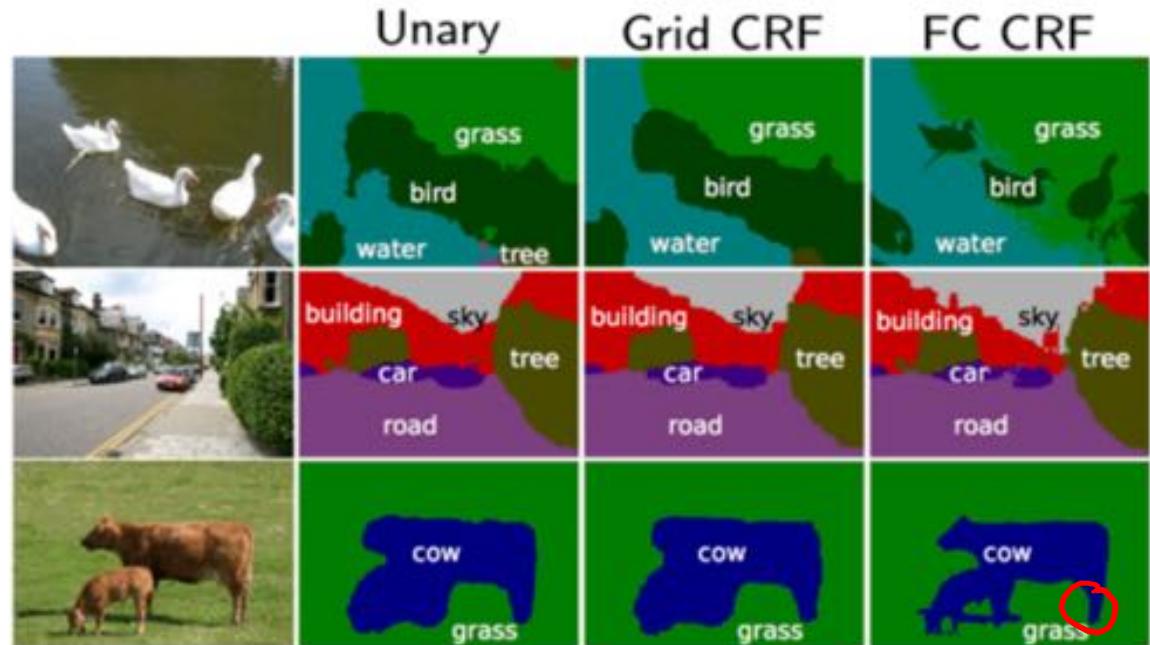


Convergence



Results: MSRC

- MSRC dataset
 - 591 images
 - 21 classes



	Time	Global	Avg
Unary	-	84.0	76.6
Grid CRF	1s	84.6	77.2
FC CRF	0.2s	86.0	78.3

Results: PASCAL VOC

- PASCAL dataset
 - 1928 images
 - 21 classes

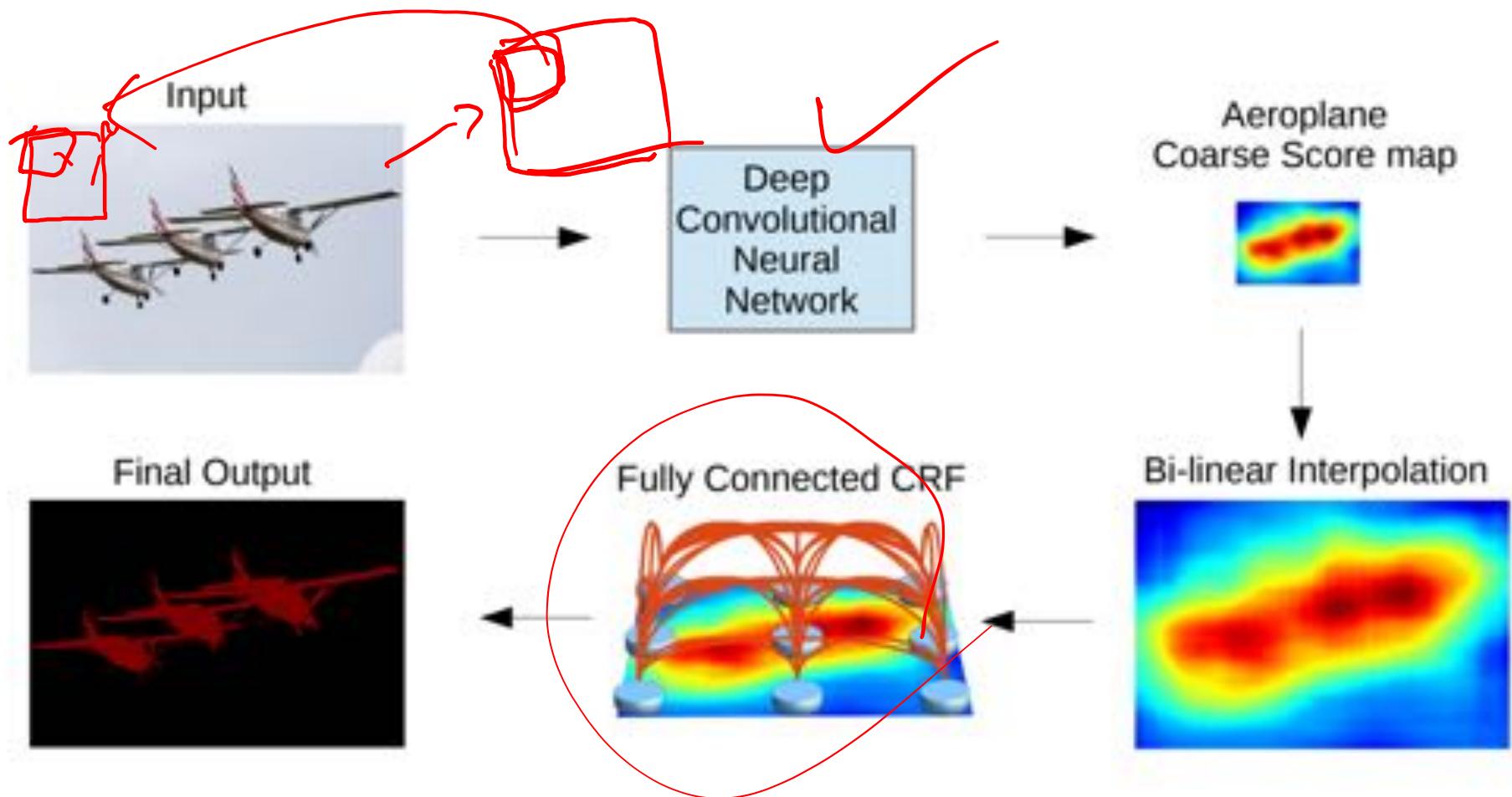
	Time	Acc
Unary	-	27.6
Grid CRF	2.5s	28.3
FC Potts	0.5s	29.1
FC label comp	0.5s	30.2



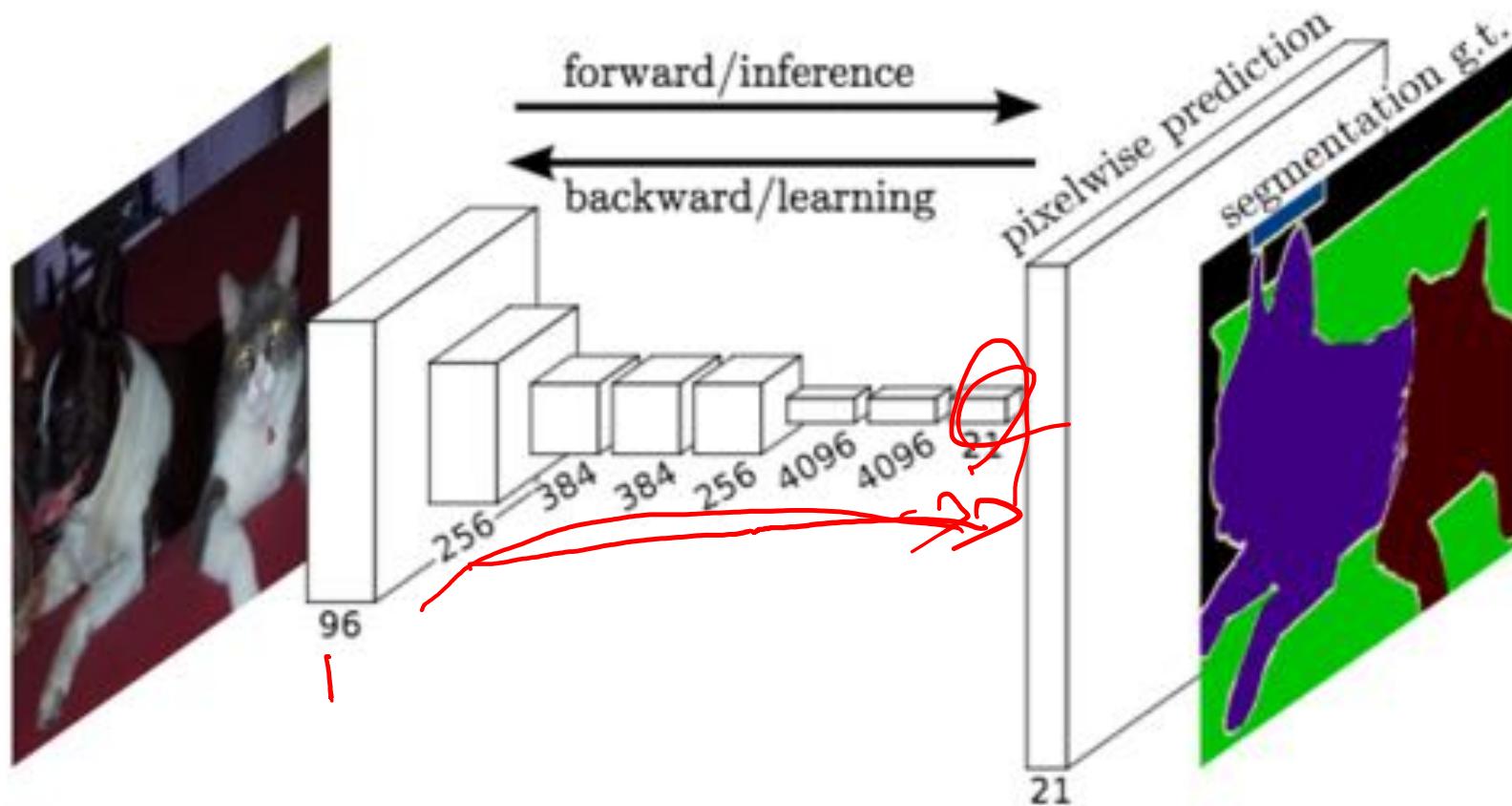
Convolutional Neural Nets

- CRF can refine the results from unary classifiers
- Limitation: the unary classifiers trained on handcrafted features
- Solution: Deep convolutional nets for dense feature extraction

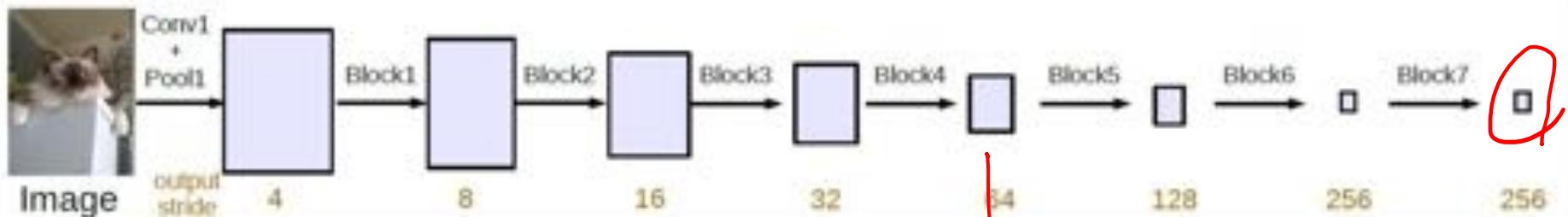
DeepLab-CRF



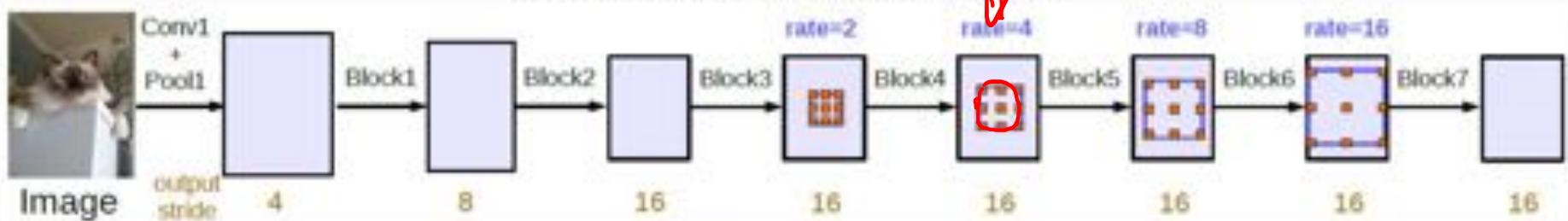
Fully Convolutional Nets



Atrous Convolution



(a) Going deeper without atrous convolution.



(b) Going deeper with atrous convolution. Atrous convolution with $rate > 1$ is applied after block3 when $output_stride = 16$.
Figure 3. Cascaded modules without and with atrous convolution.

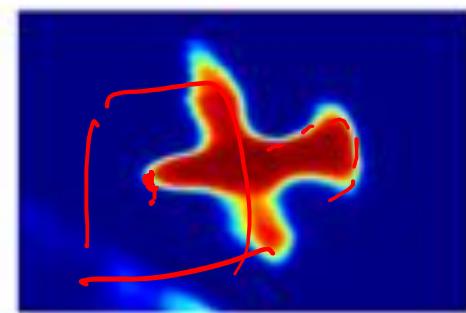
Results: Example



image



ground truth



DCNN output



CRF 1 iteration



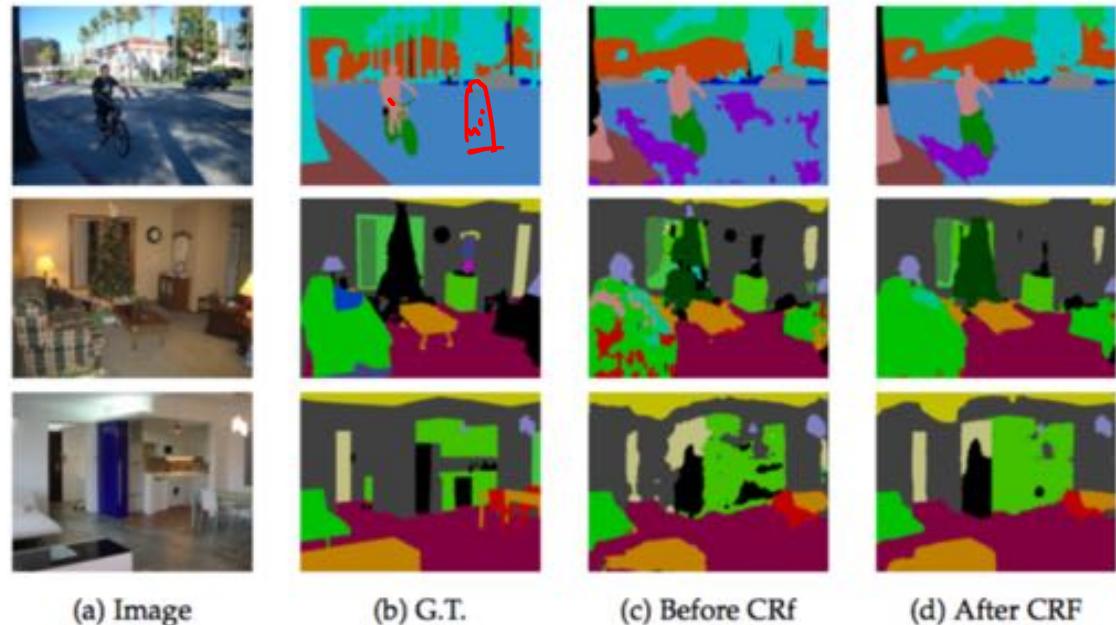
CRF 2 iteration



CRF 10 iteration

Results: PASCAL VOC

- PASCAL dataset
 - 10, 582 images
 - 21 classes



(a) Image

(b) G.T.

(c) Before CRF

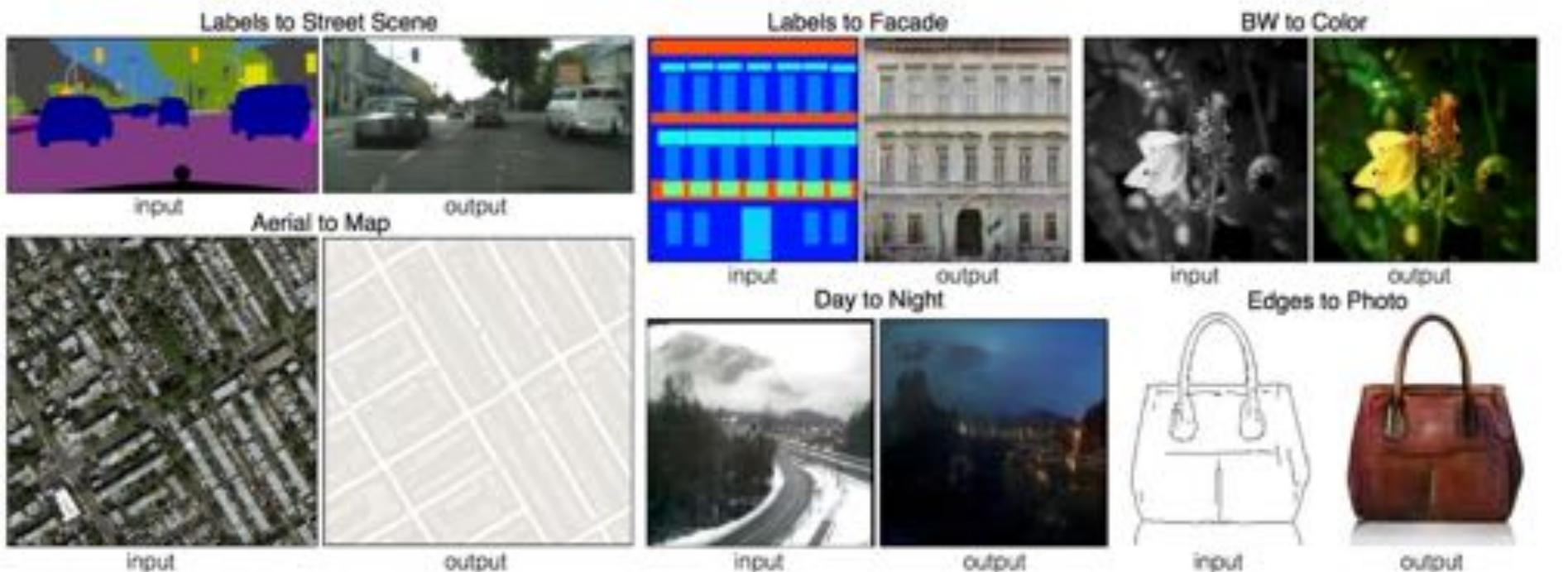
(d) After CRF

Method	before CRF	after CRF
LargeFOV	65.76	69.84
ASPP-S	66.98	69.73
ASPP-L	68.96	71.57

Outline

- Semantic Segmentation
- Image-to-Image Translation

Image-to-Image Translation

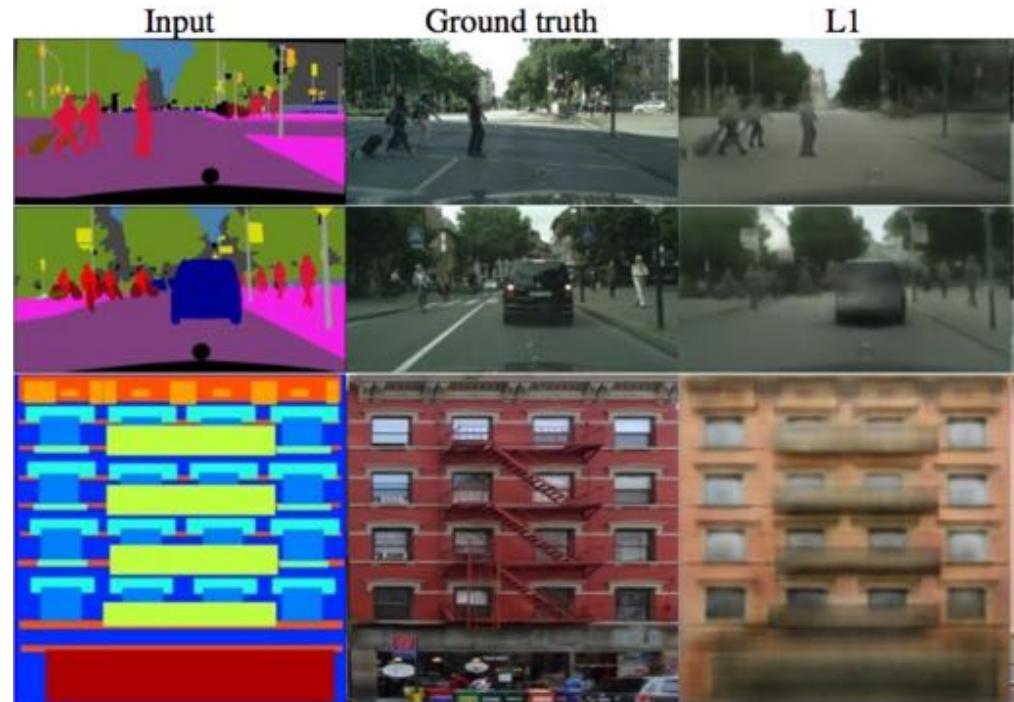


Supervised Learning

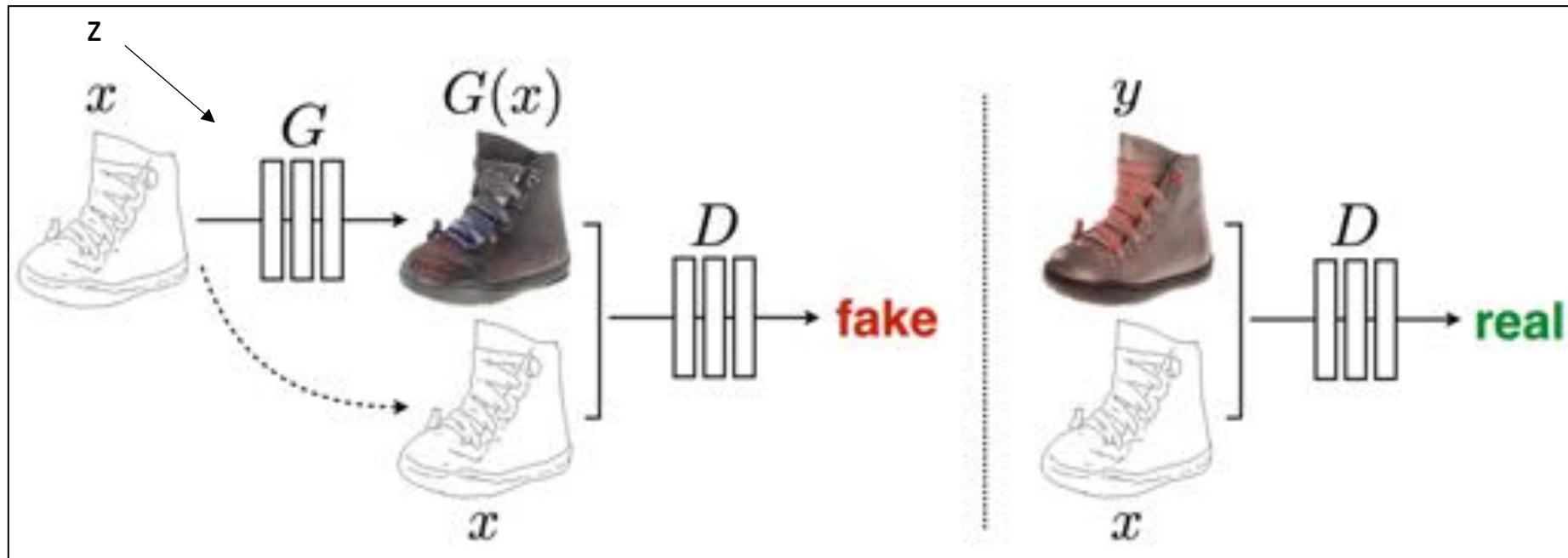


- Multi-output regression

- $\|G(X) - Y\|_1$



Conditional GAN

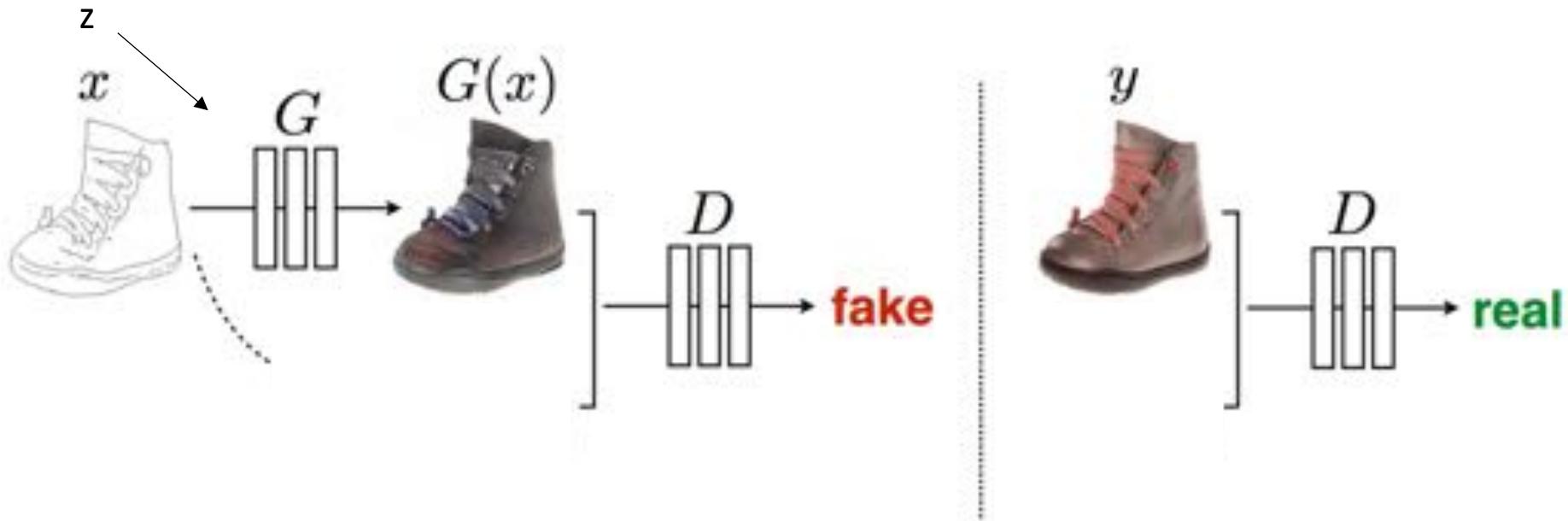


$$\begin{aligned}\mathcal{L}_{cGAN}(G, D) = & \mathbb{E}_{x,y}[\log D(x, y)] + \\ & \mathbb{E}_{x,z}[\log(1 - D(x, G(x, z)))]\end{aligned}$$

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y,z}[\|y - G(x, z)\|_1].$$

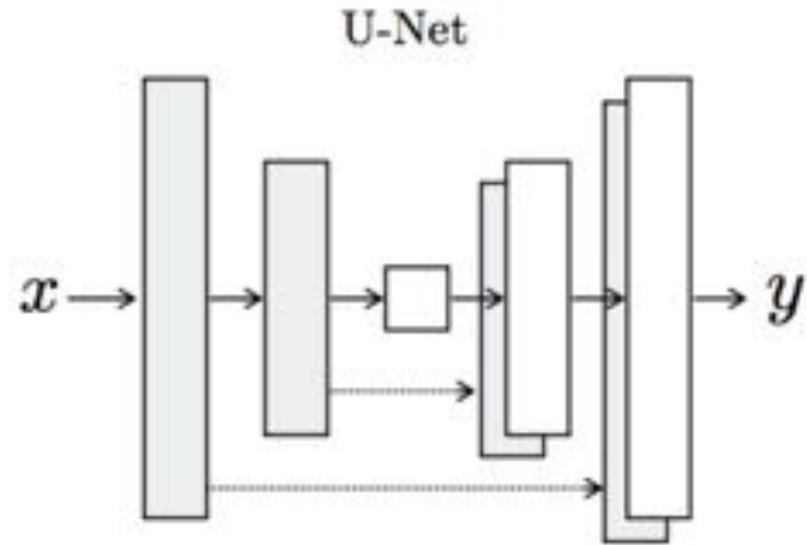
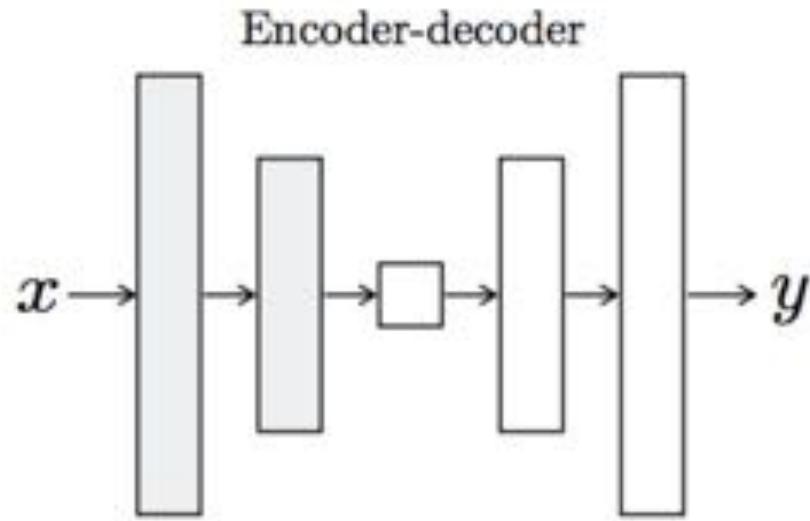
Isola, Phillip, et al. "Image-to-image translation with conditional adversarial networks." *arXiv preprint* (2017).

GAN

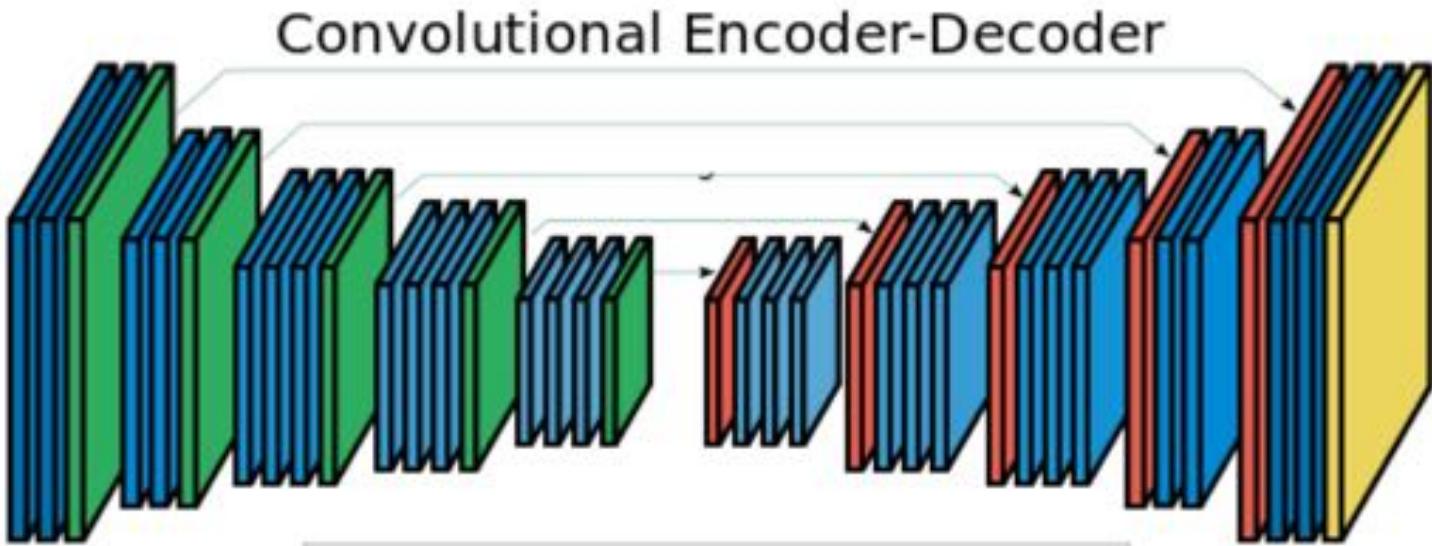


$$\begin{aligned}\mathcal{L}_{GAN}(G, D) = & \mathbb{E}_{x,y}[\log D(\cdot, y)] + \\ & \mathbb{E}_{x,z}[\log(1 - D(\cdot, G(x, z)))]\end{aligned}$$

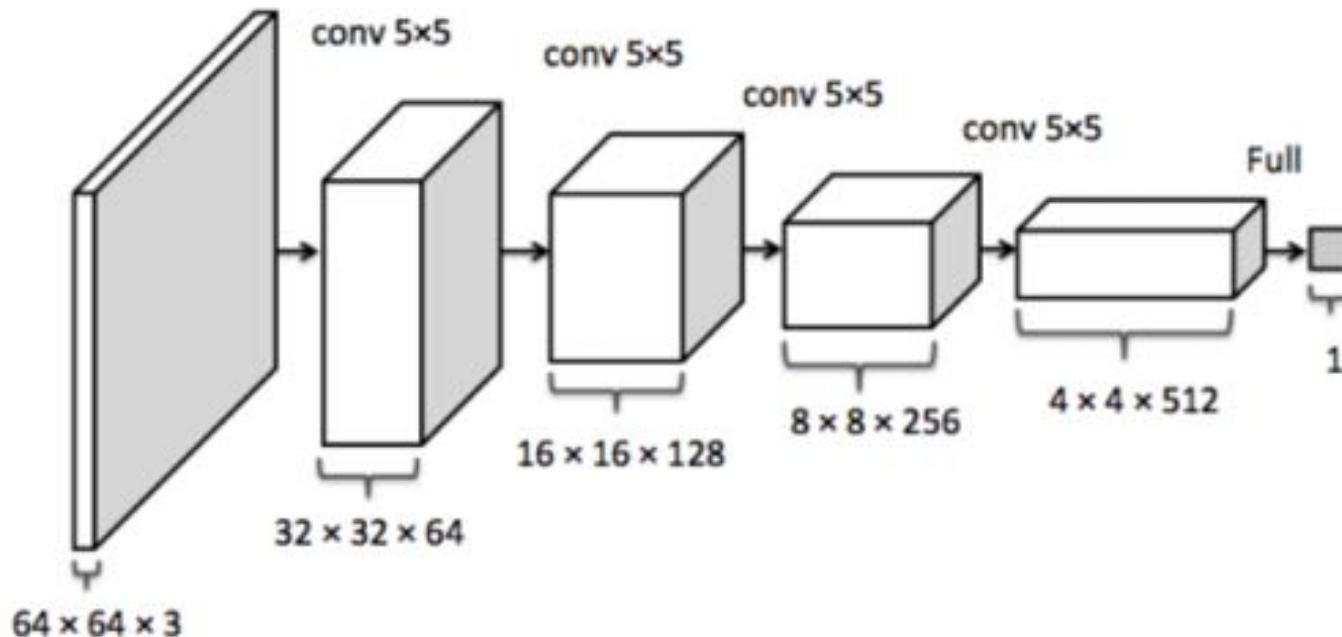
Generator (G)



Generator (G)



Discriminator (D)

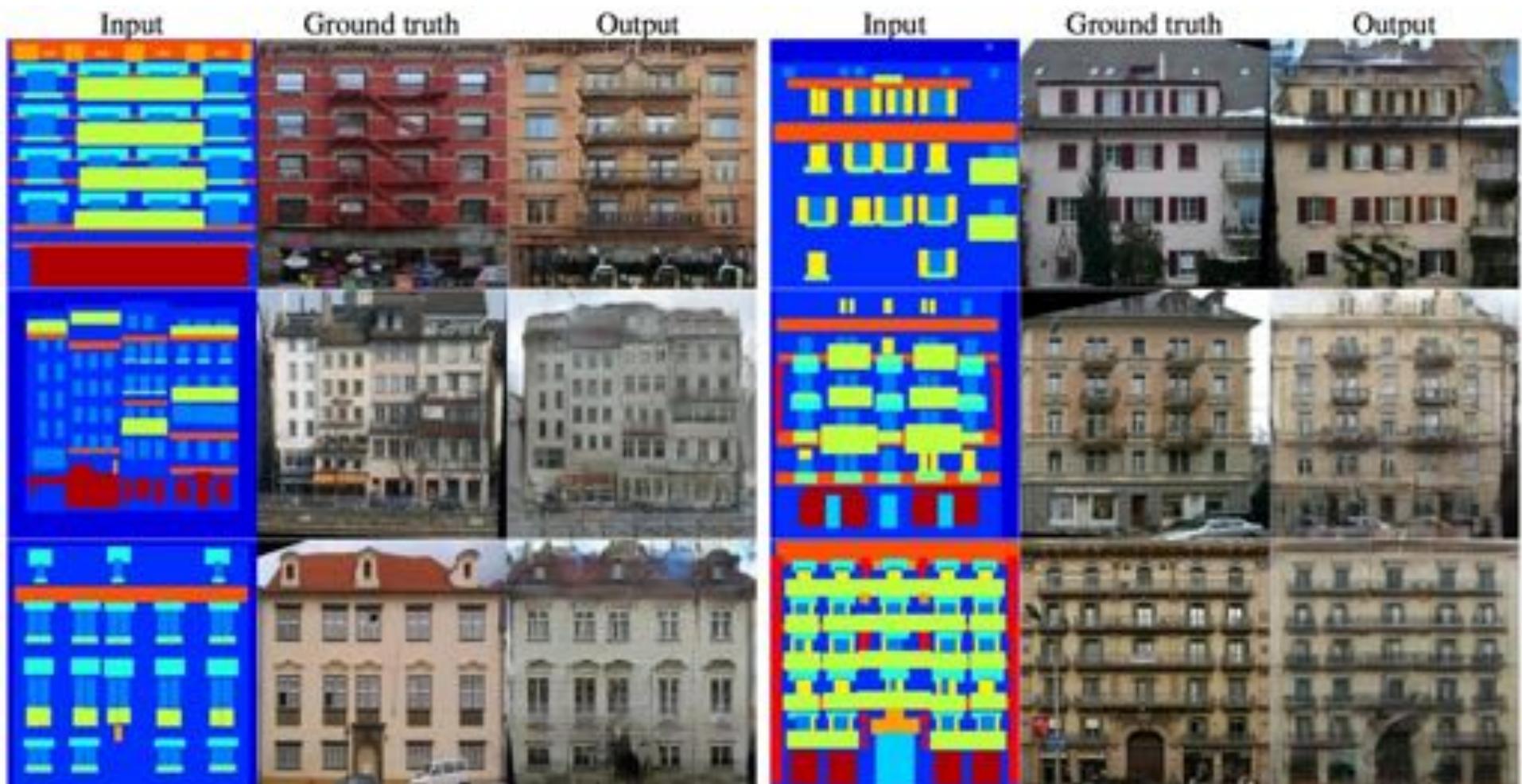


Radford, Alec, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks." *arXiv preprint arXiv:1511.06434* (2015).

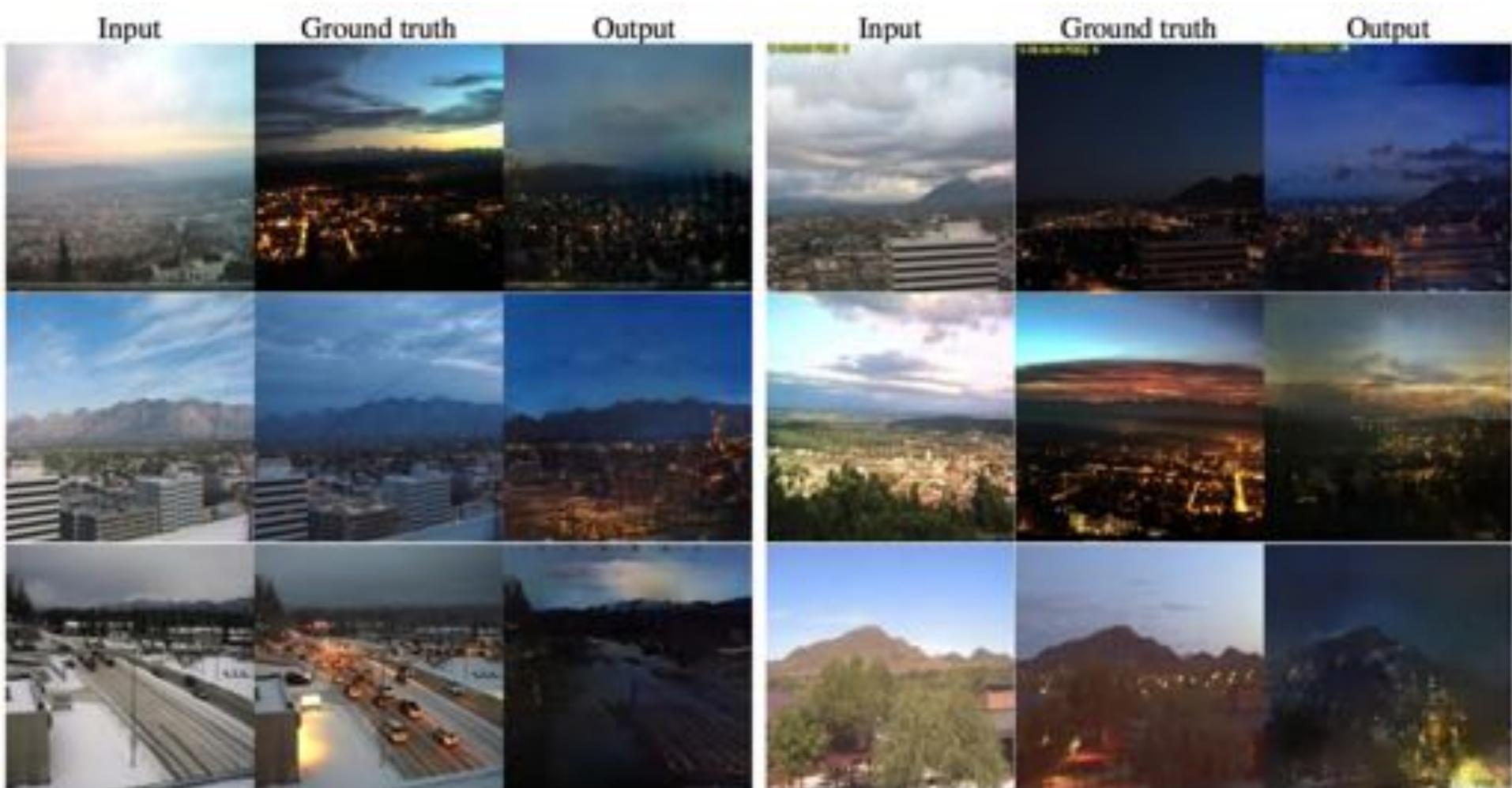
Edges->Images



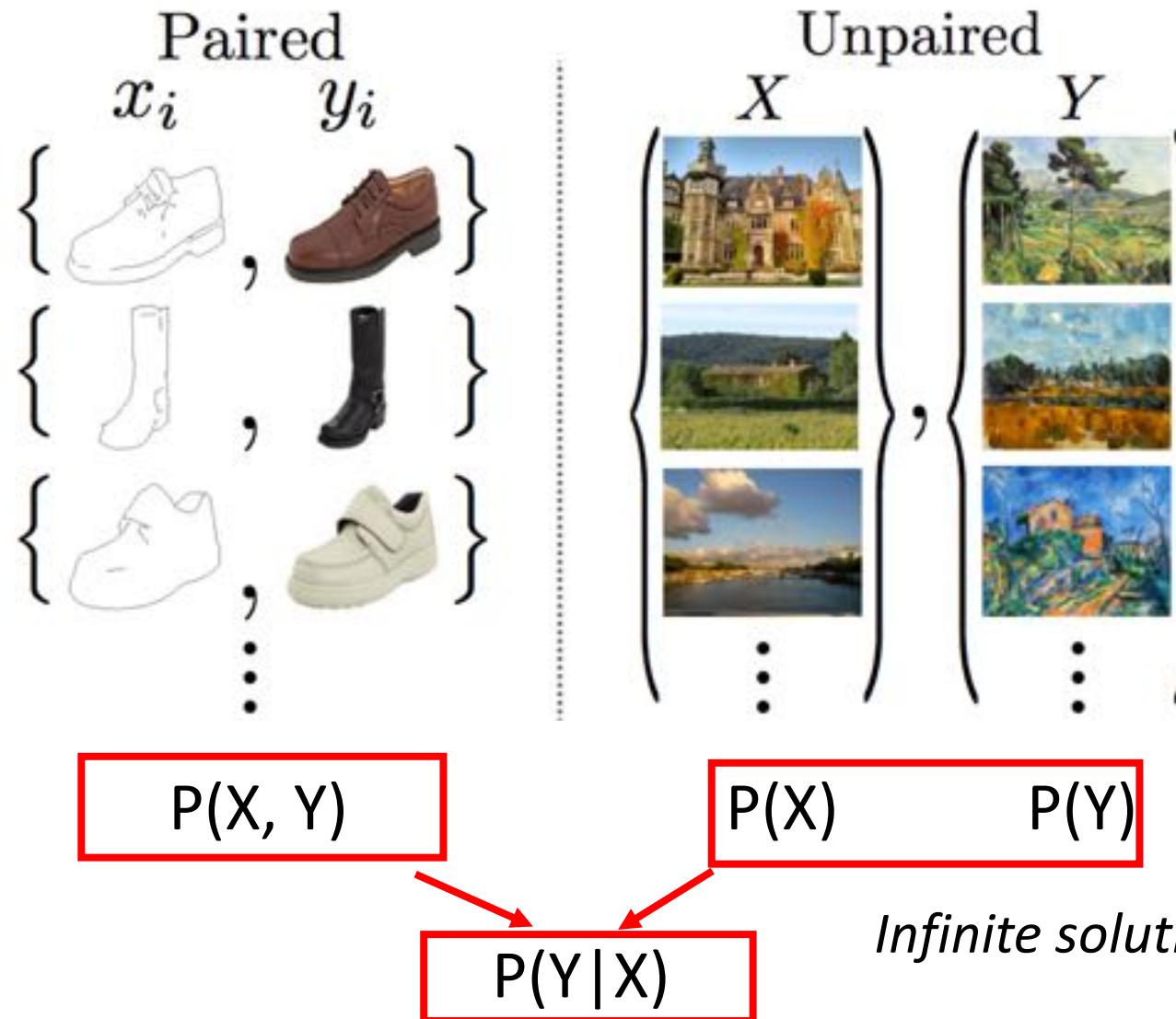
Labels -> Images



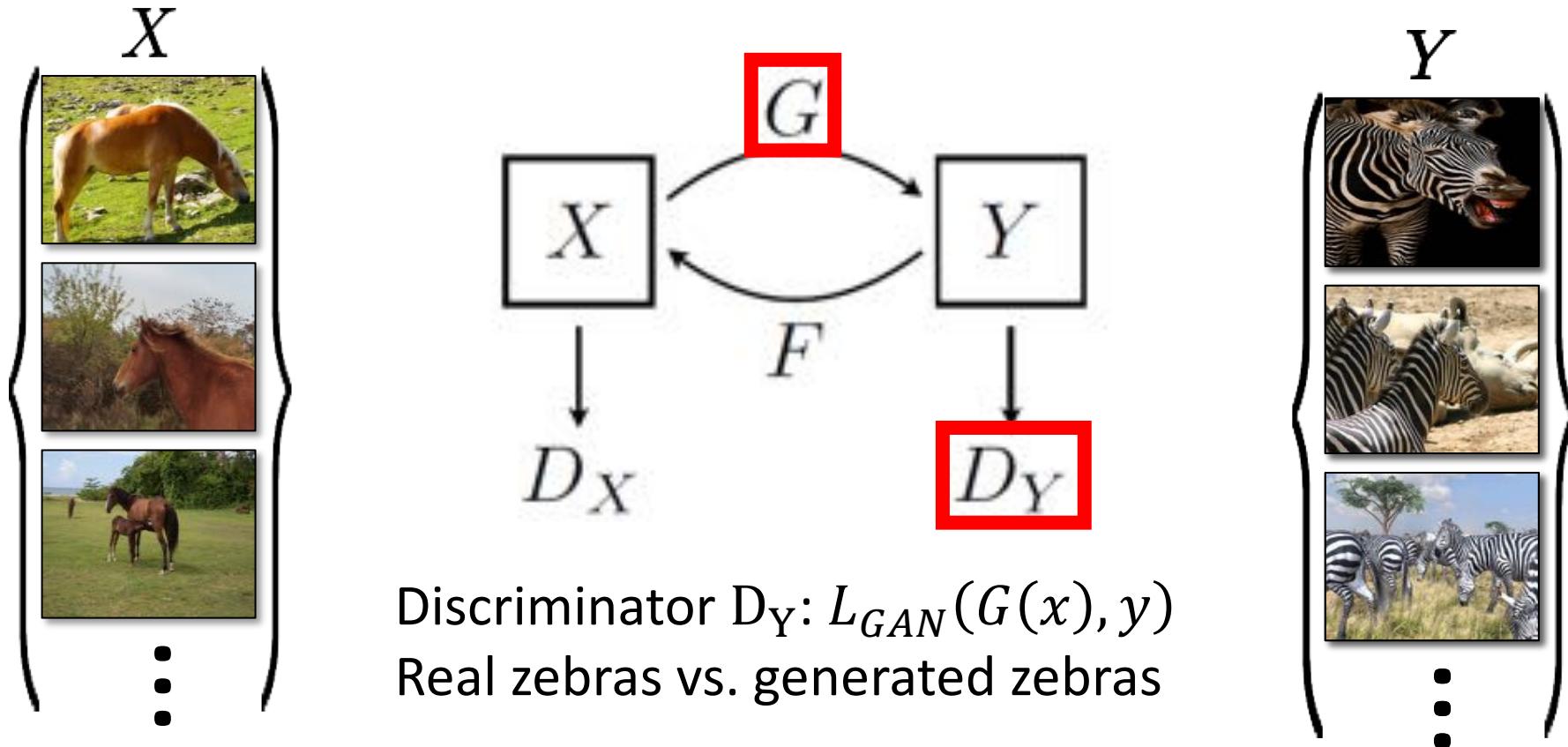
Day-> Night



Unpaired Data



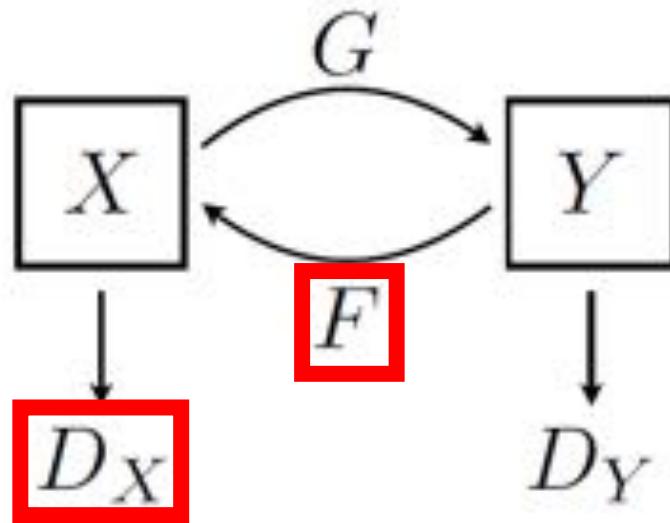
Cycle GAN



Zhu, Jun-Yan, et al. "Unpaired image-to-image translation using cycle-consistent adversarial networks." *arXiv preprint arXiv:1703.10593* (2017).

http://people.eecs.berkeley.edu/~junyanz/talks/pix2pix_cyclegan.pptx

Cycle GAN

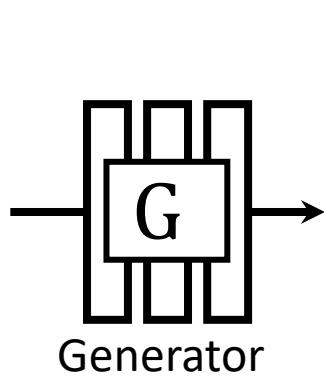


Discriminator D_Y : $L_{GAN}(G(x), y)$
Real zebras vs. generated zebras
Discriminator D_X : $L_{GAN}(F(y), x)$
Real horses vs. generated horses

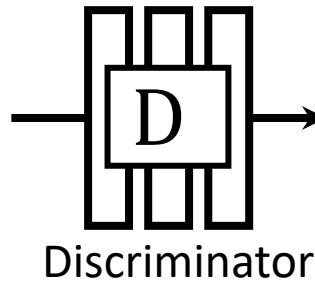


mode collapse!

x



$G(x)$

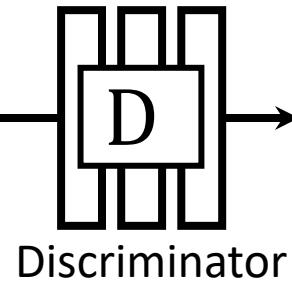
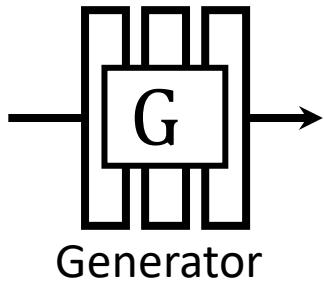


Real!

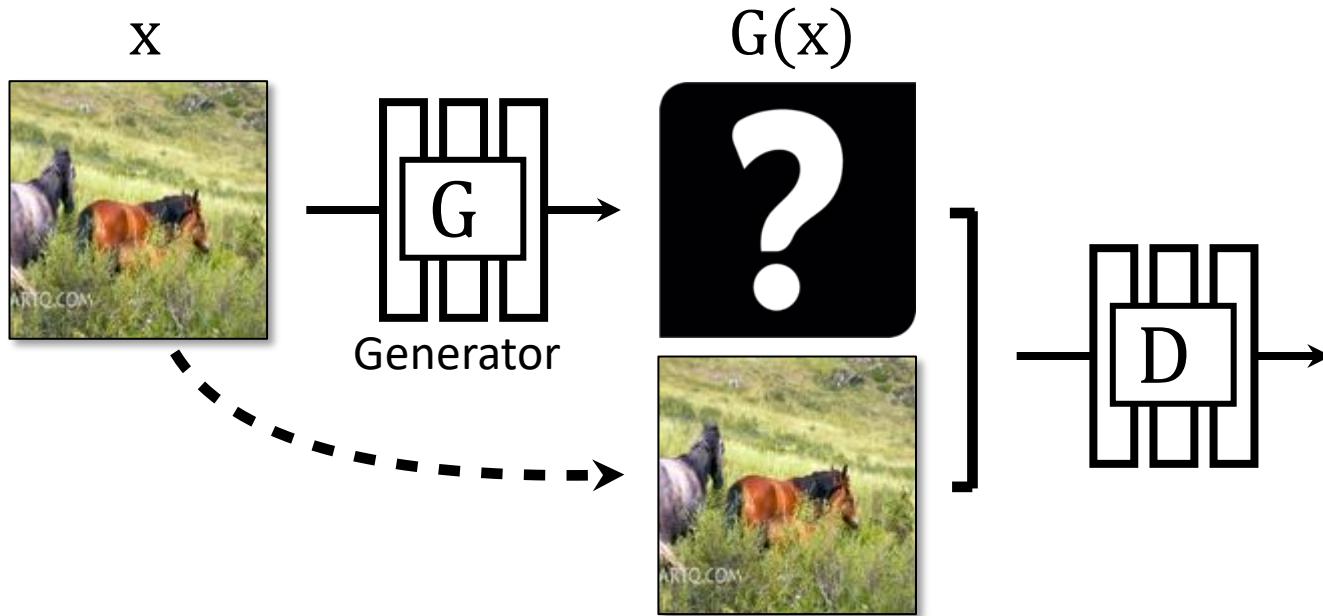
x



$G(x)$



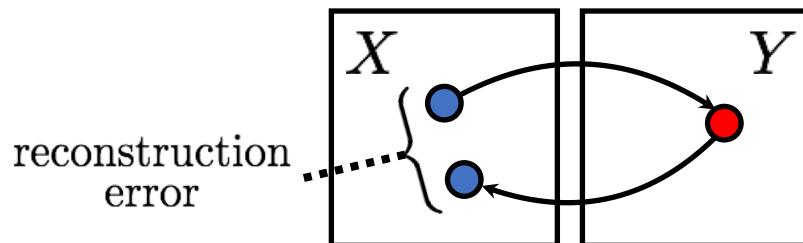
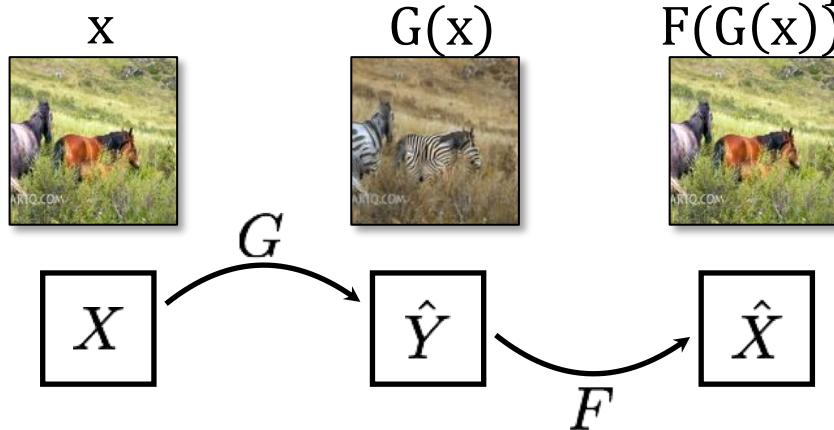
Real too!



No input-output pairs!

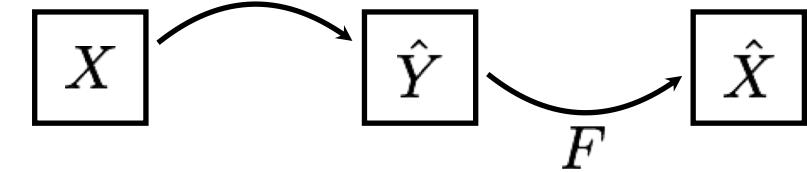
Cycle-consistency Loss

Forward cycle loss: $\|F(G(x)) - x\|_1$

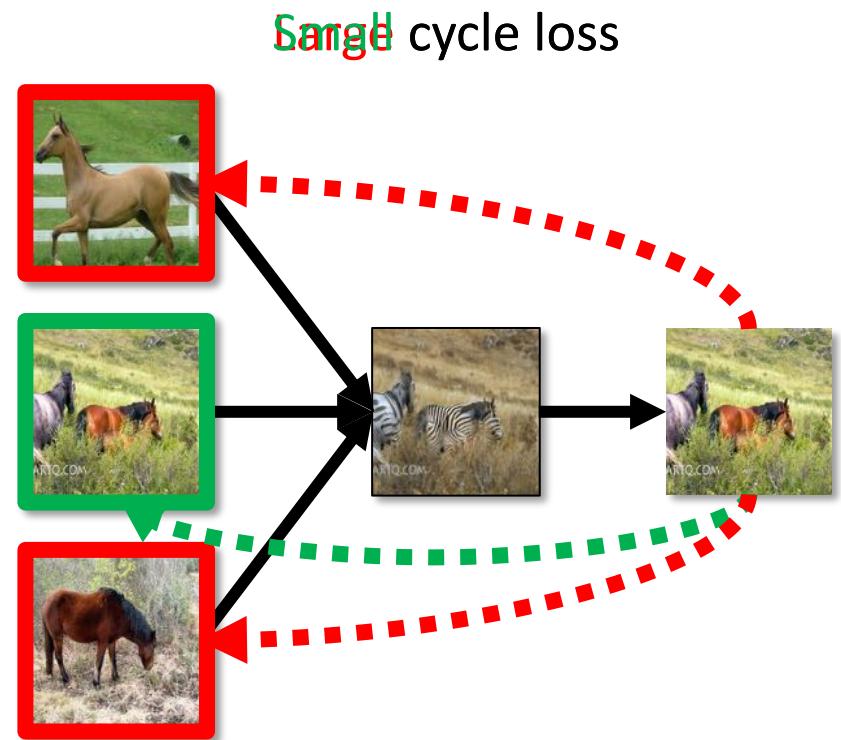


Cycle-consistency Loss

Forward cycle loss: $\|F(G(x)) - x\|_1$



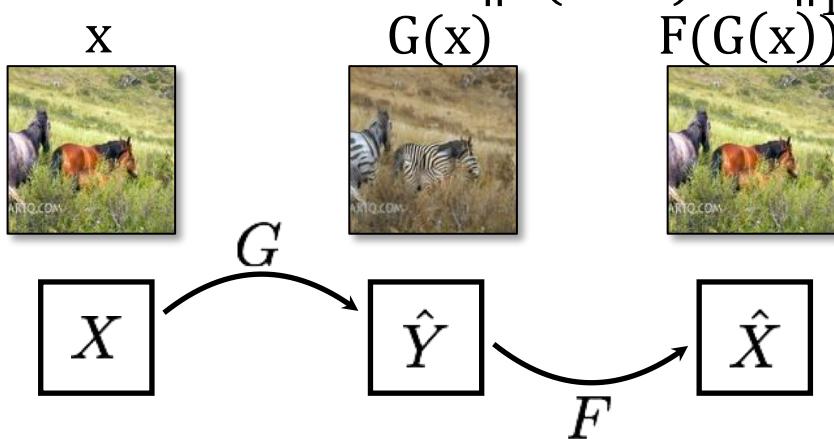
reconstruction
error



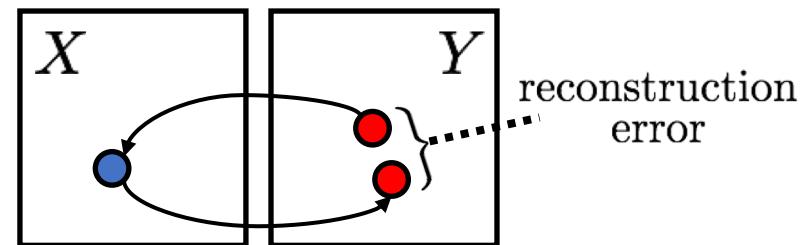
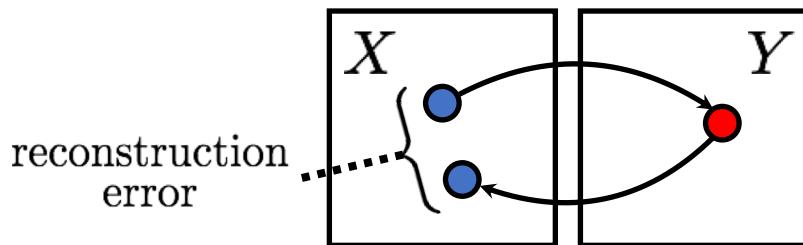
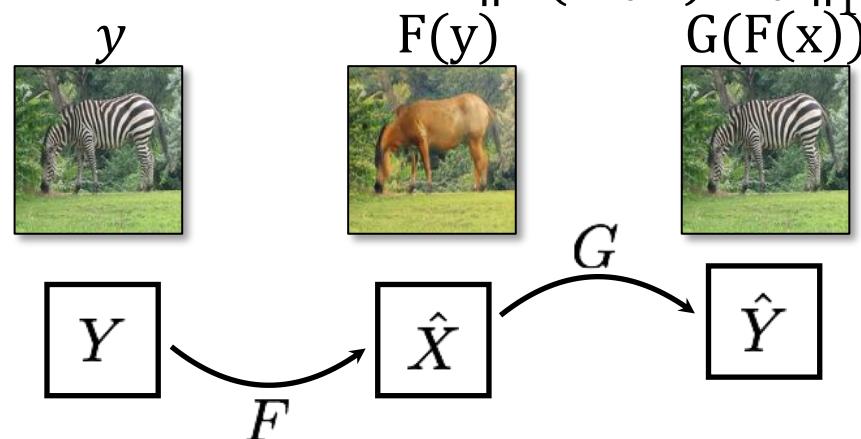
Sample cycle loss

Cycle-consistency Loss

Forward cycle loss: $\|F(G(x)) - x\|_1$



Backward cycle loss: $\|G(F(y)) - y\|_1$



Objective

GAN loss:

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

Cycle consistent loss:

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

Autoencoder

Input



Monet



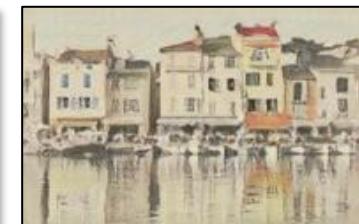
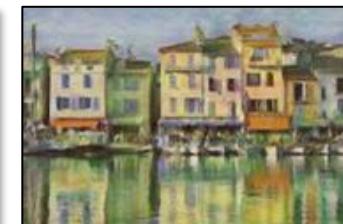
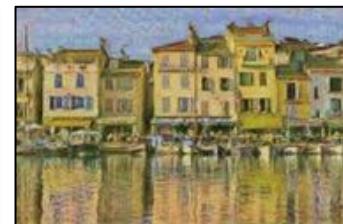
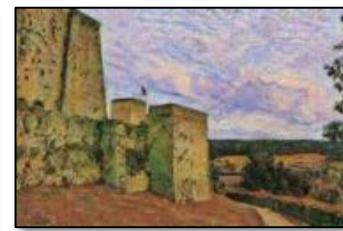
Van Gogh



Cezanne

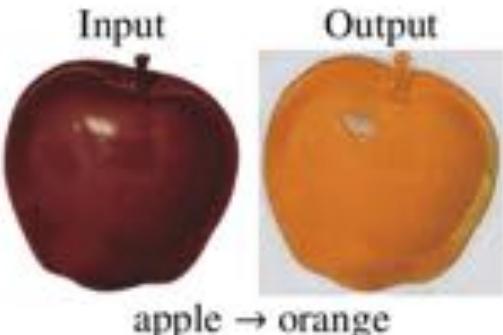


Ukiyo-e

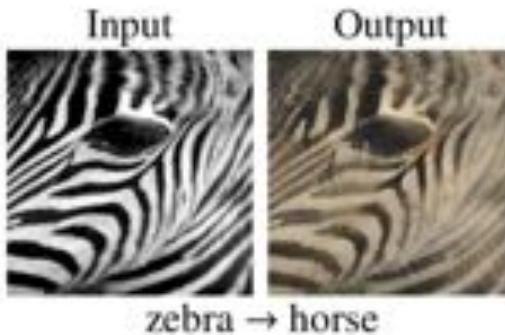




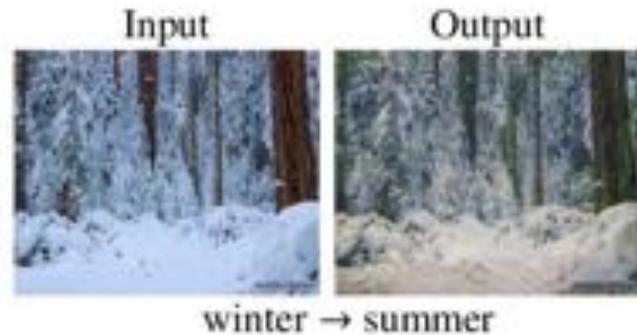
Failure cases



apple → orange



zebra → horse



winter → summer



dog → cat



cat → dog



Monet → photo



photo → Ukiyo-e



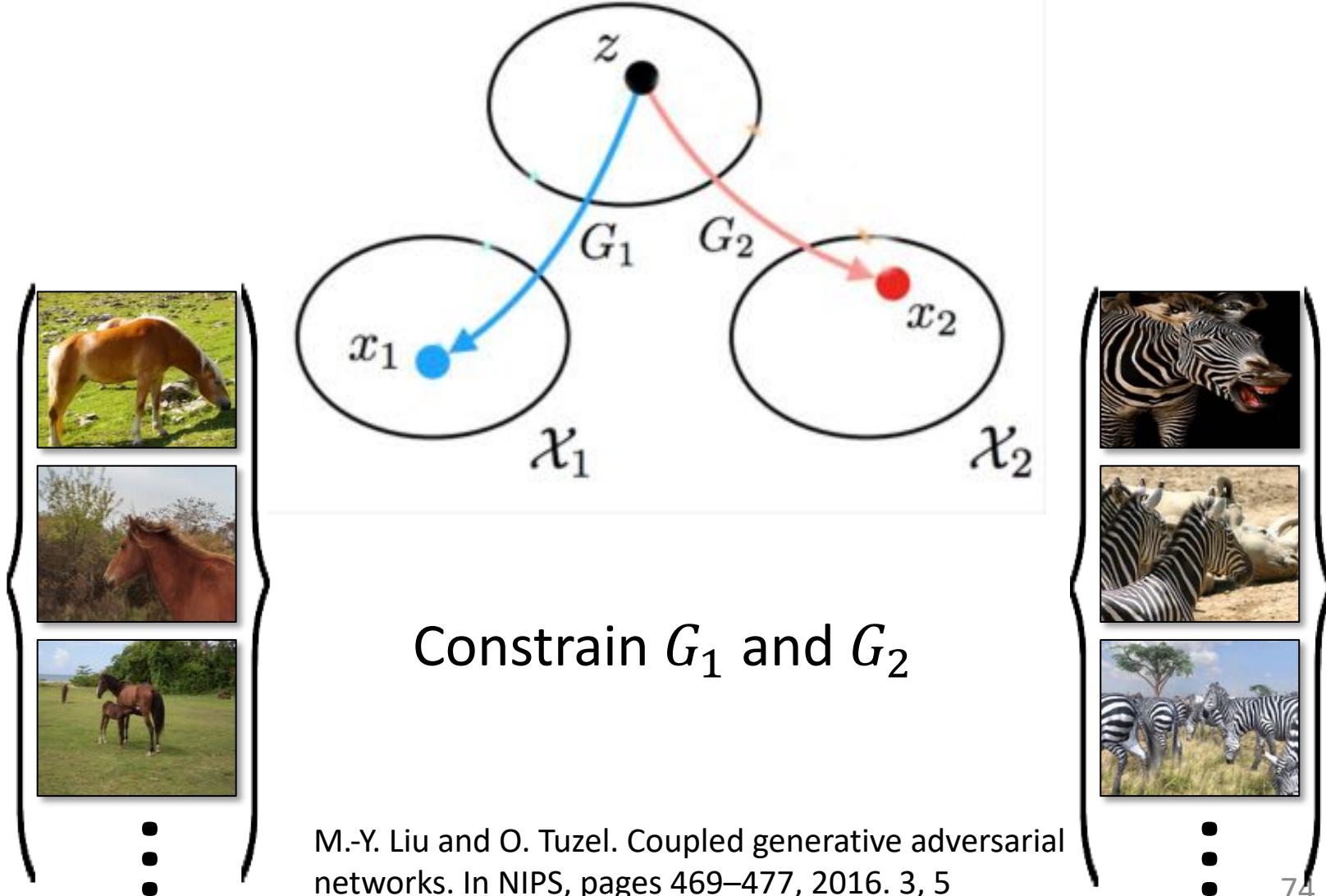
photo → Van Gogh



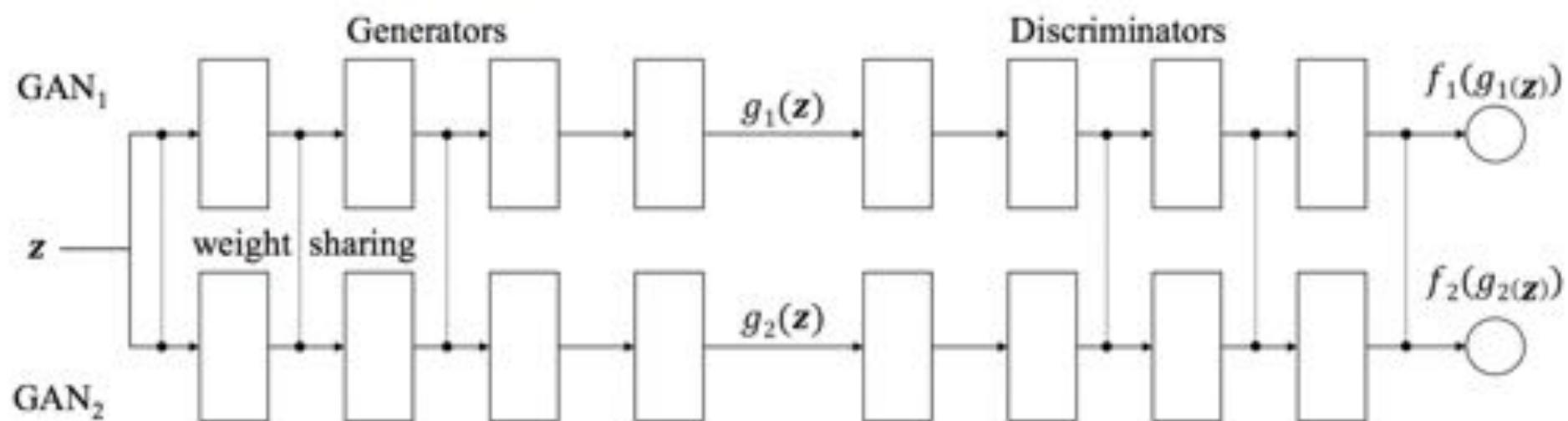
iPhone photo → DSLR photo

Coupled GAN

\mathcal{Z} : shared latent space

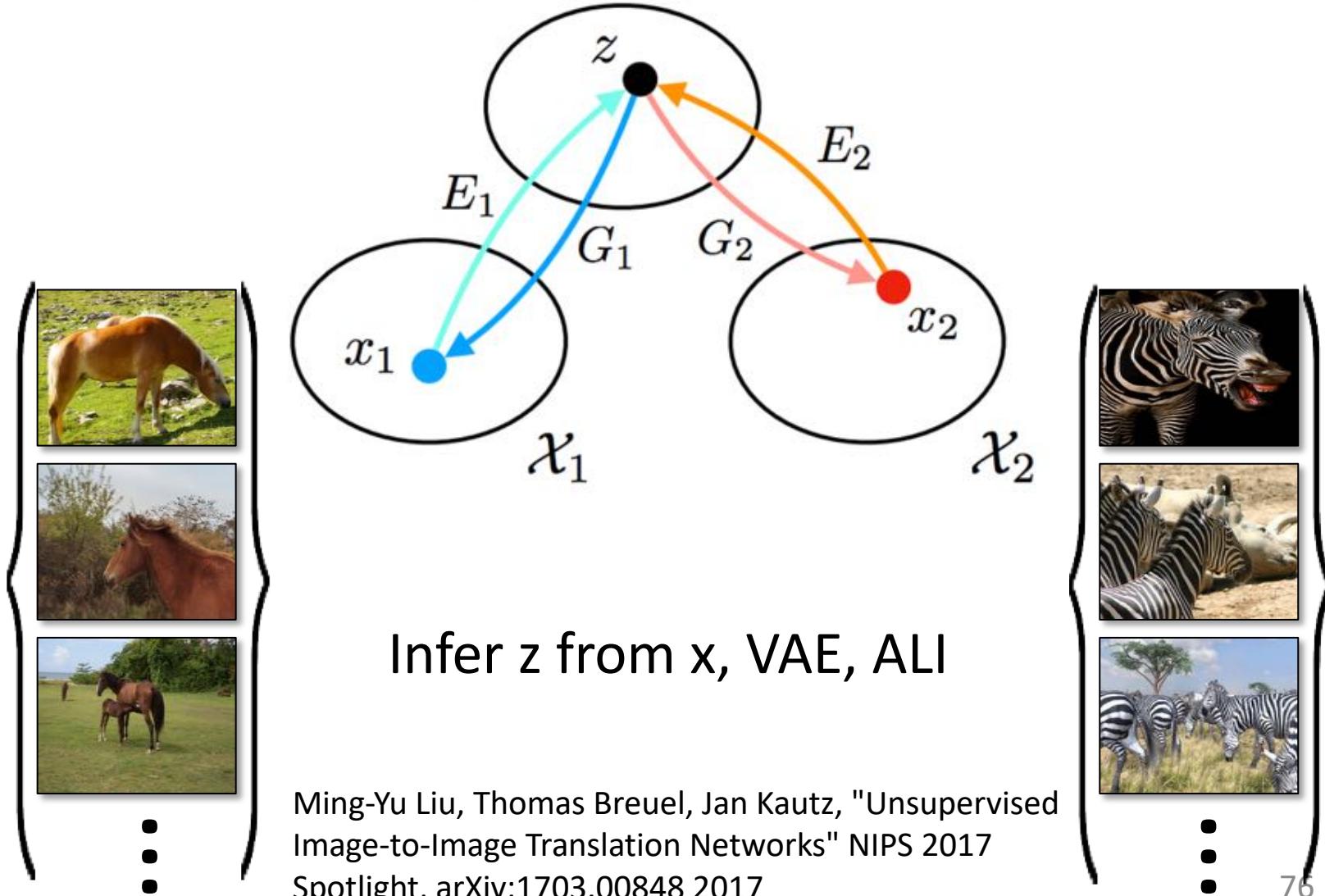


Layer Sharing

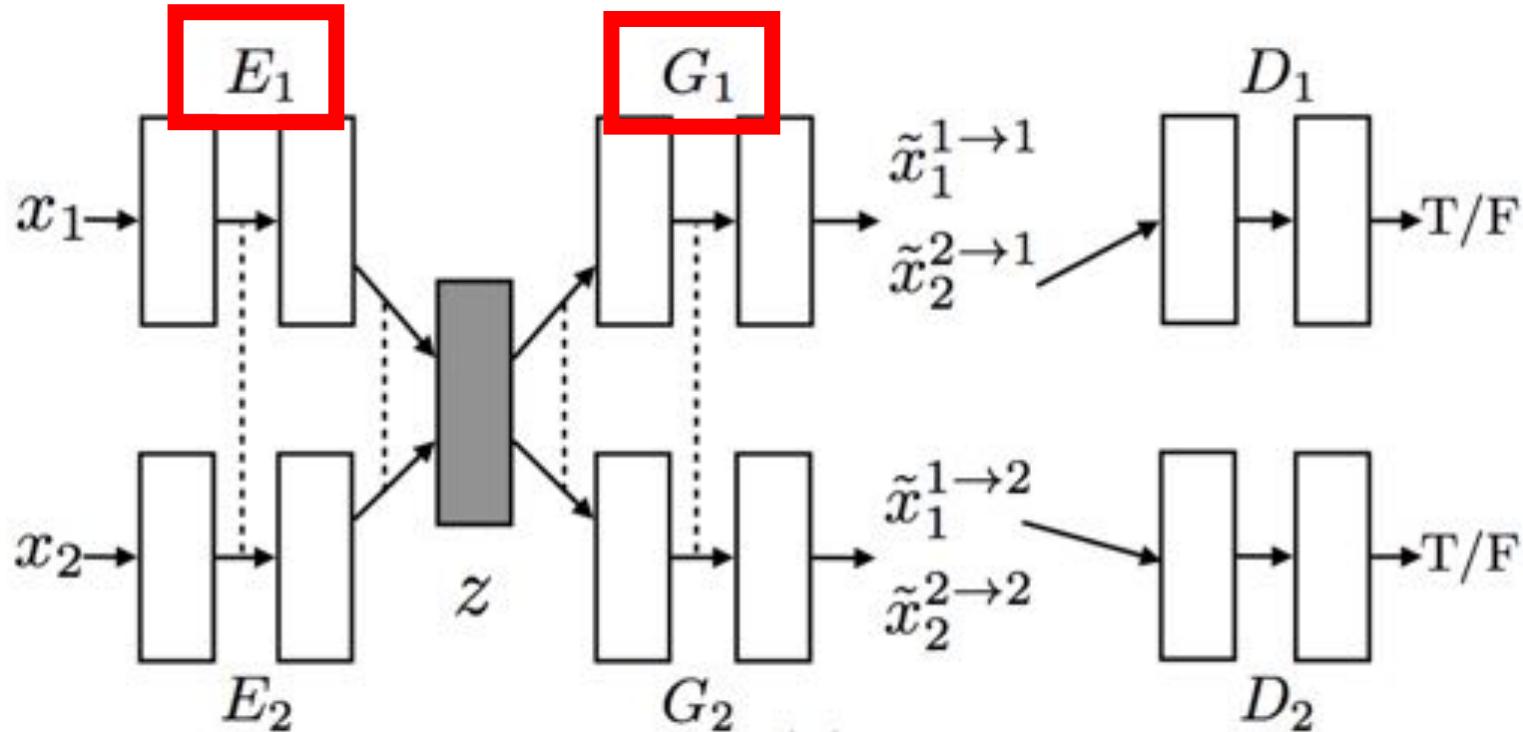


UNIT

\mathcal{Z} : shared latent space



UNIT

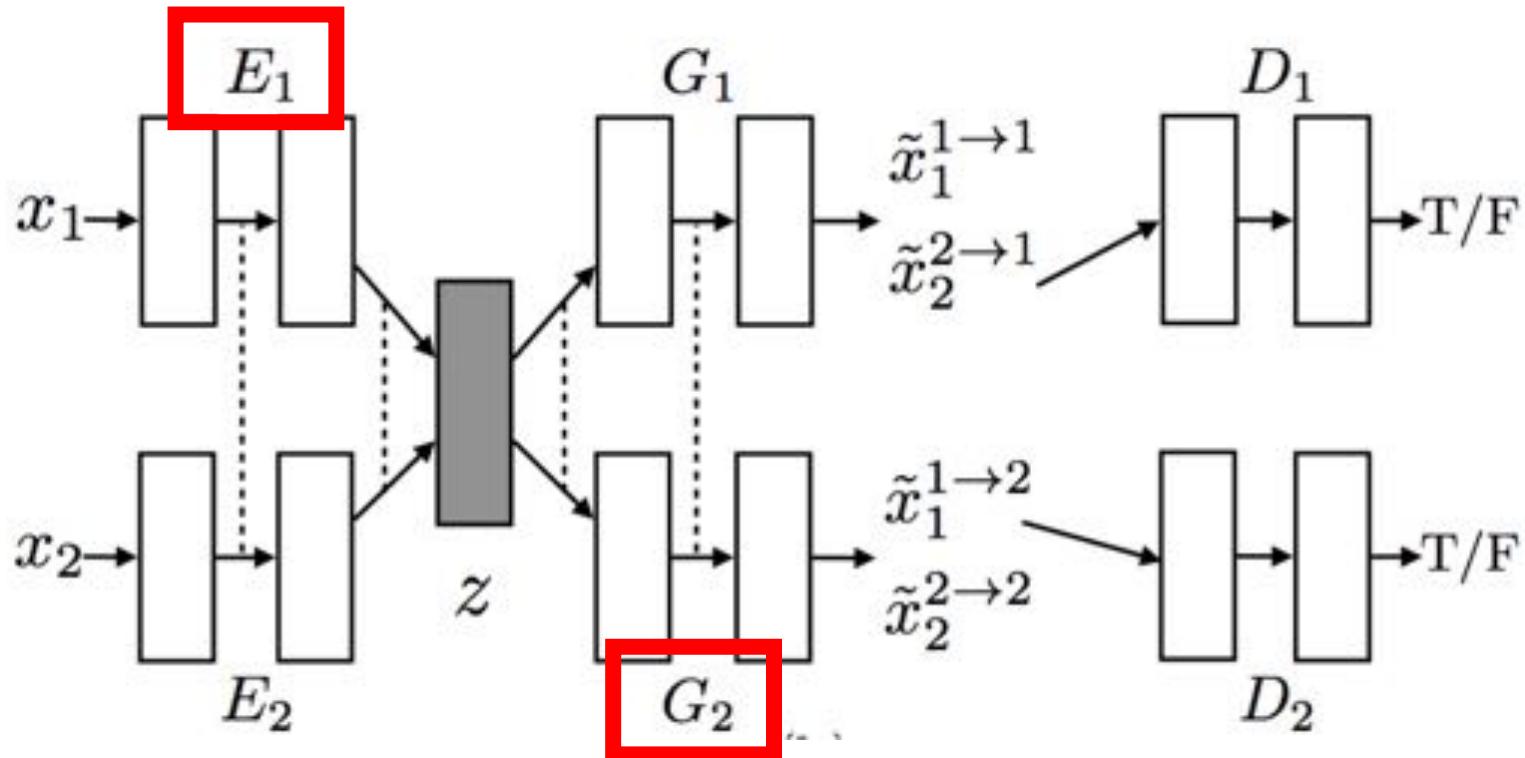


$$\begin{aligned} \mathcal{L}_{\text{VAE}_1}(E_1, G_1) = & \lambda_1 \text{KL}(q_1(z_1|x_1) || p_\eta(z)) \\ & - \lambda_2 \mathbb{E}_{z_1 \sim q_1(z_1|x_1)} [\log p_{G_1}(x_1|z_1)] \end{aligned}$$

Table 1: Interpretation of the roles of the subnetworks in the proposed framework.

Networks	$\{E_1, G_1\}$	$\{E_1, G_2\}$	$\{G_1, D_1\}$	$\{E_1, G_1, D_1\}$	$\{G_1, G_2, D_1, D_2\}$
Roles	VAE for \mathcal{X}_1	Image Translator $\mathcal{X}_1 \rightarrow \mathcal{X}_2$	GAN for \mathcal{X}_1	VAE-GAN [14]	CoGAN [17]

UNIT

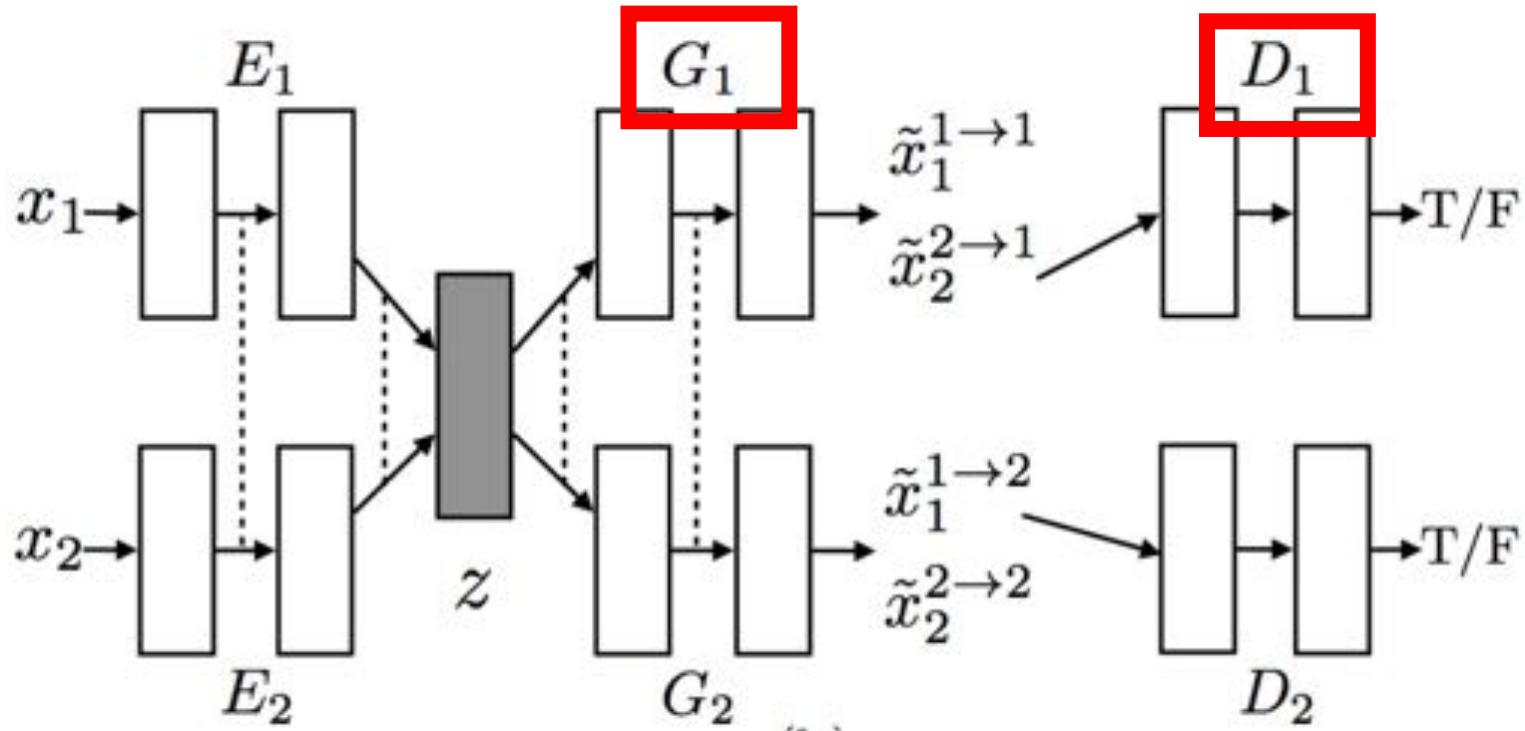


$$\tilde{x}_1^{1 \rightarrow 2} = G_2(z_1 \sim q_1(z_1 | x_1))$$

Table 1: Interpretation of the roles of the subnetworks in the proposed framework.

Networks	$\{E_1, G_1\}$	$\{E_1, G_2\}$	$\{G_1, D_1\}$	$\{E_1, G_1, D_1\}$	$\{G_1, G_2, D_1, D_2\}$
Roles	VAE for \mathcal{X}_1	Image Translator $\mathcal{X}_1 \rightarrow \mathcal{X}_2$	GAN for \mathcal{X}_1	VAE-GAN [14]	CoGAN [17]

UNIT



$$\mathcal{L}_{\text{GAN}_1}(E_1, G_1, D_1) = \lambda_0 \mathbb{E}_{x_1 \sim P_{\mathcal{X}_1}} [\log D_1(x_1)] + \lambda_0 \mathbb{E}_{z_2 \sim q_2(z_2|x_2)} [\log(1 - D_1(G_1(z_2)))]$$

Table 1: Interpretation of the roles of the subnetworks in the proposed framework.

Networks	$\{E_1, G_1\}$	$\{E_1, G_2\}$	$\{G_1, D_1\}$	$\{E_1, G_1, D_1\}$	$\{G_1, G_2, D_1, D_2\}$
Roles	VAE for \mathcal{X}_1	Image Translator $\mathcal{X}_1 \rightarrow \mathcal{X}_2$	GAN for \mathcal{X}_1	VAE-GAN [14]	CoGAN [17]

UNIT

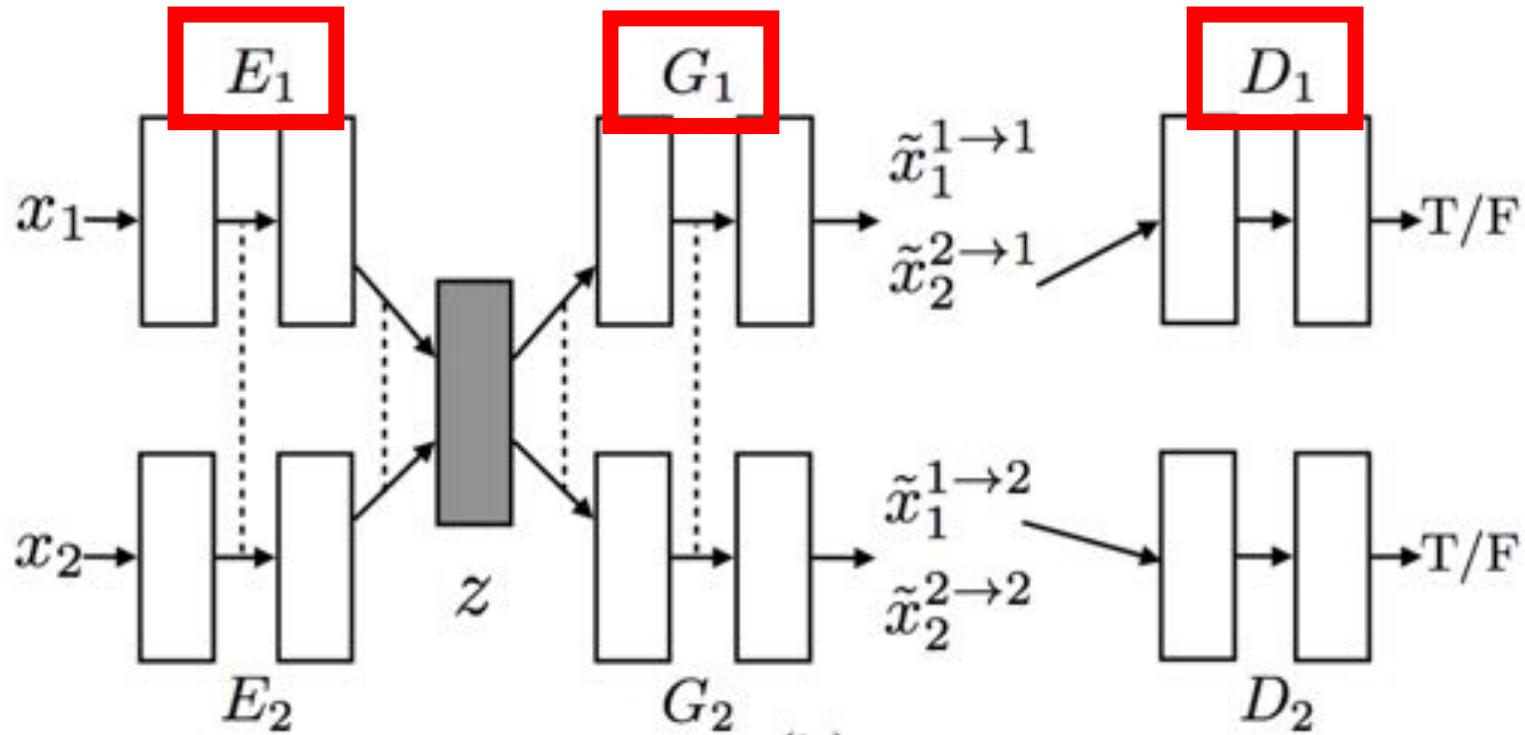


Table 1: Interpretation of the roles of the subnetworks in the proposed framework.

Networks	$\{E_1, G_1\}$	$\{E_1, G_2\}$	$\{G_1, D_1\}$	$\{E_1, G_1, D_1\}$	$\{G_1, G_2, D_1, D_2\}$
Roles	VAE for \mathcal{X}_1	Image Translator $\mathcal{X}_1 \rightarrow \mathcal{X}_2$	GAN for \mathcal{X}_1	VAE-GAN [14]	CoGAN [17]

UNIT

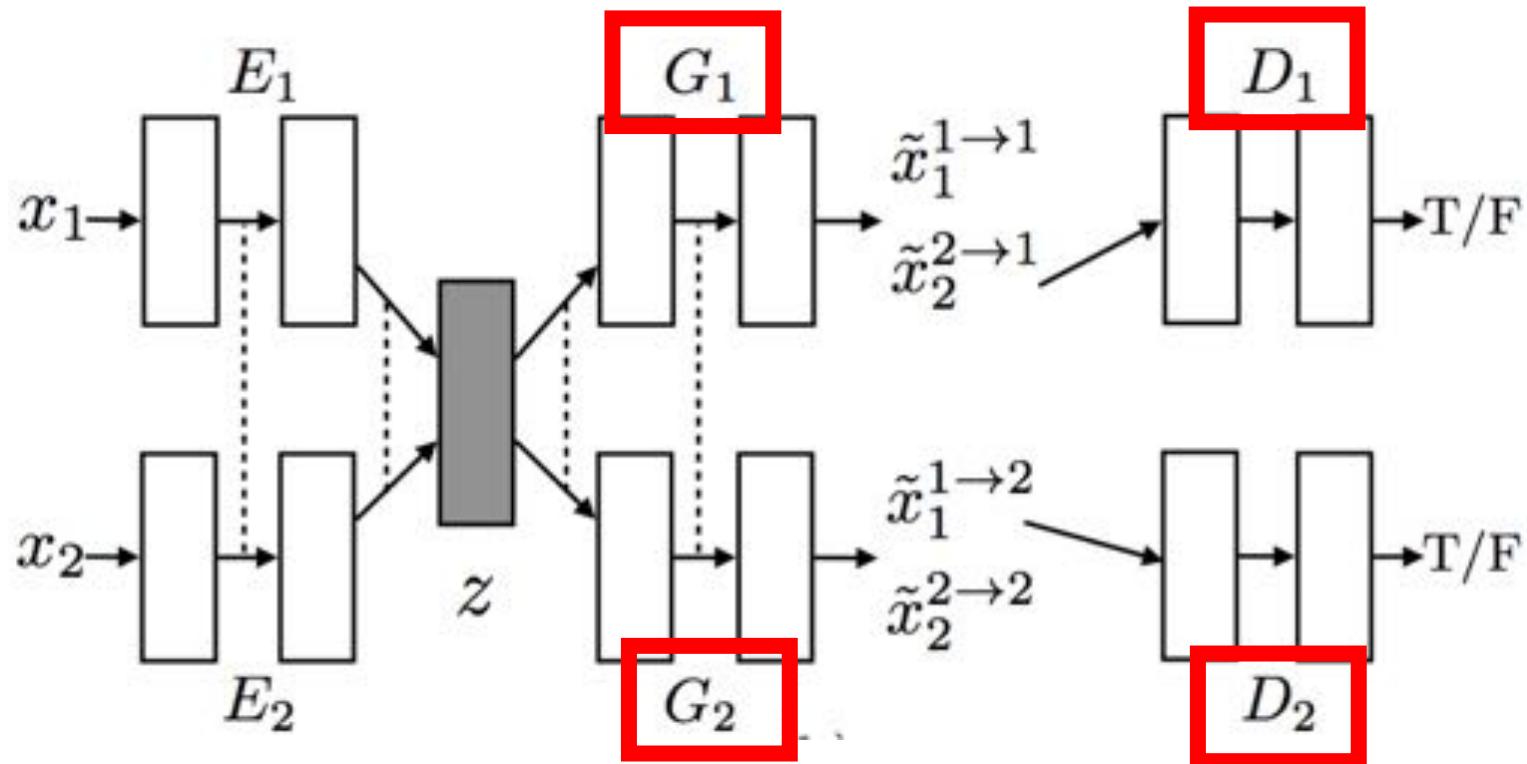


Table 1: Interpretation of the roles of the subnetworks in the proposed framework.

Networks	$\{E_1, G_1\}$	$\{E_1, G_2\}$	$\{G_1, D_1\}$	$\{E_1, G_1, D_1\}$	$\{G_1, G_2, D_1, D_2\}$
Roles	VAE for \mathcal{X}_1	Image Translator $\mathcal{X}_1 \rightarrow \mathcal{X}_2$	GAN for \mathcal{X}_1	VAE-GAN [14]	CoGAN [17]

Results



Results



Figure 4: Dog breed translation results.



Results

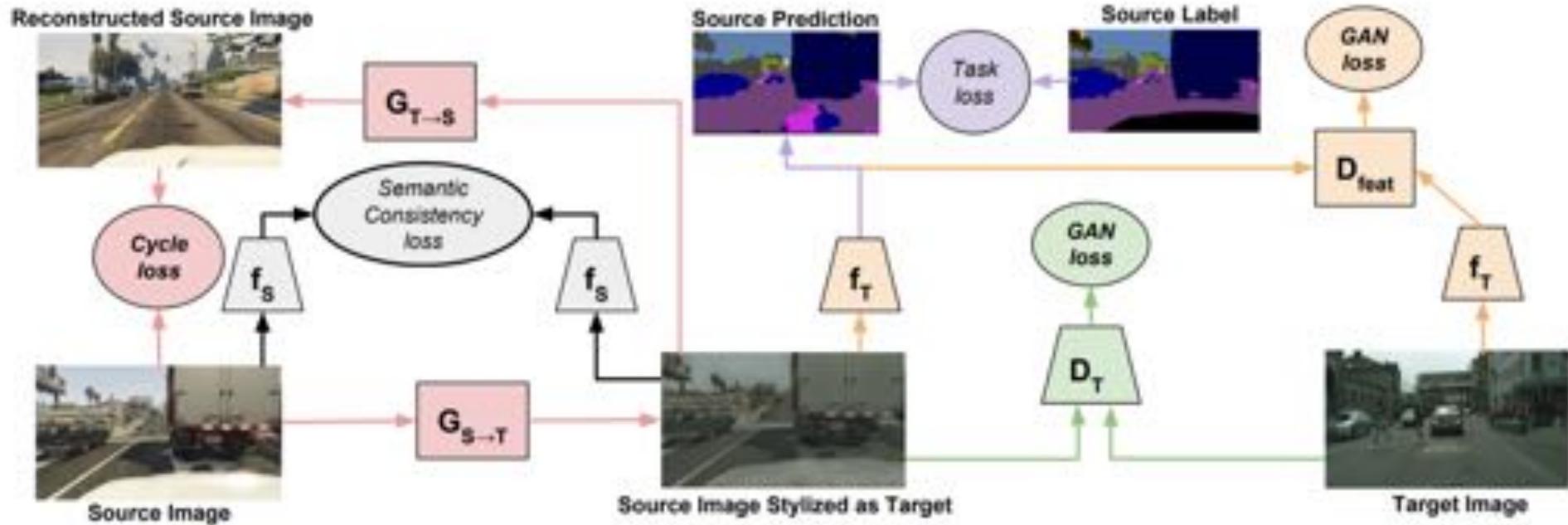


Figure 5: Cat species translation results.



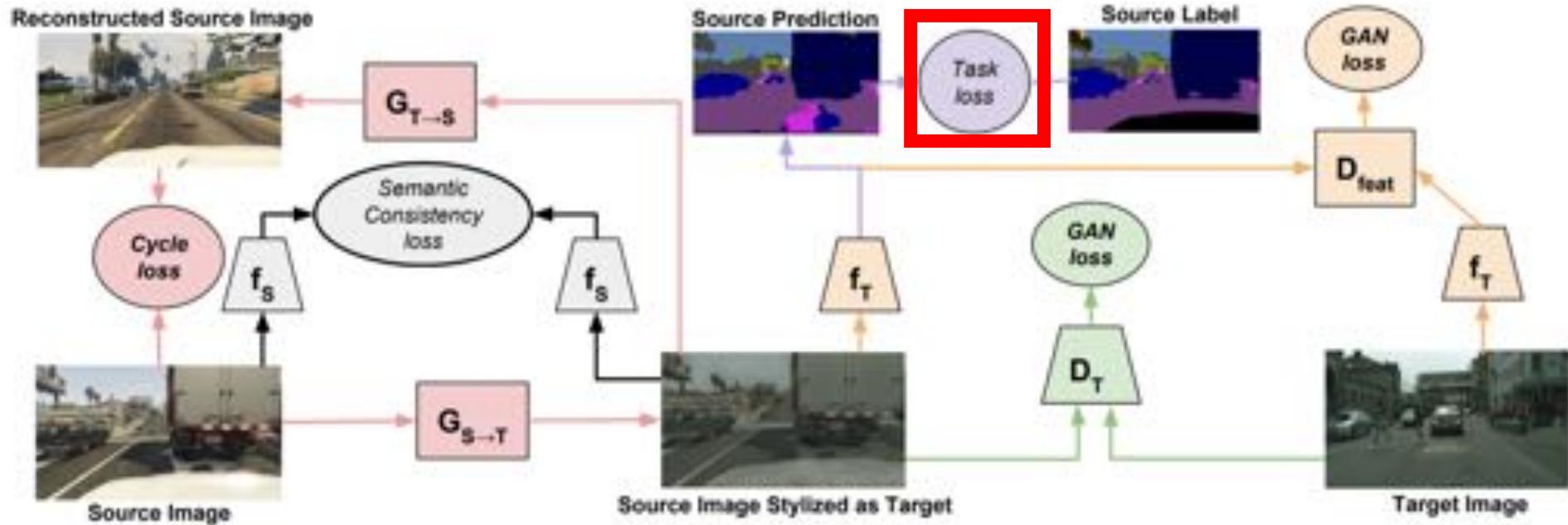
Figure 6: Attribute-based face translation results.

Application to Domain Adaptation



Hoffman, Judy, et al. "CyCADA: Cycle-Consistent Adversarial Domain Adaptation." *arXiv preprint arXiv:1711.03213* (2017).

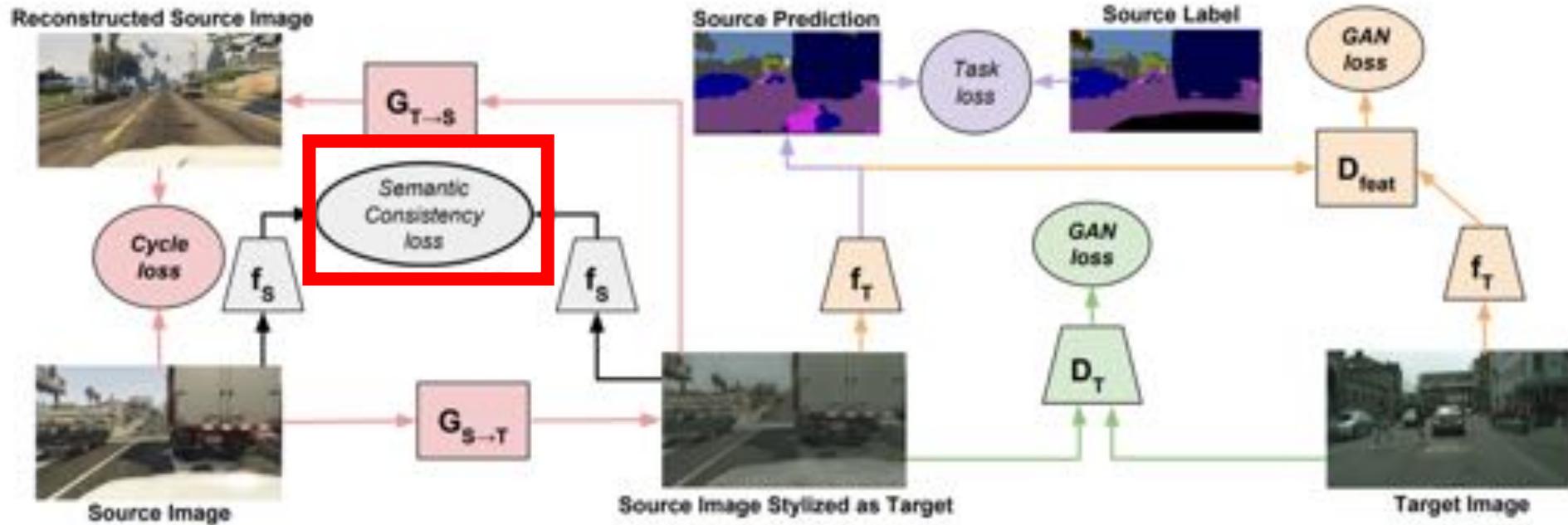
Application to Domain Adaptation



$$\mathcal{L}_{\text{task}}(f_S, X_S, Y_S) = -\mathbb{E}_{(x_s, y_s) \sim (X_S, Y_S)} \sum_{k=1}^K \mathbb{1}_{[k=y_s]} \log \left(\sigma(f_S^{(k)}(x_s)) \right)$$

$$\begin{aligned} \mathcal{L}_{\text{sem}}(G_{S \rightarrow T}, G_{T \rightarrow S}, X_S, X_T, f_S) &= \mathcal{L}_{\text{task}}(f_S, G_{T \rightarrow S}(X_T), p(f_S, X_T)) \\ &\quad + \mathcal{L}_{\text{task}}(f_S, G_{S \rightarrow T}(X_S), p(f_S, X_S)) \end{aligned}$$

Application to Domain Adaptation



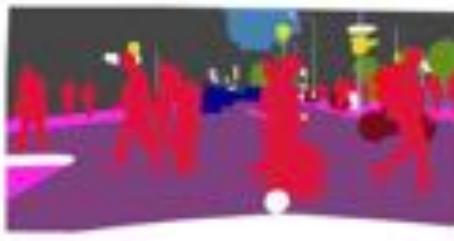
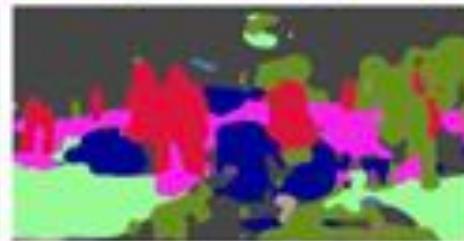
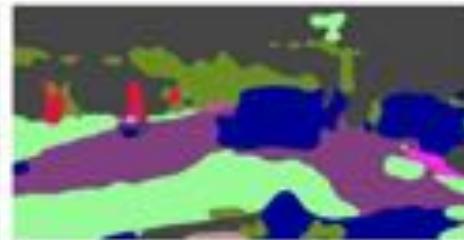
$$\begin{aligned}\mathcal{L}_{\text{sem}}(G_{S \rightarrow T}, G_{T \rightarrow S}, X_S, X_T, f_S) = & \mathcal{L}_{\text{task}}(f_S, G_{T \rightarrow S}(X_T), p(f_S, X_T)) \\ & + \mathcal{L}_{\text{task}}(f_S, G_{S \rightarrow T}(X_S), p(f_S, X_S))\end{aligned}$$

Results



(a) GTA5

(b) GTA5 → Cityscapes



(a) Test Image

(b) Source Prediction

(c) CyCADA Prediction

(d) Ground Truth