Applications in Computer Vision (cont'd)

+

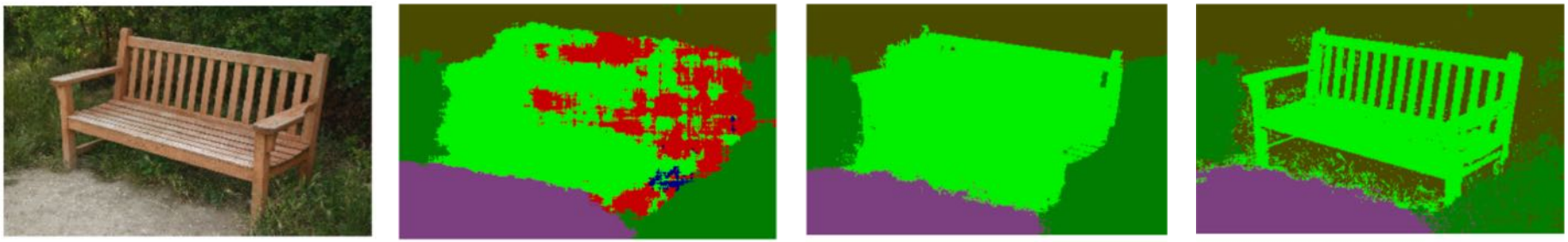
Gaussian Processes

Mingming Gong
Kayhan Batmanghelich

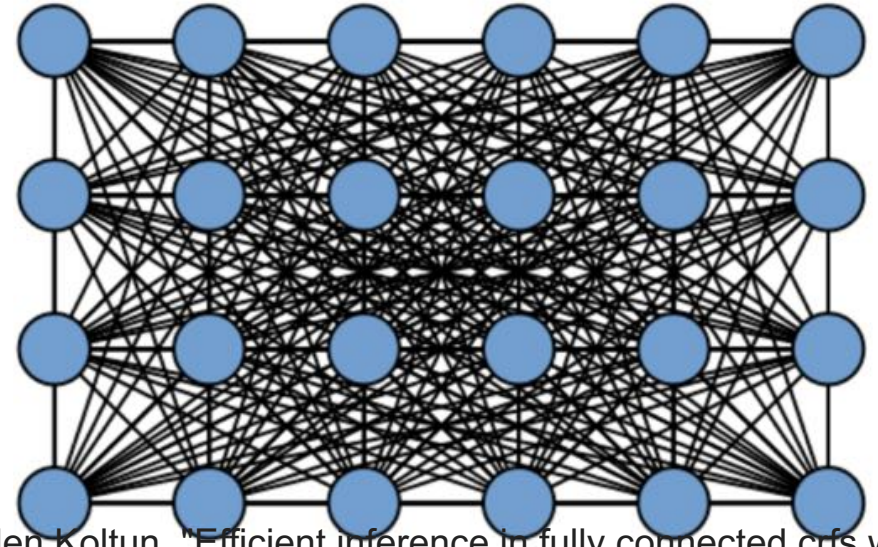
Outline

- Semantic Segmentation
- Image-to-Image Translation

Recap: Fully Connected CRF



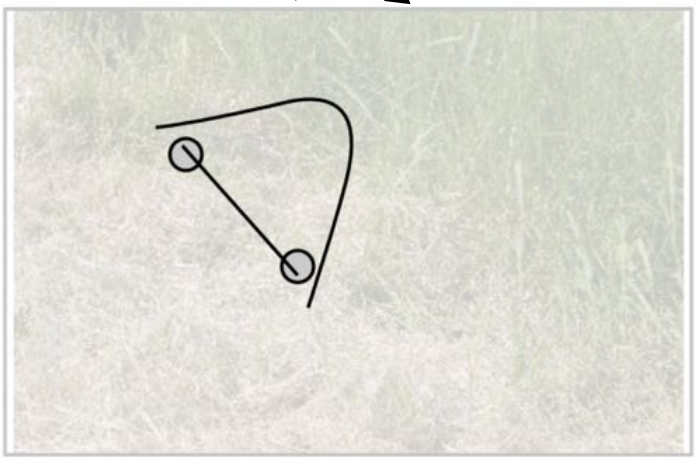
(a) Image (b) Unary classifiers (c) Robust P^n CRF (d) Fully connected CRF, MCMC inference, 36 hrs

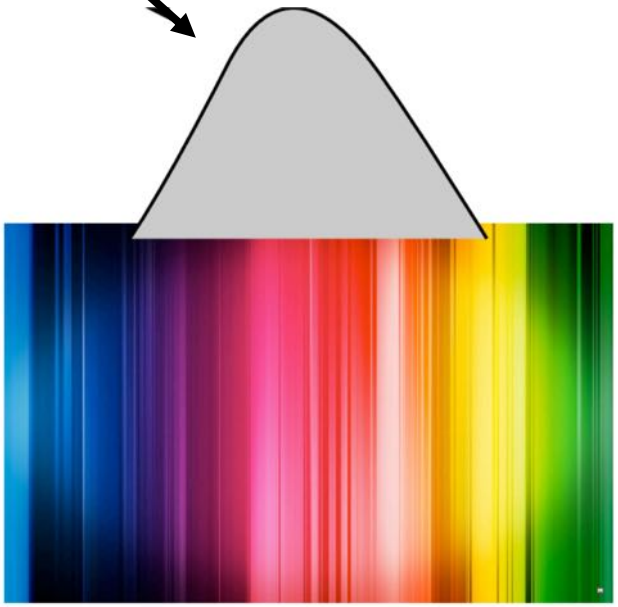


Krähenbühl, Philipp, and Vladlen Koltun. "Efficient inference in fully connected crfs with gaussian edge potentials." *Advances in neural information processing systems*. 2011.

Recap: Detailed model

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





https://www.cs.cmu.edu/~efros/courses/LBMV12/crf_deconstruction.pdf

Recap: Message Passing

- Mean field approximation
  - Minimize KL divergence $D(Q||P)$,

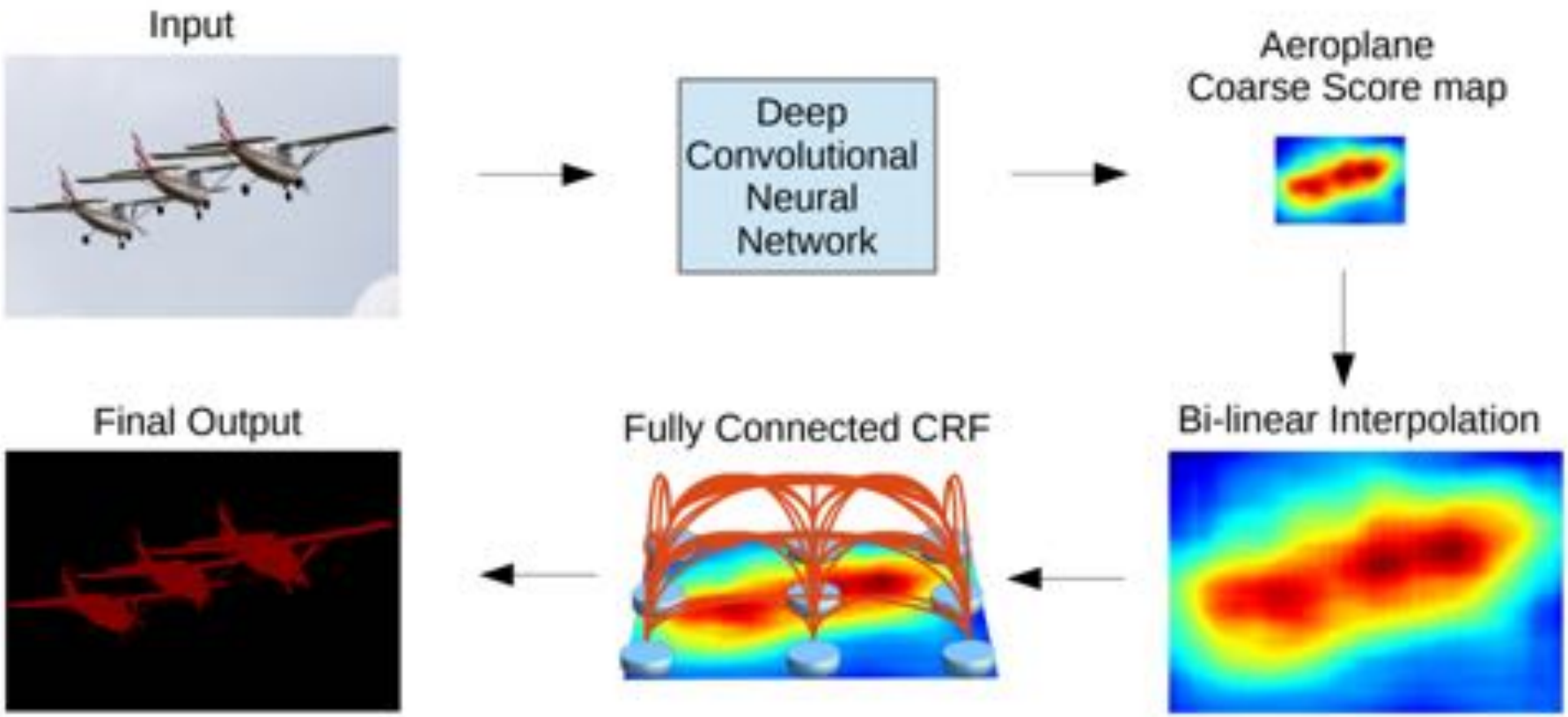
$$\text{s.t. } Q(X) = \prod_i Q_i(X_i)$$

- Iterative update equation

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

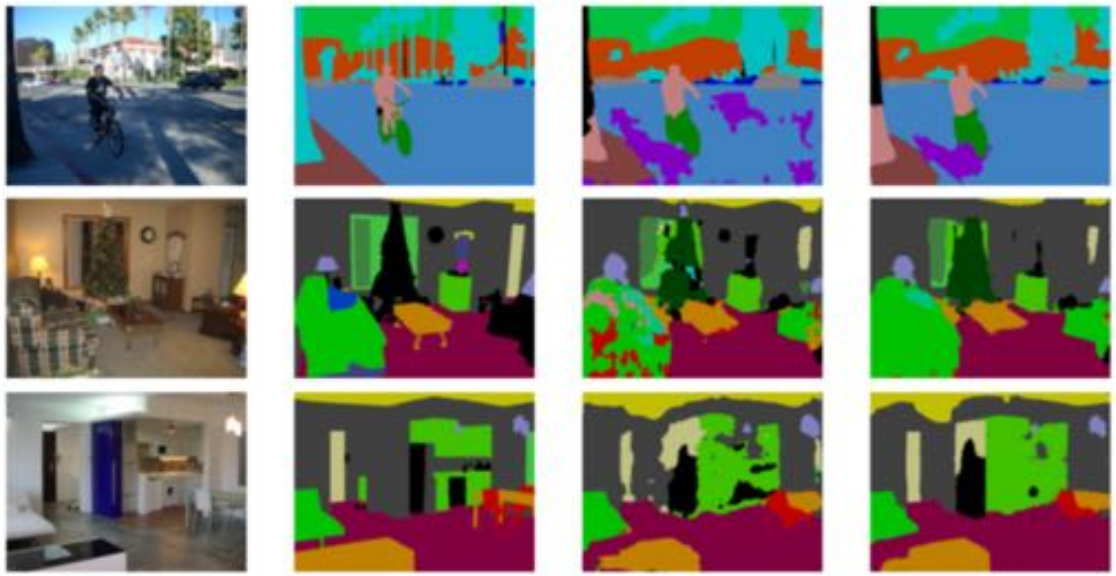
- Message passing from all X_j to all X_i

Recap: DeepLab-CRF



Recap: Some Result (PASCAL VOC)

- PASCAL dataset
  - 10, 582 images
  - 21 classes



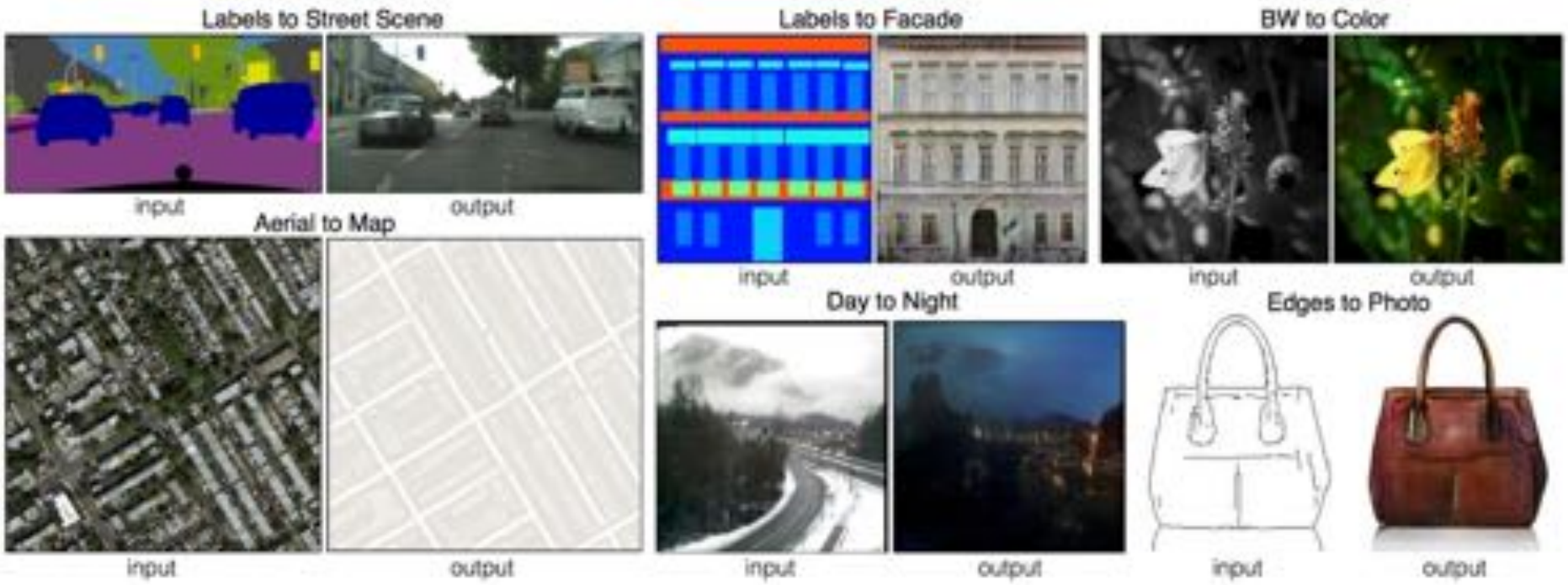
(a) Image (b) G.T. (c) Before CRf (d) After CRF

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

Outline

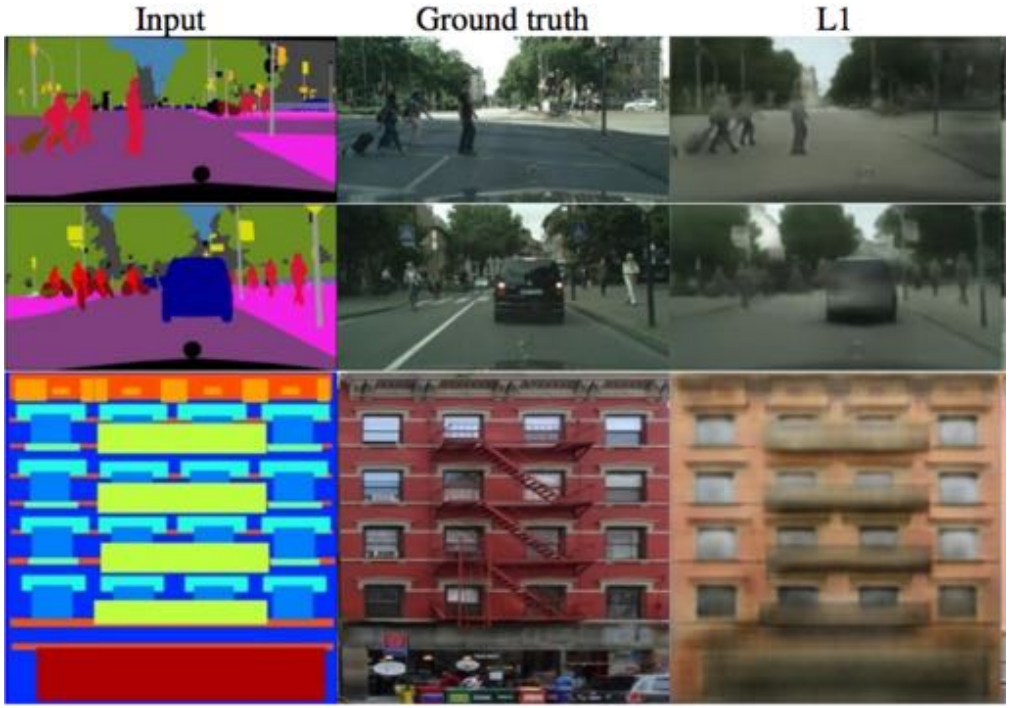
- Semantic Segmentation
- Image-to-Image Translation

Image-to-Image Translation

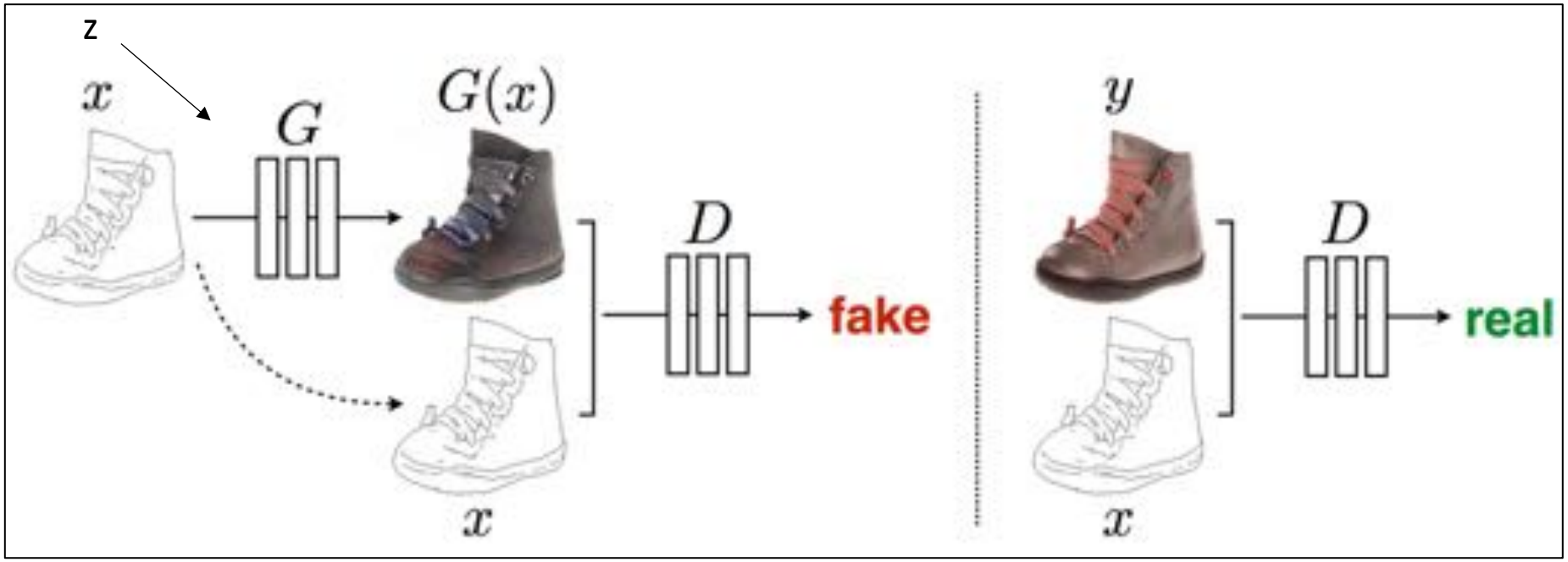


Supervised Learning

- Multi-output regression
- $\left|\left|G(X) - Y\right|\right|_1$



Conditional GAN

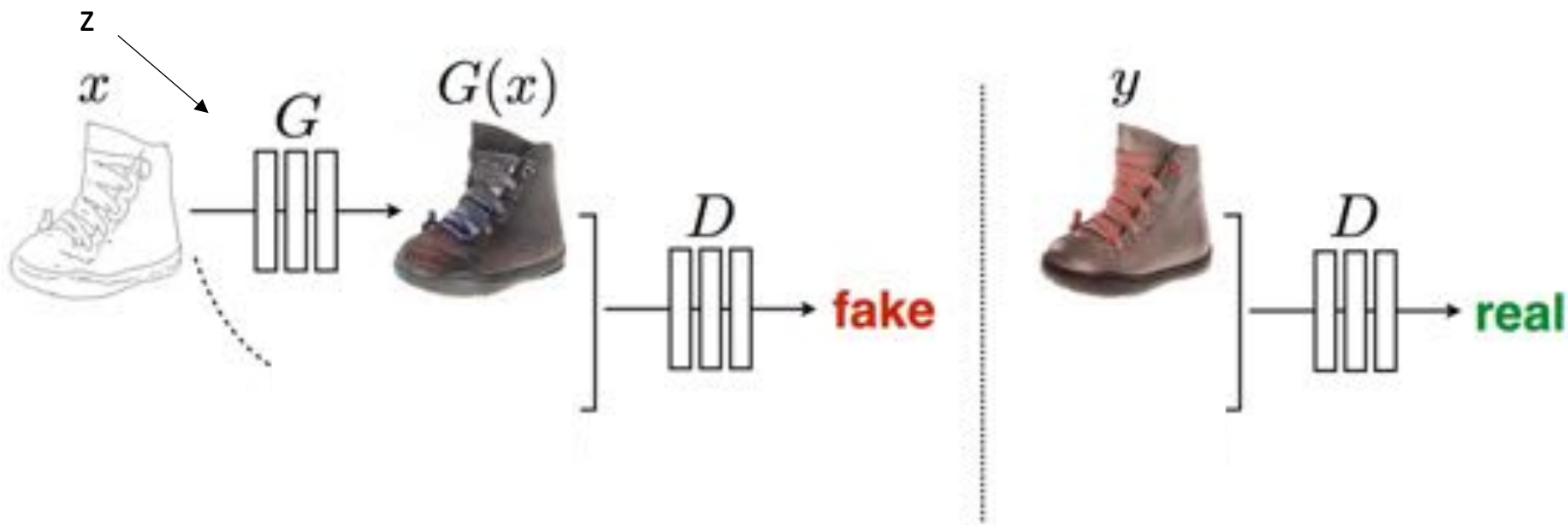


$$\mathcal{L}_{cGAN}(G, D) = \mathbb{E}_{x,y}[\log D(x, y)] + \mathbb{E}_{x,z}[\log(1 - D(x, G(x, z))],$$

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

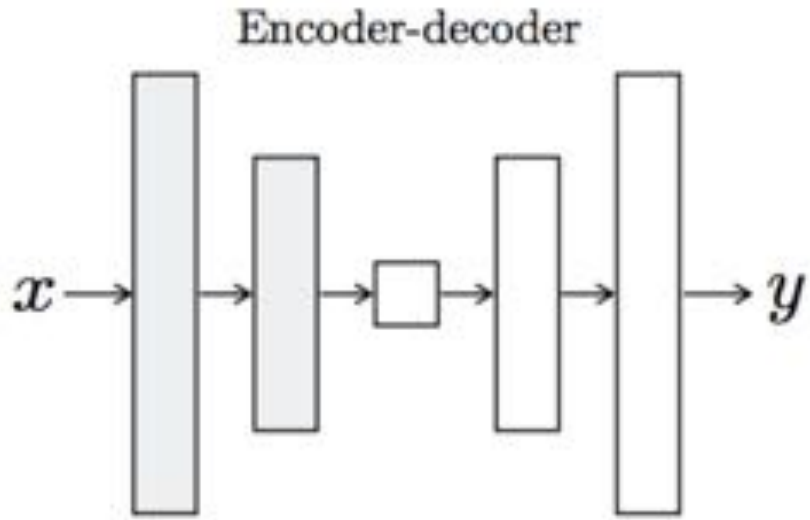
Isola, Phillip, et al. "Image-to-image translation with conditional adversarial networks." *arXiv preprint* (2017).

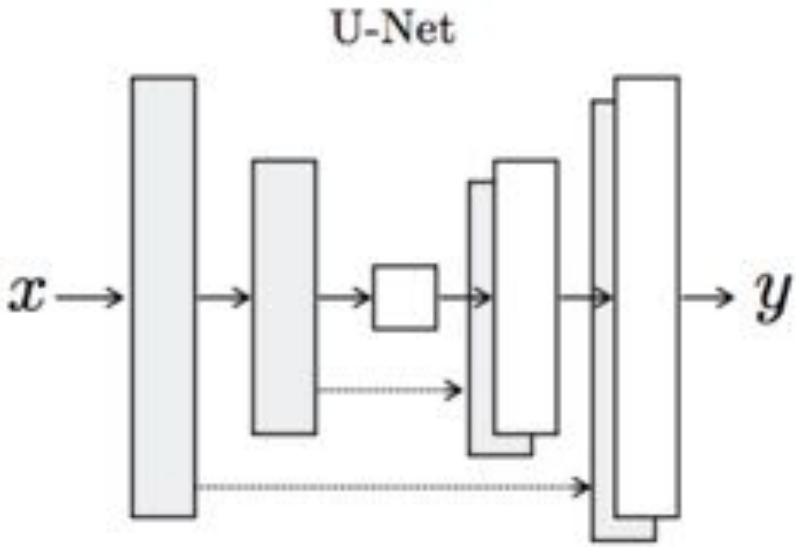
GAN



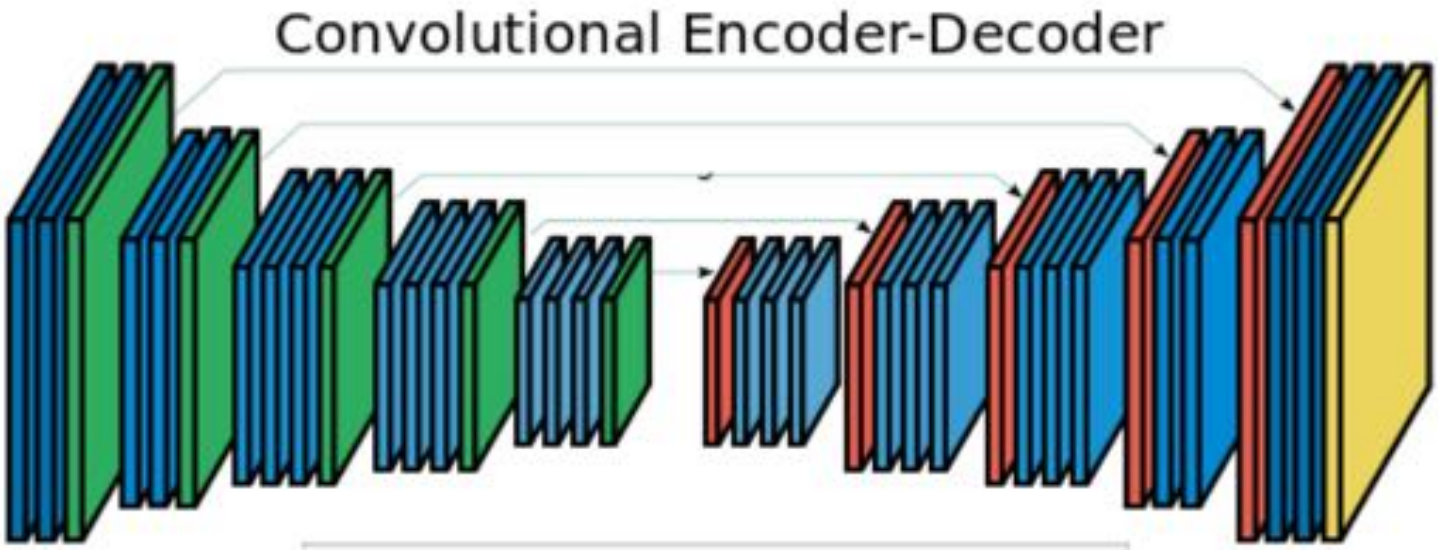
$$\mathcal{L}_{GAN}(G, D) = \mathbb{E}_{x,y}[\log D(\ \ , y)] + \mathbb{E}_{x,z}[\log(1 - D(\ \ , G(x, z))],$$

Generator (G)

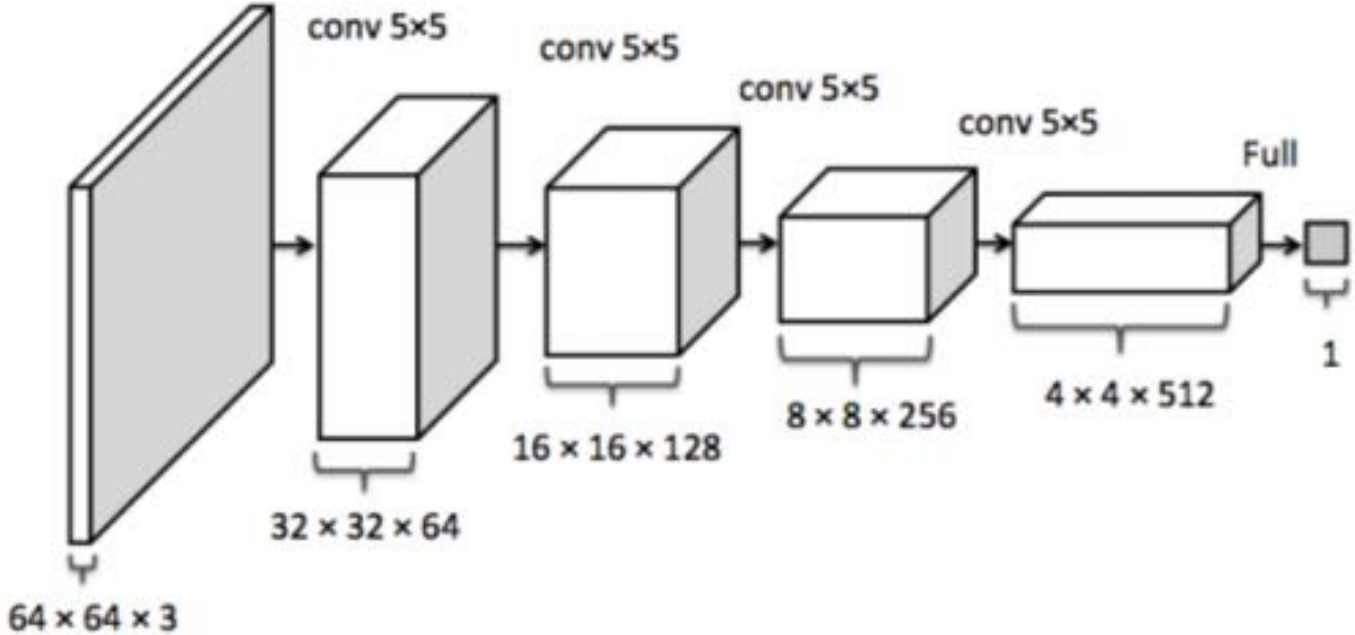




Generator (G)



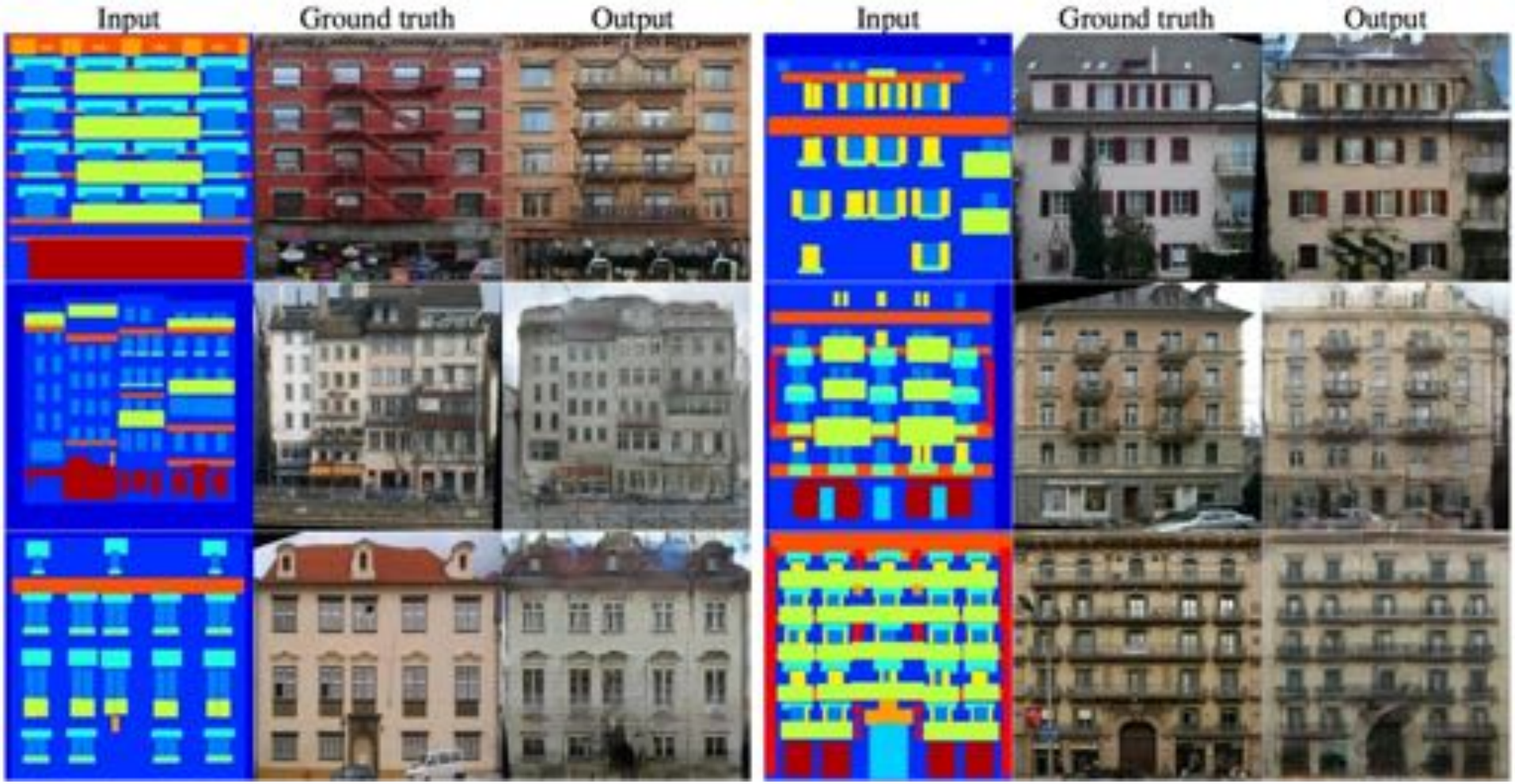
Discriminator (D)



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

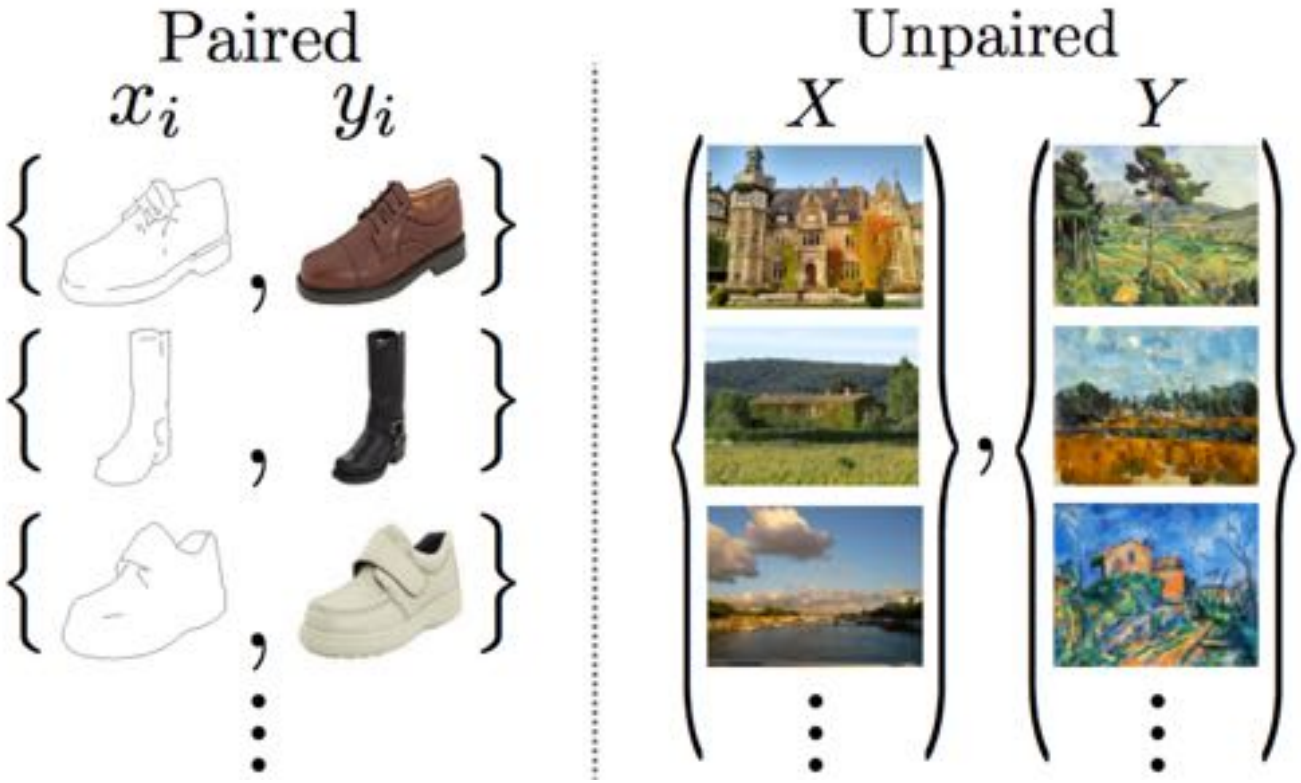
Edges->Images

Labels -> Images



Day-> Night

Unpaired Data



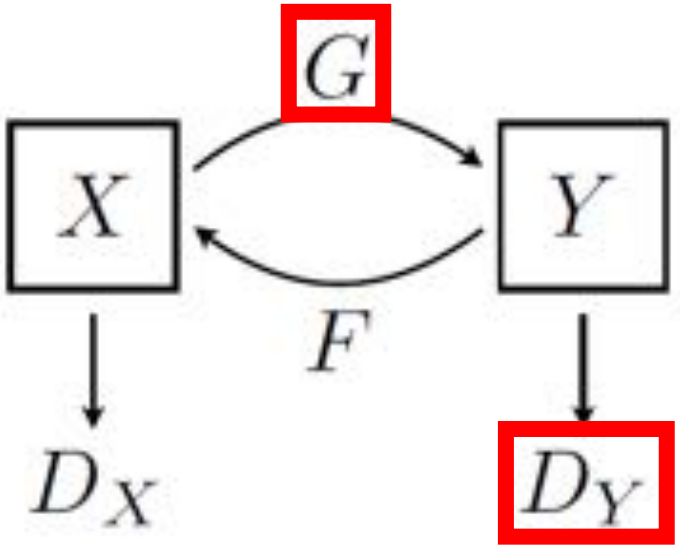
P(X, Y)

P(X) P(Y)

P(Y|X)

Infinite solutions!!

Cycle GAN

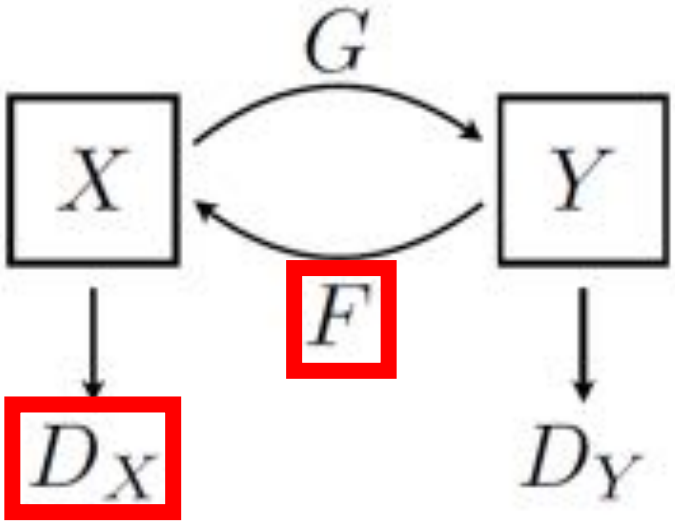


Discriminator $\mathrm{D_Y}$: $L_{GAN}(G(x), y)$
Real zebras vs. generated zebras

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

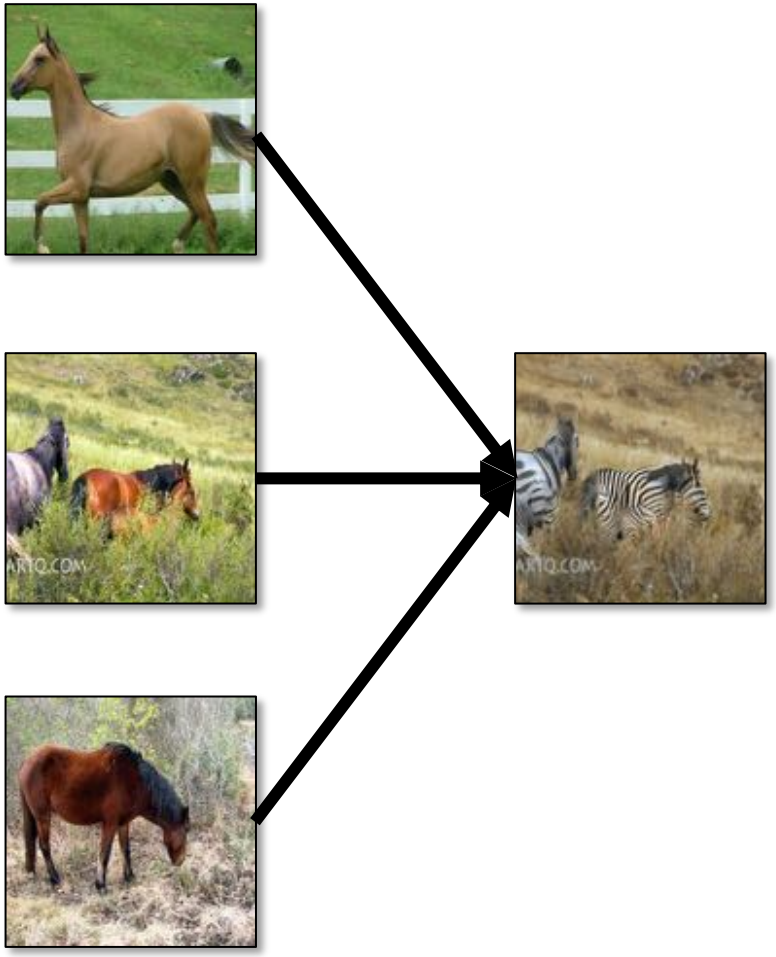


Discriminator $\mathrm{D_Y}$: $L_{GAN}(G(x), y)$

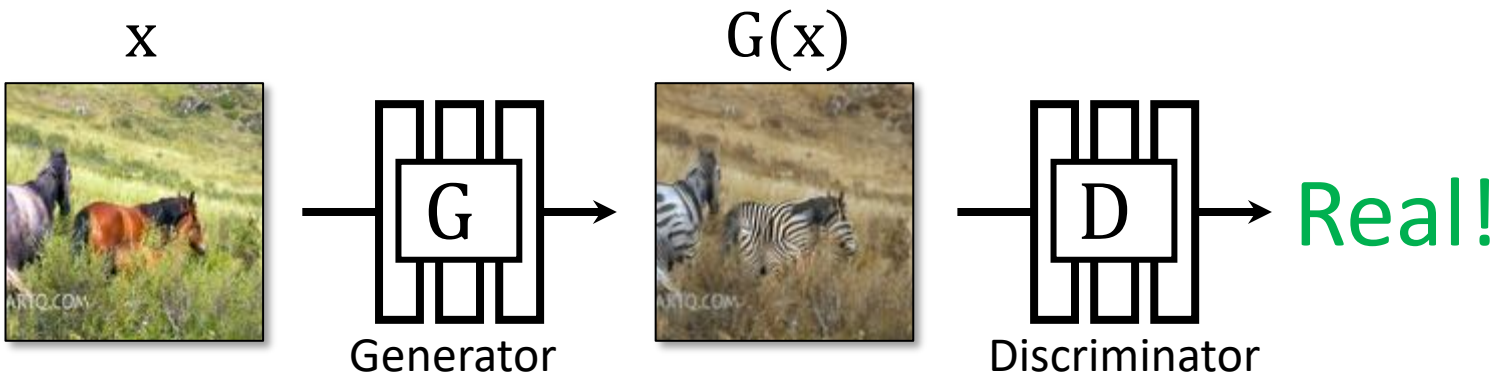
Real zebras vs. generated zebras

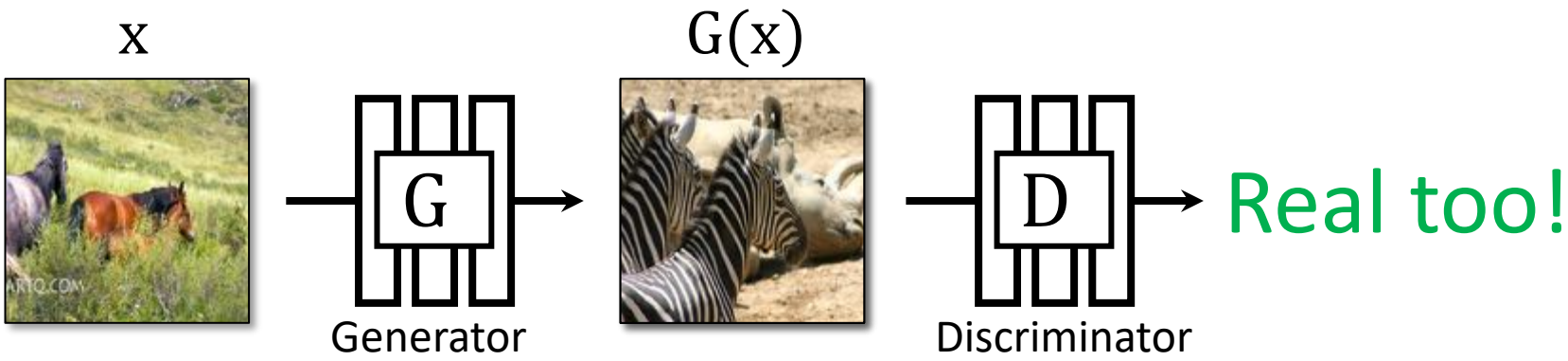
Discriminator $\mathrm{D_X}$: $L_{GAN}(F(y), x)$

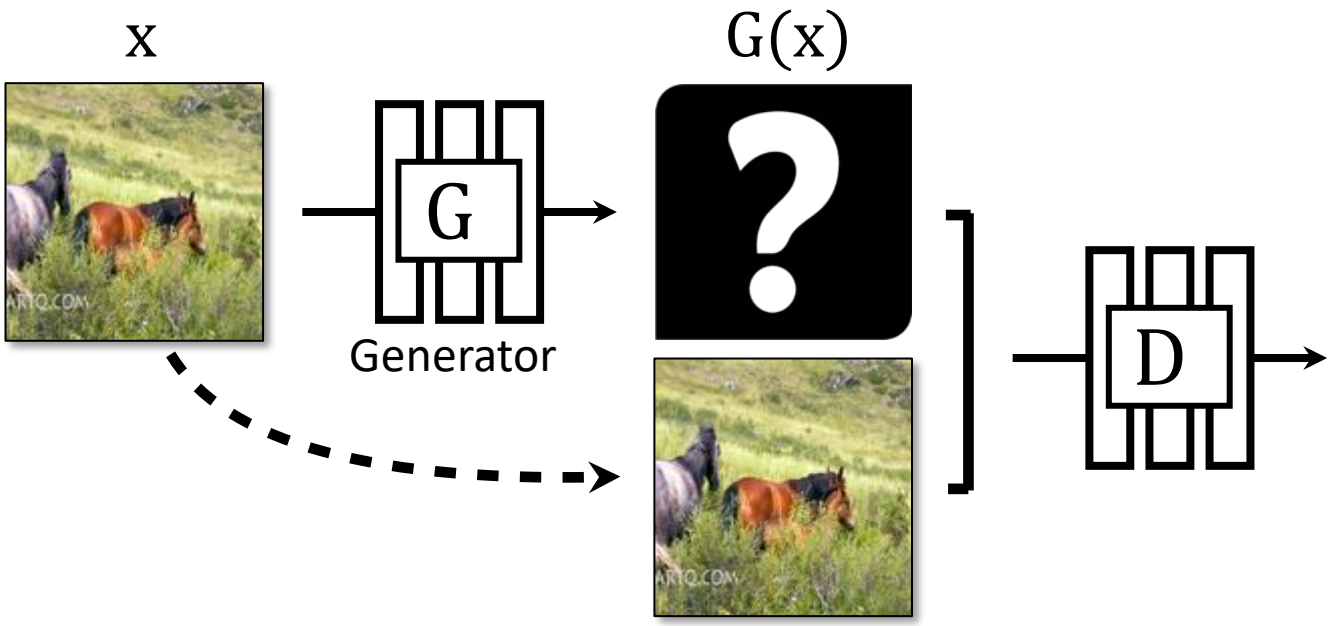
Real horses vs. generated horses



mode collapse!


x
G
Generator
G(x)
D
Discriminator
Real!

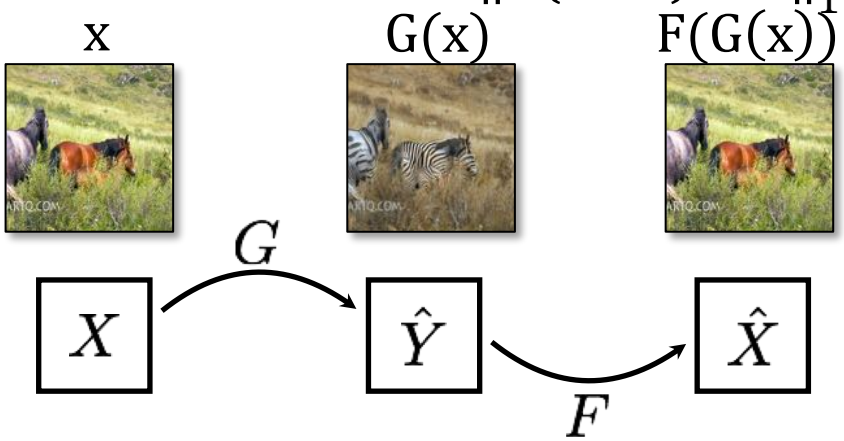

x
G
Generator
G(x)
D
Discriminator
Real too!

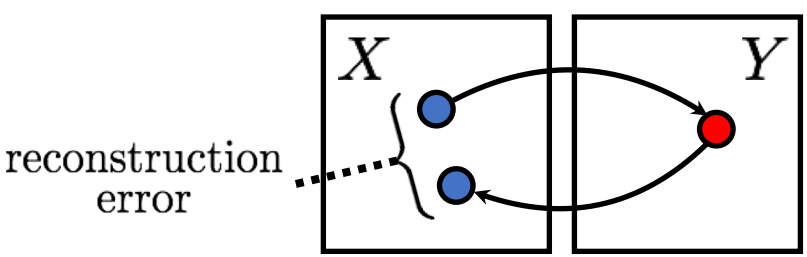


No input-output pairs!

Cycle-consistency Loss

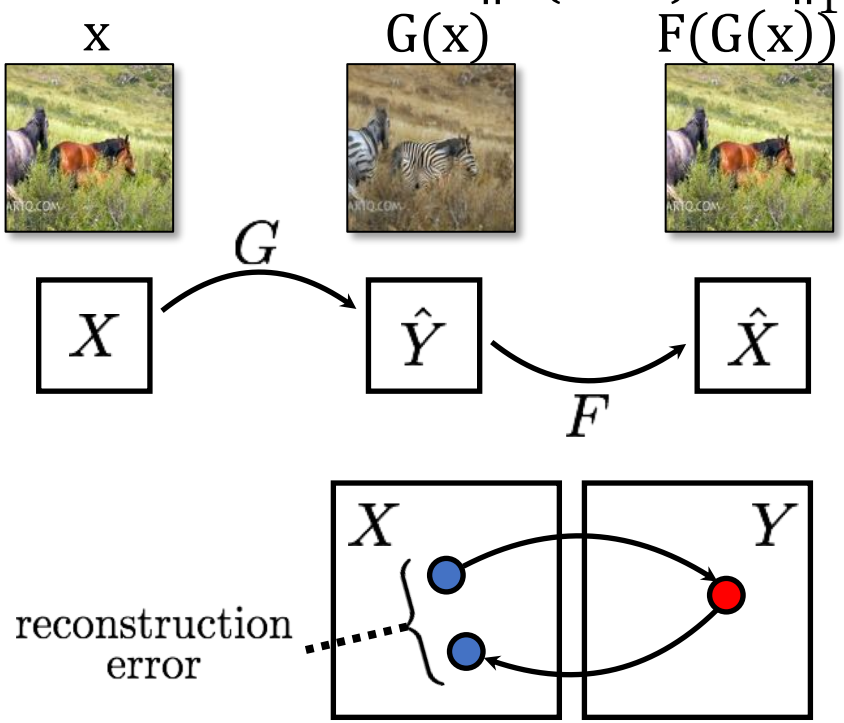
Forward cycle loss: $\|F(G(x)) - x\|_1$



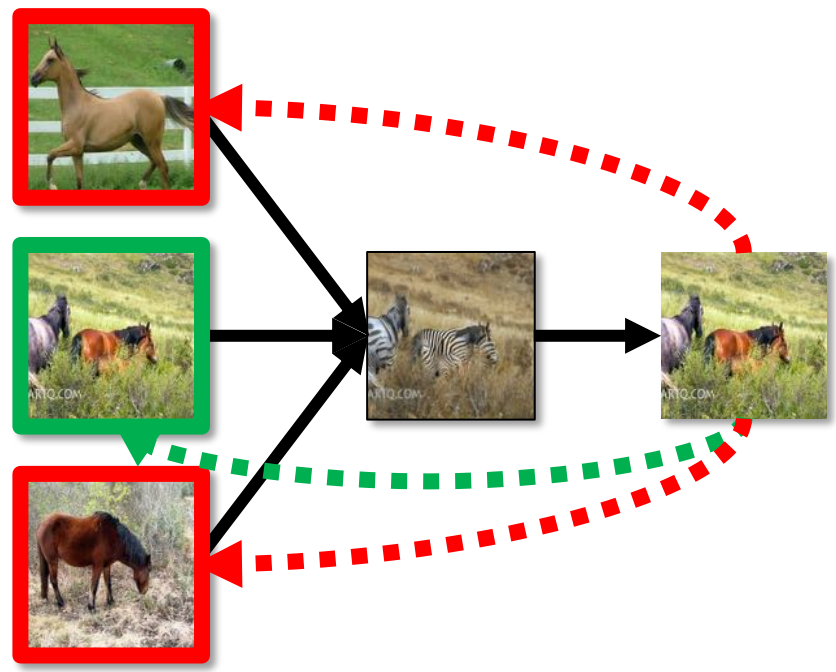


Cycle-consistency Loss

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



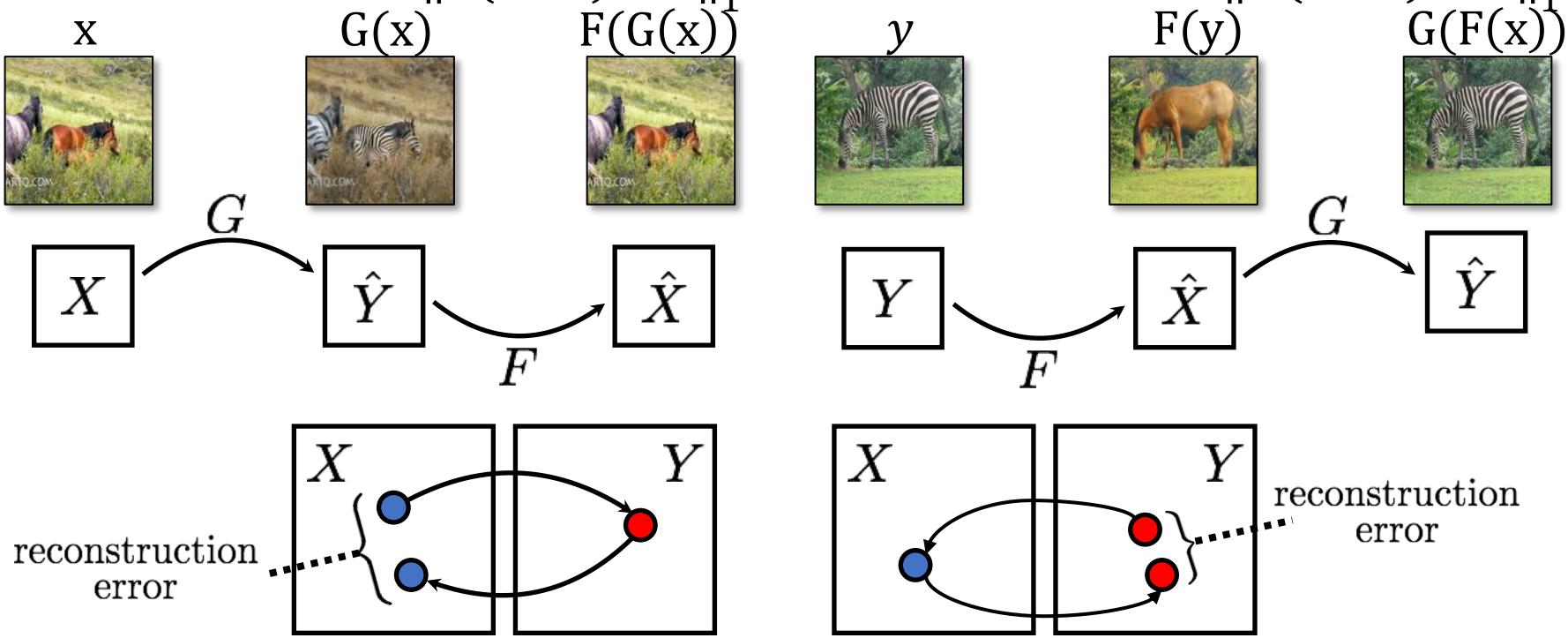
Small cycle loss



Cycle-consistency Loss

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

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



Objective

GAN loss:

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

Cycle consistent loss:

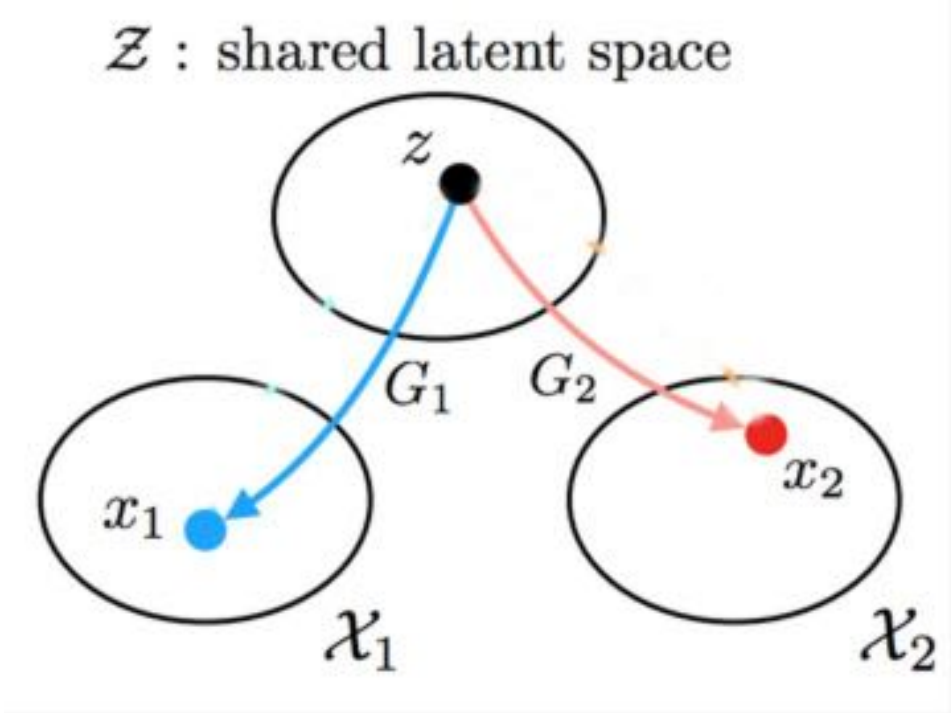
$$\mathcal{L}_{\text{cyc}}(G, F) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\|F(G(x)) - x\|_1] + \mathbb{E}_{y \sim p_{\text{data}}(y)}[\|G(F(y)) - y\|_1].$$

Autoencoder


Input
Monet
Van Gogh
Cezanne
Ukiyo-e

Failure cases

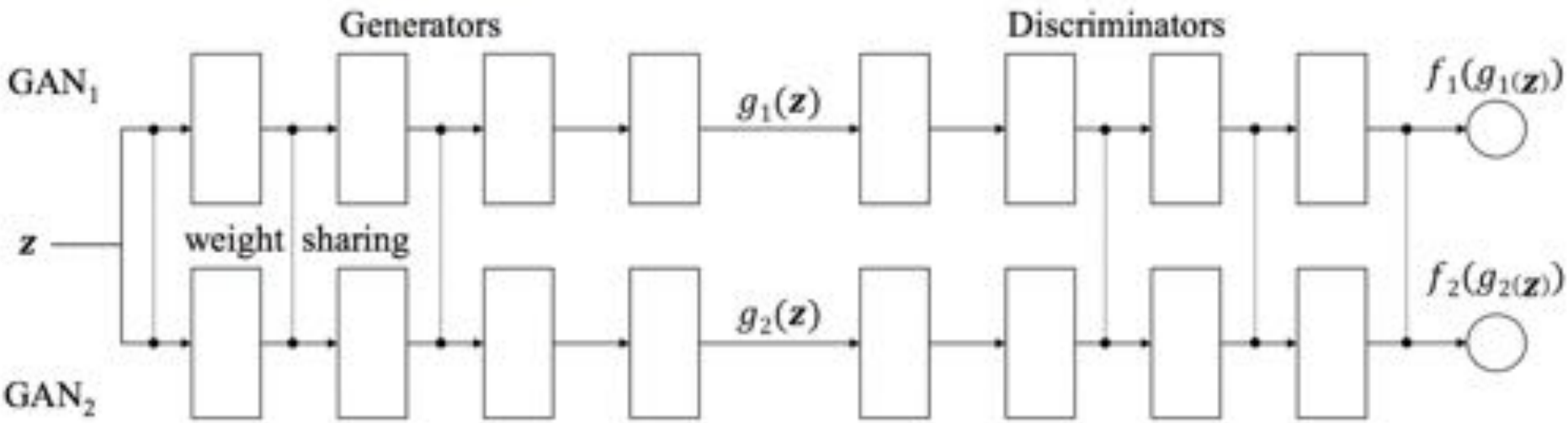
Coupled GAN



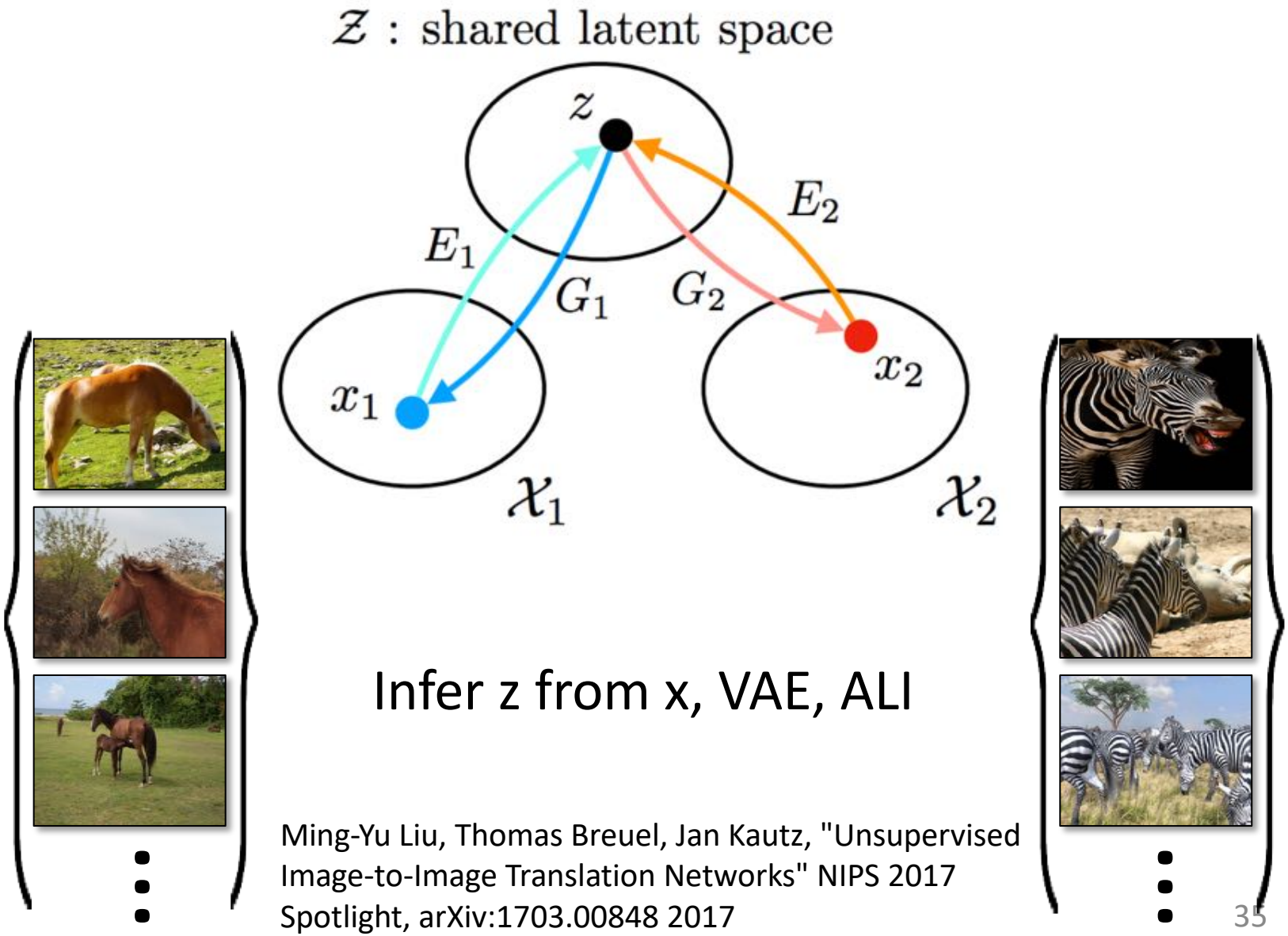
Constrain G_1 and G_2

M.-Y. Liu and O. Tuzel. Coupled generative adversarial networks. In NIPS, pages 469–477, 2016. 3, 5

Layer Sharing



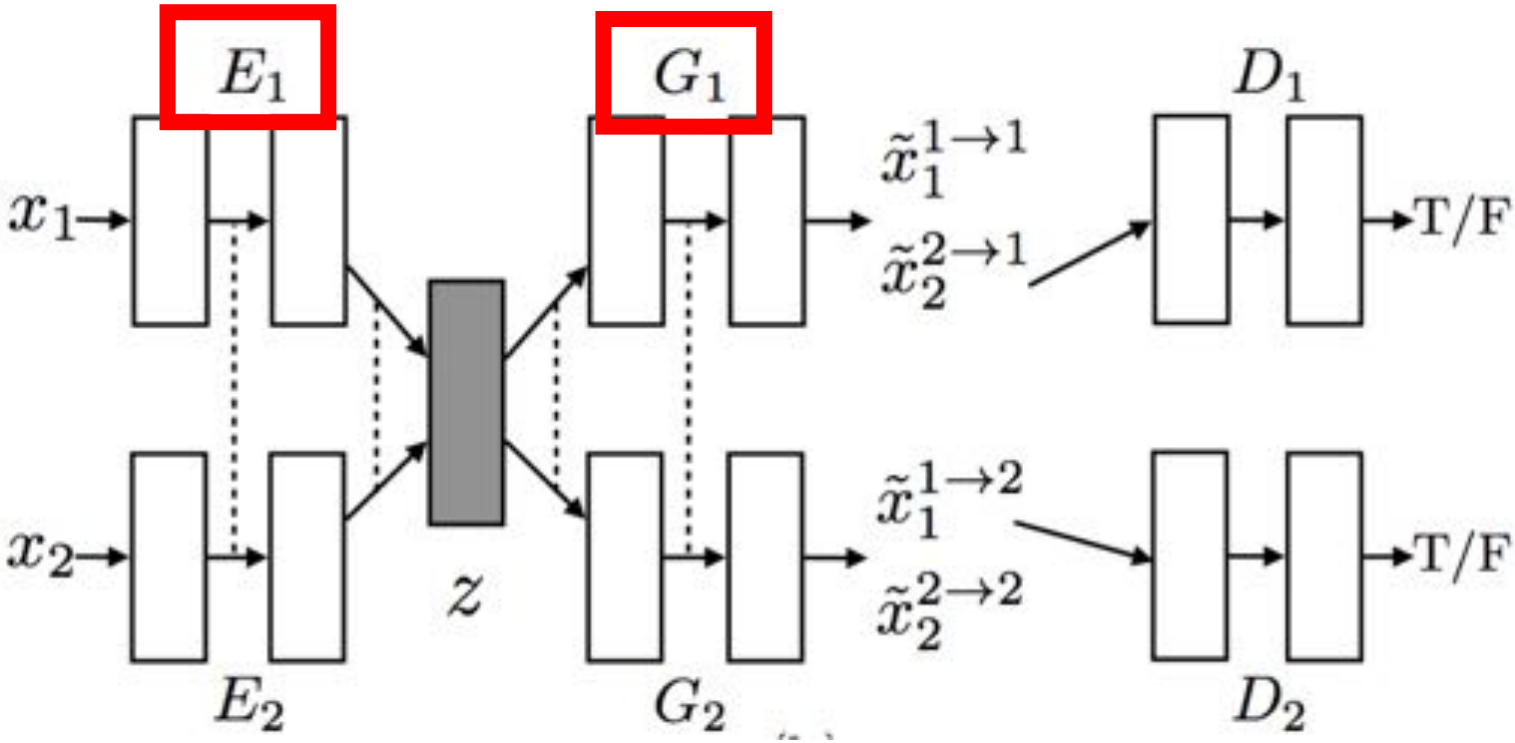
UNIT



Infer z from x, VAE, ALI

Ming-Yu Liu, Thomas Breuel, Jan Kautz, "Unsupervised Image-to-Image Translation Networks" NIPS 2017 Spotlight, arXiv:1703.00848 2017

UNIT

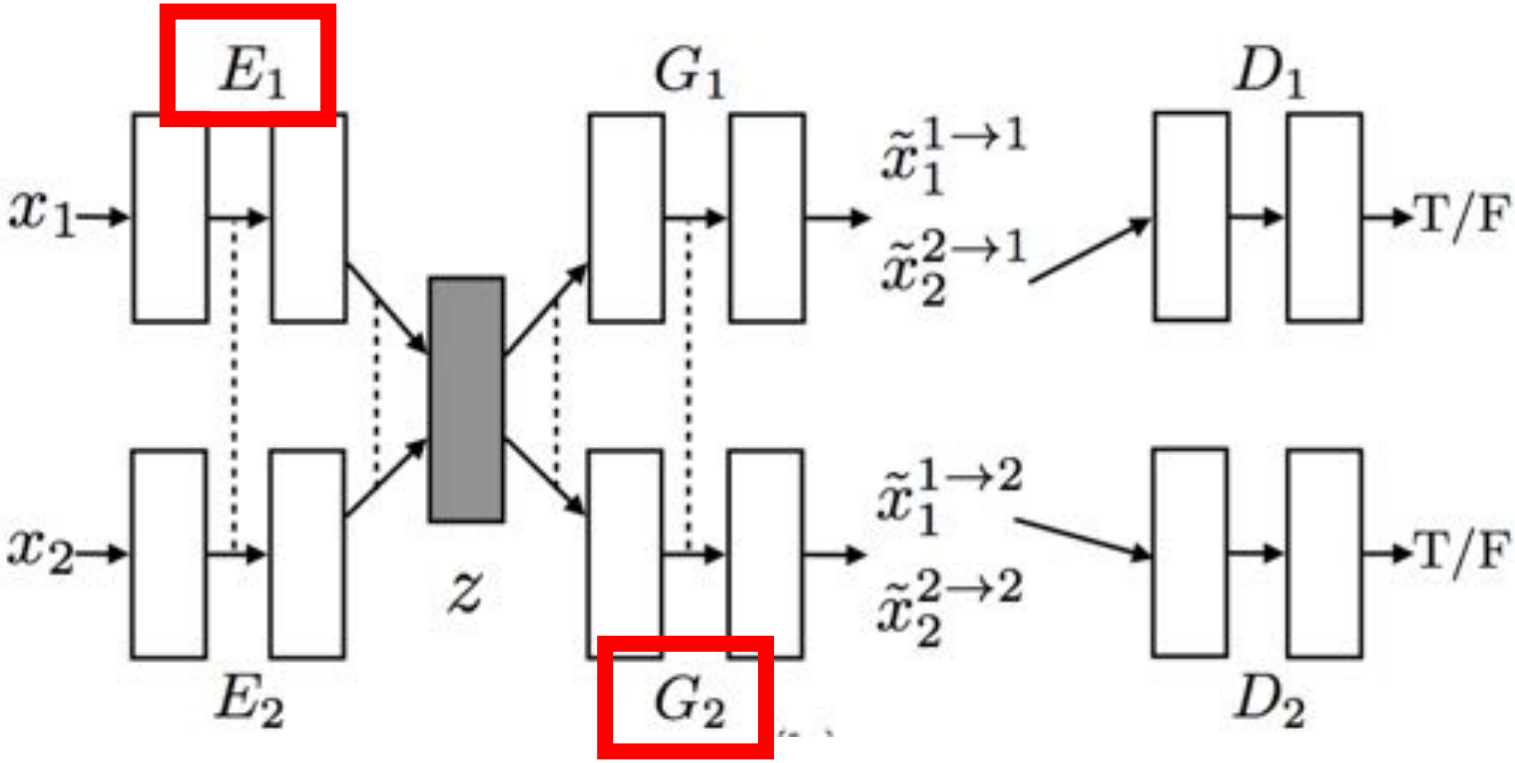


$$\mathcal{L}_{\text{VAE}_1}(E_1, G_1) = \lambda_1 \text{KL}(q_1(z_1|x_1)||p_\eta(z)) - \lambda_2 \mathbb{E}_{z_1 \sim q_1(z_1|x_1)}[\log p_{G_1}(x_1|z_1)]$$

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 $\mathcal{X}_1 \to \mathcal{X}_2$	GAN for $\mathcal{X}_1$	VAE-GAN [14]	CoGAN [17]

UNIT

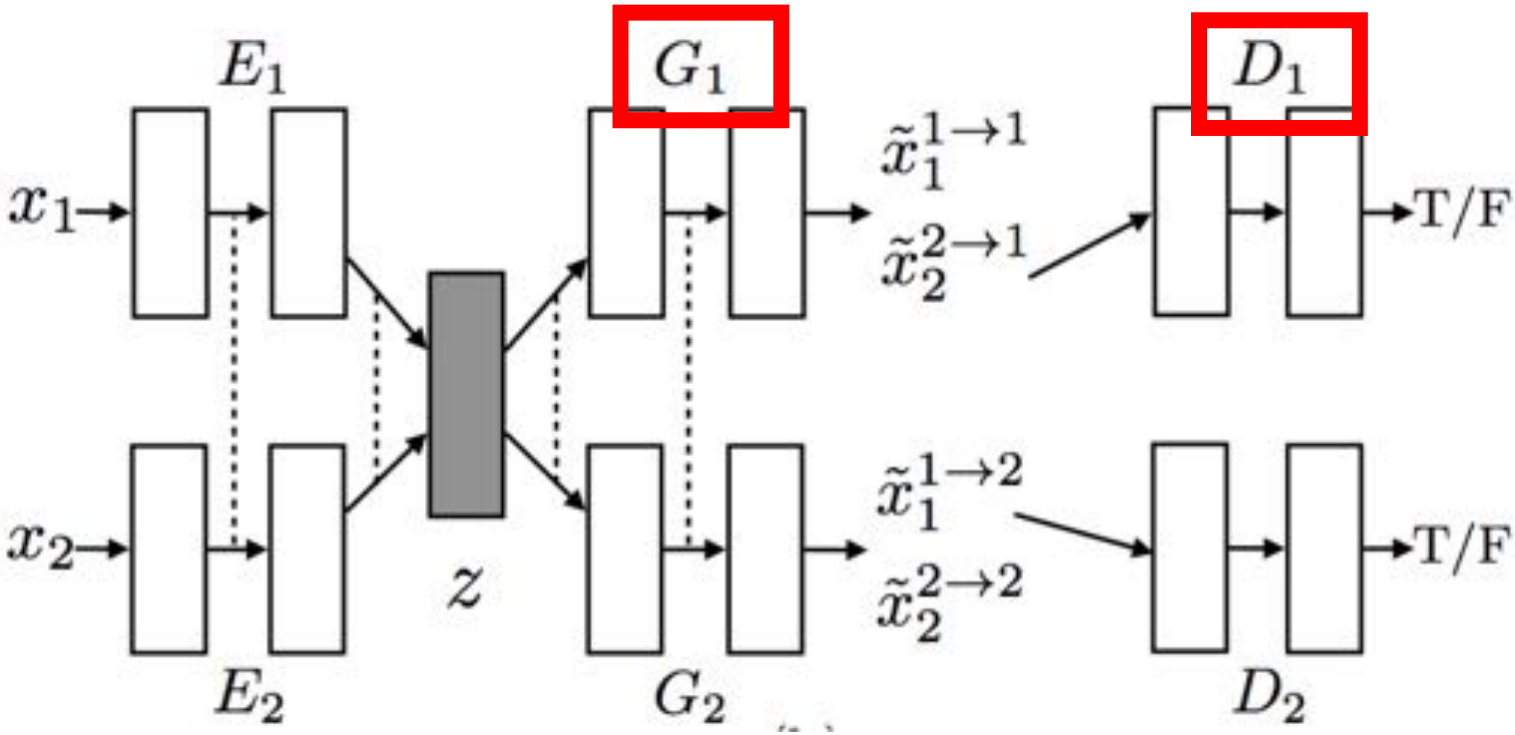


$$\tilde{x}_1^{1\to2} = G_2(z_1 \sim q_1(z_1|x_1))$$

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 $\mathcal{X}_1 \to \mathcal{X}_2$	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 $\mathcal{X}_1 \to \mathcal{X}_2$	GAN for $\mathcal{X}_1$	VAE-GAN [14]	CoGAN [17]

UNIT

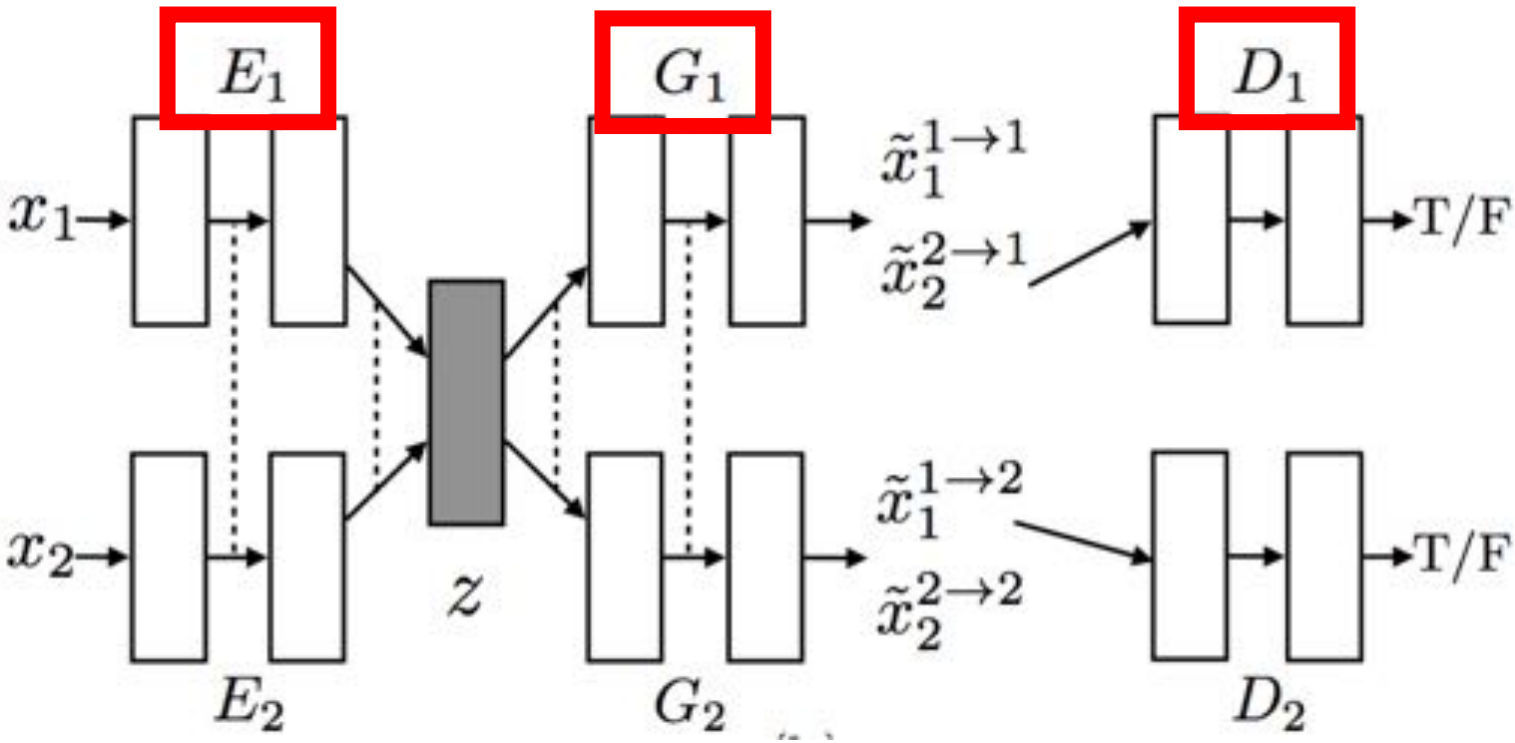


Table 1: Interpretation of the roles of the subnetworks in the proposed framework.

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 $\mathcal{X}_1 \to \mathcal{X}_2$	GAN for $\mathcal{X}_1$	VAE-GAN [14]	CoGAN [17]

UNIT

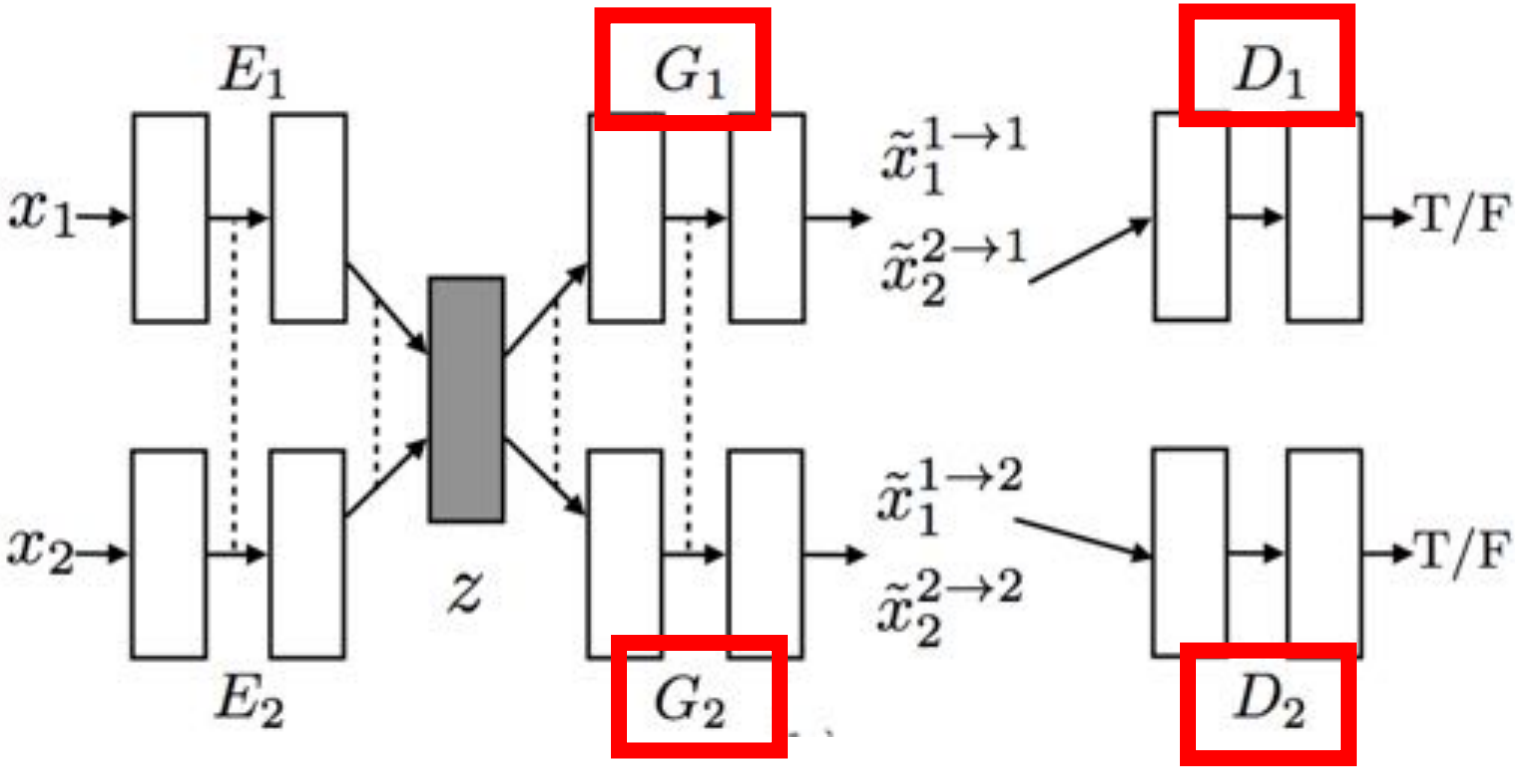


Table 1: Interpretation of the roles of the subnetworks in the proposed framework.

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 $\mathcal{X}_1 \to \mathcal{X}_2$	GAN for $\mathcal{X}_1$	VAE-GAN [14]	CoGAN [17]

Results

Results

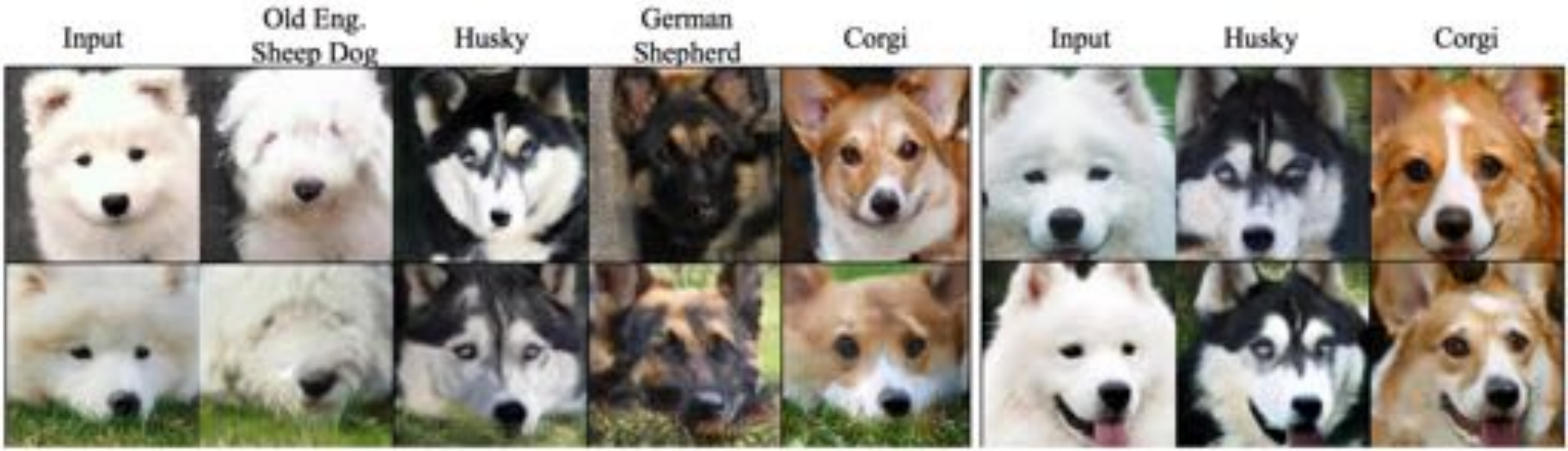


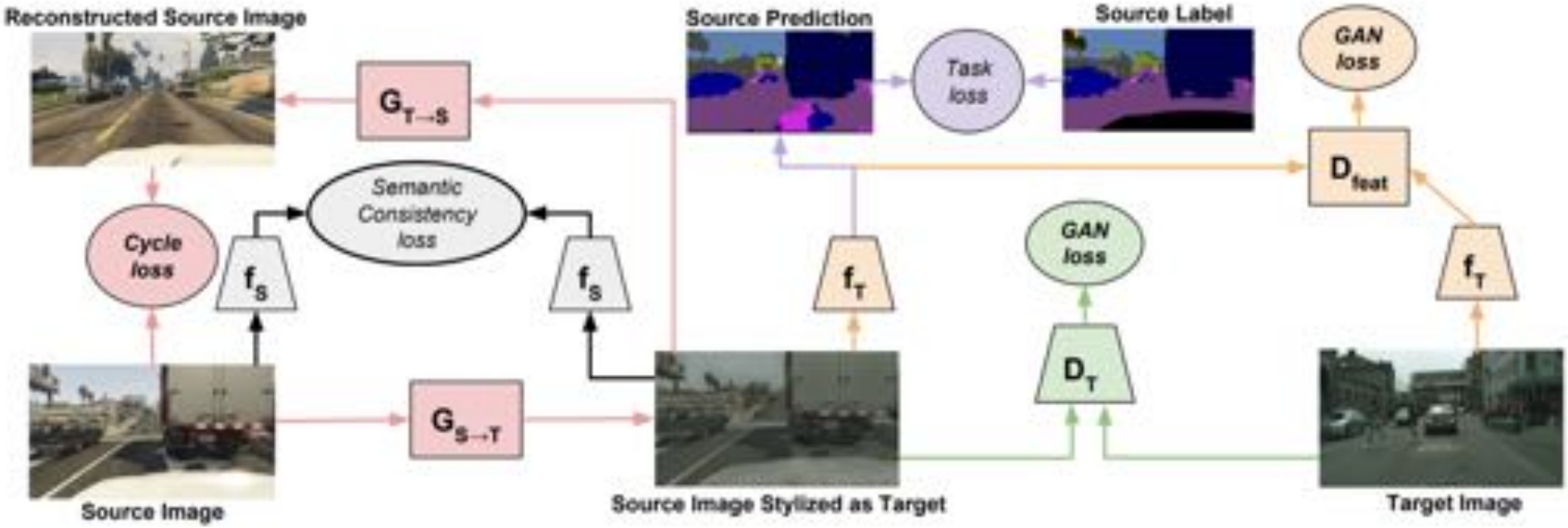
Figure 4: Dog breed translation results.

Results

Figure 5: Cat species translation results.

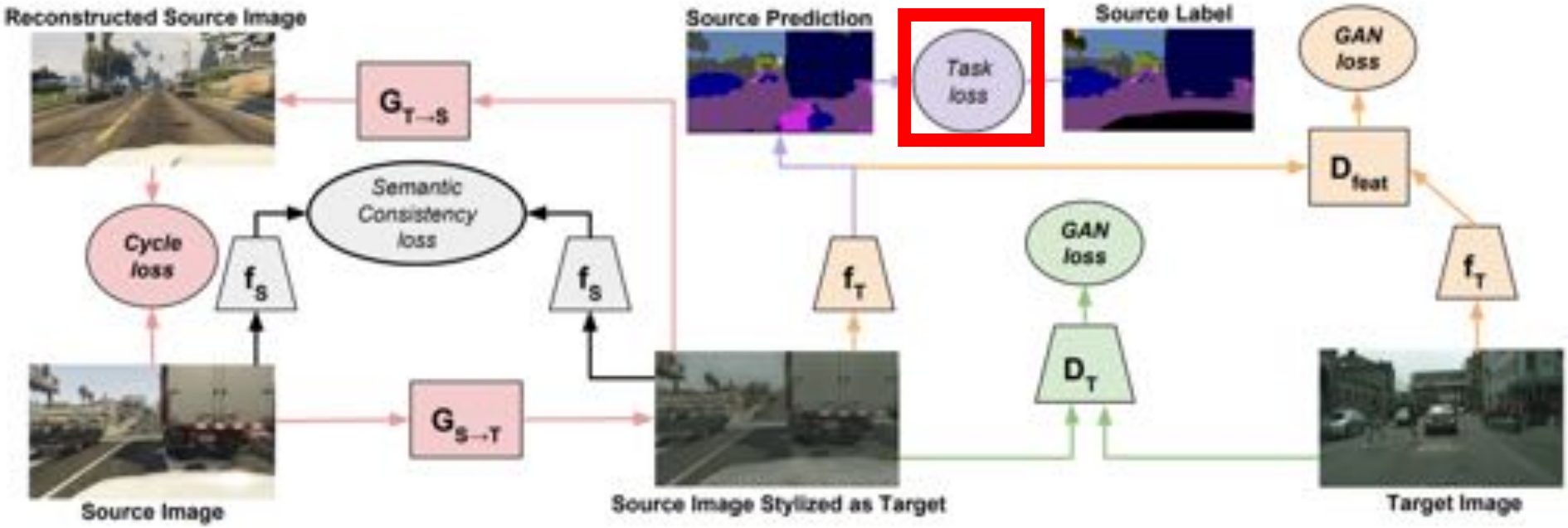
Figure 6: Attribute-based face translation results.

Application to Domain Adaptation



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

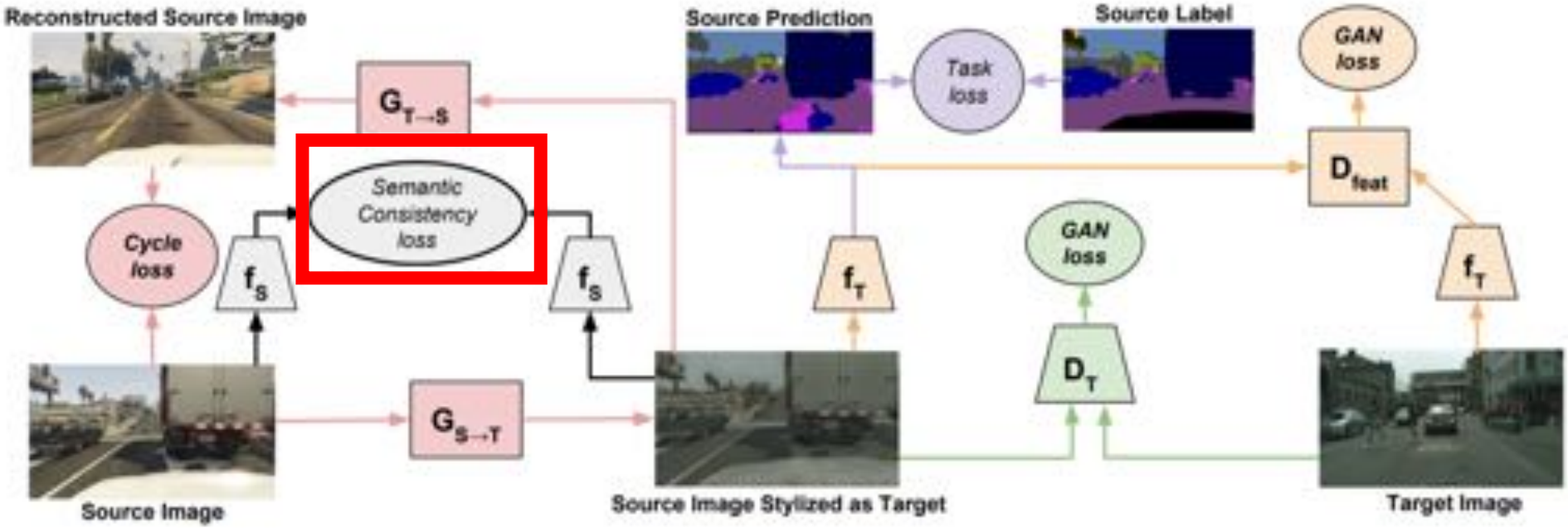
Application to Domain Adaptation



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

$$\mathcal{L}_{\text{sem}}(G_{S\to T}, G_{T\to S}, X_S, X_T, f_S) = \mathcal{L}_{\text{task}}(f_S, G_{T\to S}(X_T), p(f_S, X_T)) + \mathcal{L}_{\text{task}}(f_S, G_{S\to T}(X_S), p(f_S, X_S))$$

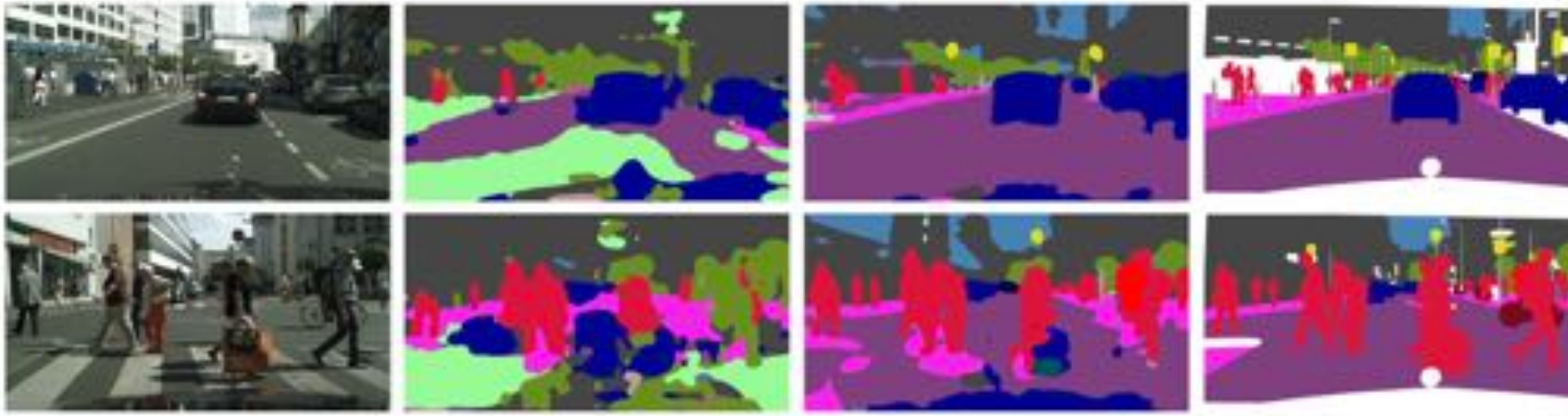
Application to Domain Adaptation



$$\mathcal{L}_{\text{sem}}(G_{S\to T}, G_{T\to S}, X_S, X_T, f_S) = \mathcal{L}_{\text{task}}(f_S, G_{T\to S}(X_T), p(f_S, X_T)) + \mathcal{L}_{\text{task}}(f_S, G_{S\to T}(X_S), p(f_S, X_S))$$

Results

(a) GTA5 (b) GTA5 → Cityscapes



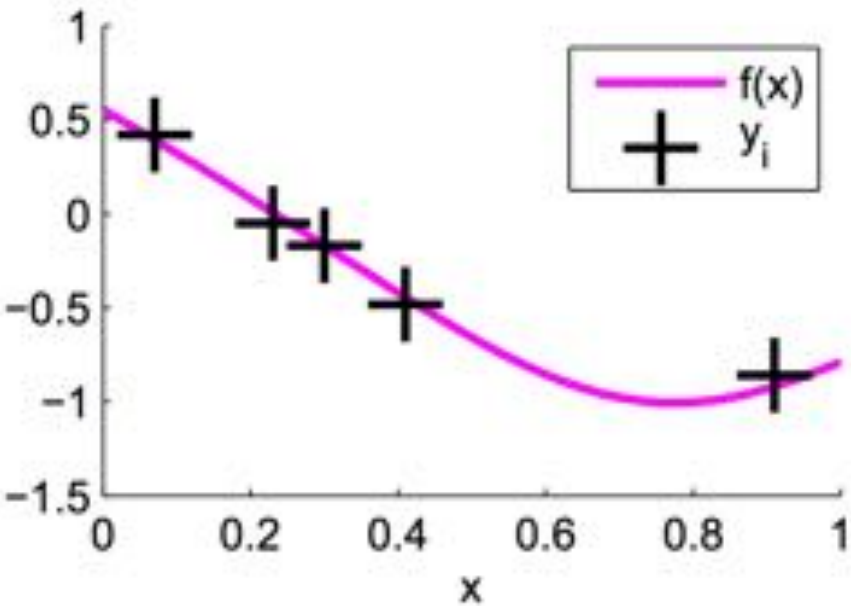
(a) Test Image (b) Source Prediction (c) CyCADA Prediction (d) Ground Truth

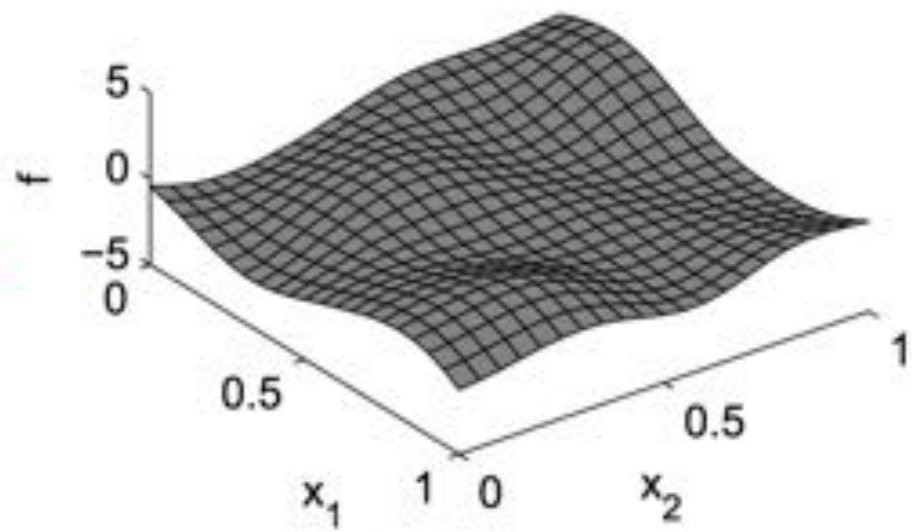
Gaussian Processes

Kayhan Batmanghelich

Goal: Learning a Function

Learn scalar function of vector values $f(\mathbf{x})$





We have (possibly noisy) observations $\{\mathbf{x}_i, y_i\}_{i=1}^n$

Slides: Iain Murray

Example Applications

Real-valued regression:

— Robotics: target state $\rightarrow$ required torque

— Process engineering: predicting yield

— Surrogate surfaces for optimization or simulation

Classification:

— Recognition: e.g. handwritten digits on cheques

— Filtering: fraud, interesting science, disease screening

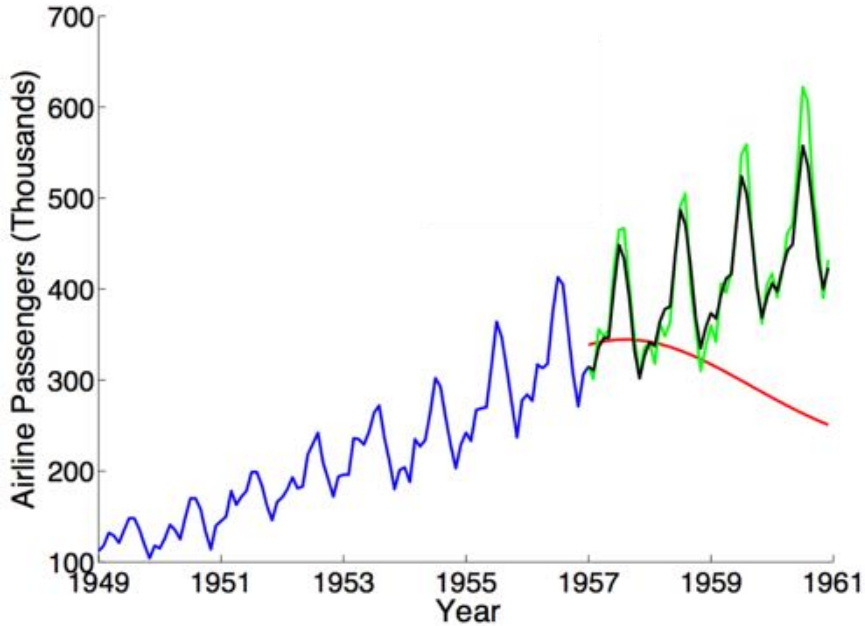
Ordinal regression:

— User ratings (e.g. movies or restaurants)

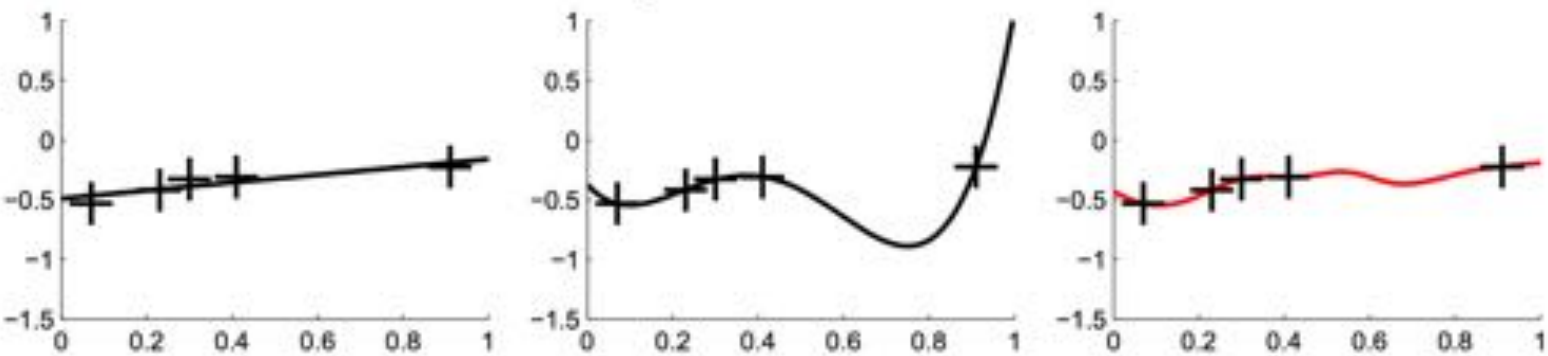
— Disease screening (e.g. predicting Gleason score)

A Regression Example

Predict the future:



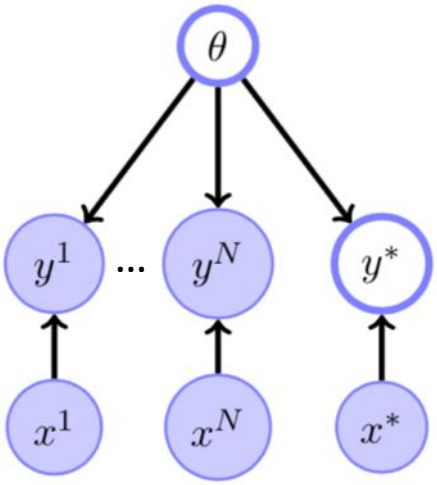
The world is often complicated:

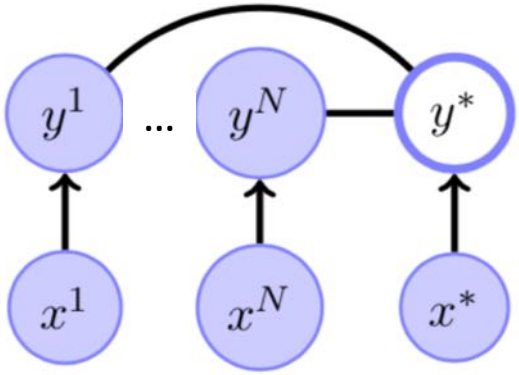


simple fit complex fit truth

Non-Parametric Approach

Let's revisit prediction (at test time):





$$p(y^*|x^*,\mathcal{D}) \propto p(y^*,x^*,\mathcal{D}) = \int_\theta p(y^*,\mathcal{Y},x^*,\mathcal{X},\theta)$$

$$\propto \int_\theta p(y^*|x^*,\theta)p(\theta)\prod_n p(y^n|\theta,x^n) \propto \int_\theta p(y^*|x^*,\theta)p(\theta|\mathcal{D})$$

Non-parametric methods directly model the joint conditional distribution!

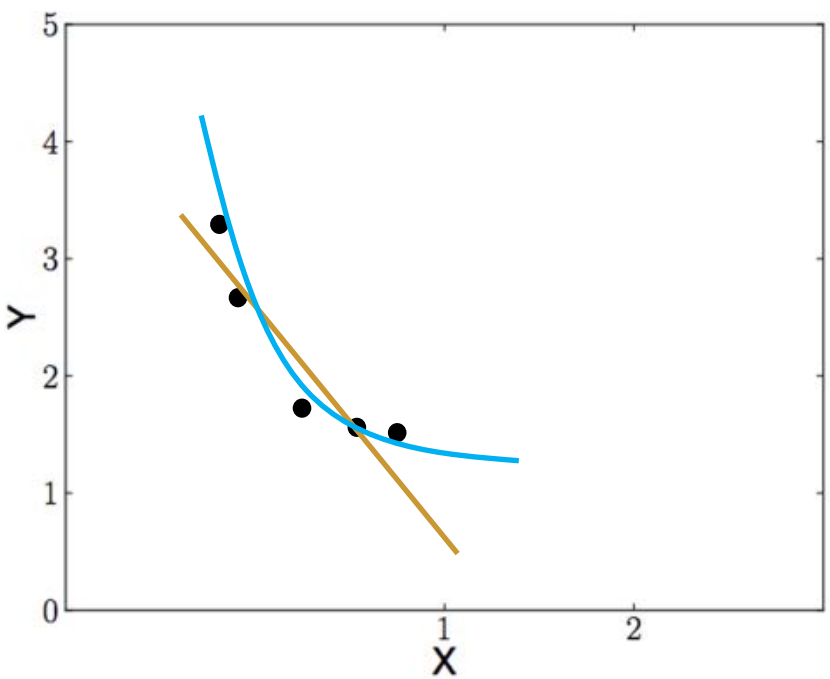
$$\boxed{p(y^*,\mathcal{Y}|x^*,\mathcal{X})} = \int_\theta p(y^*|x^*,\theta)p(\theta)\prod_n p(y^n|\theta,x^n)$$

Revisiting Linear Regression

Linear Regression with no noise:

$$y = \mathbf{w}^T \boxed{\boldsymbol{\phi}(\mathbf{x})} \qquad p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \boldsymbol{\Sigma}_{\mathbf{w}})$$

Feature vector



Example 1:

$$\boldsymbol{\phi}(\mathbf{x}) = \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}$$

Example 2:

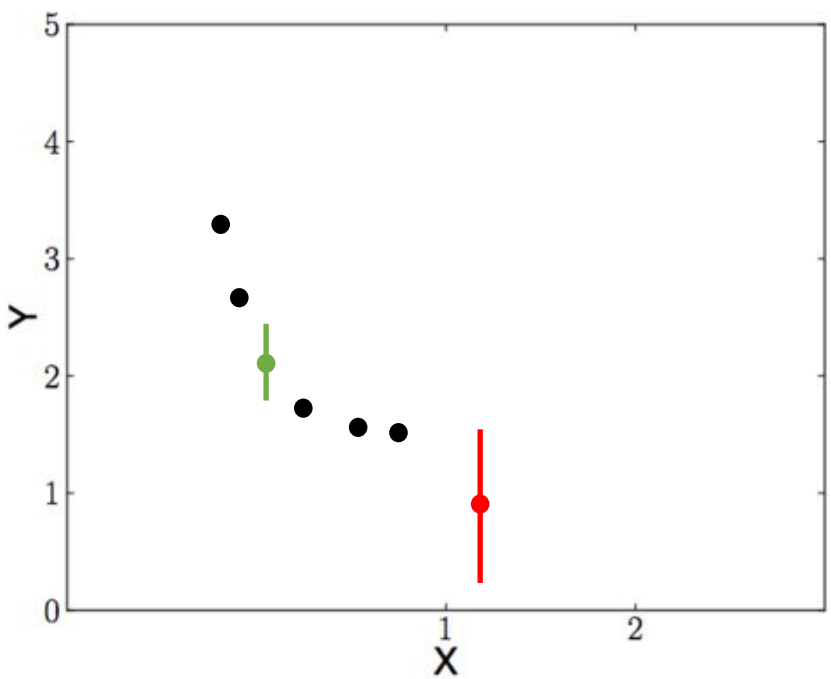
$$\boldsymbol{\phi}(\mathbf{x}) = \begin{bmatrix} \mathbf{x}^2 \\ \mathbf{x} \\ 1 \end{bmatrix}$$

Revisiting Linear Regression

Linear Regression with no noise:

$$y = \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}) \qquad p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \boldsymbol{\Sigma}_\mathbf{w})$$

Feature vector



- In many applications, one relies on generic **smoothness** assumptions:
  - If two inputs x and x' then outputs (y and y') should be similar.

Revisiting Linear Regression

Linear Regression with no noise:

$$y = \mathbf{w}^T \boxed{\phi(\mathbf{x})} \qquad p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \boldsymbol{\Sigma}_\mathbf{w})$$

Feature vector

Let's integrate *w* out to get $p(y^*, \mathcal{Y}|x^*, \mathcal{X})$:

$$\mathbf{y} = \left[y^1, \cdots, y^N\right] \qquad \boldsymbol{\Phi} = \left[\phi(x^1), \ldots, \phi(x^N)\right]^\mathsf{T}$$

$$p(\mathbf{y}|\mathbf{x}) = \mathcal{N}(\mathbf{y}|0, \boldsymbol{\Phi}\boldsymbol{\Sigma}_\mathbf{w}\boldsymbol{\Phi}^\mathsf{T}) = \mathcal{N}(\mathbf{y}|\mathbf{0}, \mathbf{K})$$

Joint conditional distribution

$$[\mathbf{K}]_{n,n'} = \phi(x^n)^\mathsf{T}\phi(x^{n'}) = k(x^n, x^{n'})$$

covariance function

What if instead of $\phi(x)$, we specify $k(x', x)$ directly?

Covariance Function

- The **covariance function** controls the behavior of the prediction function ($x \to y$) implicitly.

- Examples of $k(x, x')$:

$$v_0 \exp\left\{-\frac{1}{2}\sum_{l=1}^{D}\lambda_l\left(x_l - x_l'\right)^2\right\}$$

Squared Exponential

$$\mathbf{x}^\mathsf{T}\mathbf{x}'$$

Linear

$$\arcsin\left(\frac{2\mathbf{x}^\mathsf{T}\mathbf{x}'}{\sqrt{1+2\mathbf{x}^\mathsf{T}\mathbf{x}}\sqrt{1+2\mathbf{x}'^\mathsf{T}\mathbf{x}'}}\right)$$

So-called Neural Network

Stationary

Non-Stationary

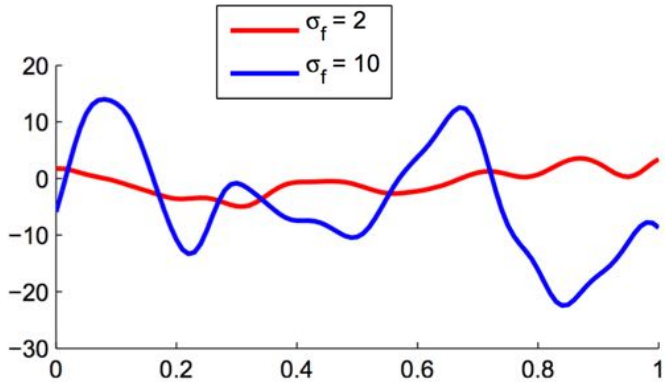
Covariance Function

- The **covariance function** controls the behavior of the prediction function ($x \to y$) implicitly.

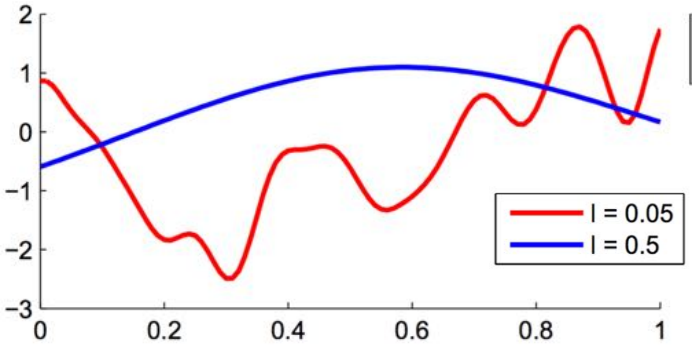
- The $k(x, x')$ specifies the prior over functions:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 \exp\left(-\frac{1}{2} \sum_{d=1}^{D} (x_{d,i} - x_{d,j})^2 / \ell_d^2 \right)$$

Consider $\mathbf{x}_i = \mathbf{x}_j$, $\Rightarrow$ marginal function variance is σ_f^2



Typical distance between peaks $\approx \ell$



Covariance Function

- The **covariance function** controls the behavior of the prediction function ($x \to y$) implicitly.
- The $k(x, x')$ specifies the prior over functions.
- Potentially, the $\phi(x)$ function can be infinite dimensional.
- $k(x, x')$ should be positive definite (Mercer Theorem):

$$\int_{\mathcal{X}^2} k(x, x') f(x) f(x') \, d\mu(x) d\mu(x') \geq 0$$

Covariance Function

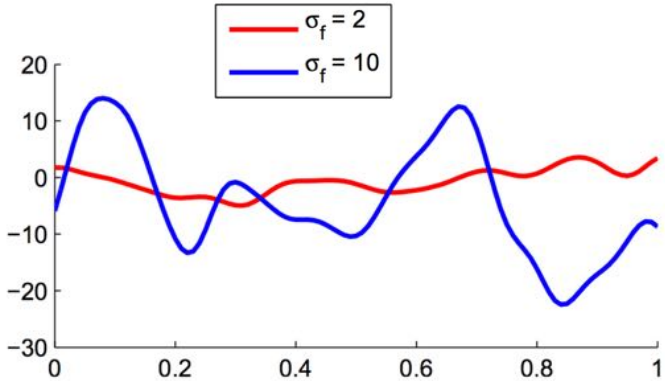
- The **covariance function** controls the behavior of the prediction function ($x \to y$) implicitly.
- The $k(x, x')$ specifies the prior over functions.
- Potentially, the $\phi(x)$ function can be infinite dimensional.
- $k(x, x')$ should be positive definite (Mercer Theorem).
- The covariance function (kernel) can be defined between many different objects.

Gaussian Process

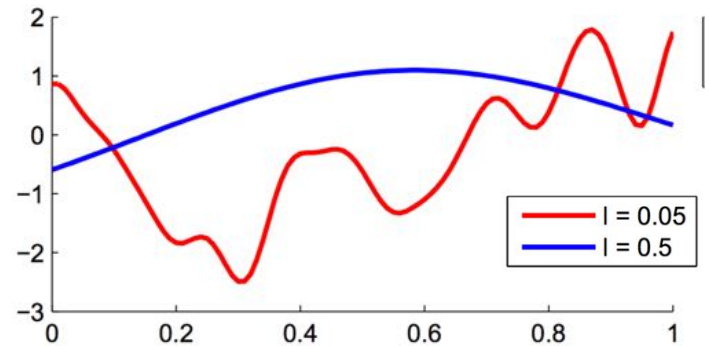
- The Gaussian process (GP) as a prior on **functions**.

$$k(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 \exp\left(-\frac{1}{2} \sum_{d=1}^{D} (x_{d,i} - x_{d,j})^2 / \ell_d^2 \right)$$

Consider $\mathbf{x}_i = \mathbf{x}_j$, $\Rightarrow$ marginal function variance is σ_f^2



Typical distance between peaks $\approx \ell$



- *Covariance function* and hyperparameters reflect the prior belief on function smoothness, length scales etc.

Gaussian Process

- The Gaussian process (GP) as a prior on **functions**.
- *Covariance function* and hyperparameters reflect the prior belief on function smoothness, length scales etc.
- A GP is a collection of random variables, any **finite number** of which have a joint **Gaussian** distribution.

Infinite dimension:

$$f(x) \sim \mathcal{GP}(m, k)$$

Finite dimension:

$$[f(x_1), \ldots, f(x_N)] \sim \mathcal{N}(\boldsymbol{\mu}, K)$$

$$\boldsymbol{\mu}_i = m(x_i)$$

$$K_{ij} = k(x_i, x_j)\,,$$

Gaussian Process

- The Gaussian process (GP) as a prior on **functions**.
- *Covariance function* and hyperparameters reflect the prior belief on function smoothness, length scales etc.
- A GP is a collection of random variables, any **finite number** of which have a joint **Gaussian** distribution.
- Easy inference for the regression.

Inference in GP

Given observed noisy data $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\}$, the joint probability over latent function values $\mathbf{f}$ and $\mathbf{f}^\star$ given $\mathbf{y}$ is

$$p([\mathbf{f}, \mathbf{f}^\star] \,|\, \mathbf{X}, \mathbf{X}^\star, \mathbf{y}, \boldsymbol{\theta}_\mathsf{K}, \sigma^2) \propto \overbrace{\mathcal{N}\left([\mathbf{f}, \mathbf{f}^\star] \,\middle|\, \mathbf{0}, \begin{bmatrix} \mathbf{K}_{\mathbf{X},\mathbf{X}} & \mathbf{K}_{\mathbf{X},\mathbf{X}^\star} \\ \mathbf{K}_{\mathbf{X}^\star,\mathbf{X}} & \mathbf{K}_{\mathbf{X}^\star,\mathbf{X}^\star} \end{bmatrix}\right)}^{\text{Prior}} \times \underbrace{\prod_{n=1}^{N} \mathcal{N}\left(y_n \,|\, f_n, \sigma^2\right)}_{\text{Likelihood}},$$

Inference in GP

Given observed noisy data $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\}$, the joint probability over latent function values $\mathbf{f}$ and $\mathbf{f}^\star$ given $\mathbf{y}$ is

$$p([\mathbf{f}, \mathbf{f}^\star] \,|\, \mathbf{X}, \mathbf{X}^\star, \mathbf{y}, \boldsymbol{\theta}_\mathsf{K}, \sigma^2) \propto \overbrace{\mathcal{N}\left([\mathbf{y}, \mathbf{f}^\star] \,\middle|\, \mathbf{0}, \begin{bmatrix} \mathbf{K}_{\mathbf{X},\mathbf{X}} + \sigma^2\mathbf{I} & \mathbf{K}_{\mathbf{X},\mathbf{X}^\star} \\ \mathbf{K}_{\mathbf{X}^\star,\mathbf{X}} & \mathbf{K}_{\mathbf{X}^\star,\mathbf{X}^\star} \end{bmatrix}\right)}^{\text{Prior}} \times \underbrace{\prod_{n=1}^{N} \mathcal{N}\left(y_n \,|\, f_n, \sigma^2\right)}_{\text{Likelihood}},$$

Regression Case:

$$p(\mathbf{f}^\star \,|\, \mathbf{X}, \mathbf{y}, \mathbf{X}^\star, \boldsymbol{\theta}_\mathsf{K}, \sigma^2) = \mathcal{N}\left(\mathbf{f}^\star \,|\, \boldsymbol{\mu}^\star, \boldsymbol{\Sigma}^\star\right) \text{ with}$$

$$\boldsymbol{\mu}^\star = \mathbf{K}_{\mathbf{X}^\star,\mathbf{X}} \left[\mathbf{K}_{\mathbf{X},\mathbf{X}} + \sigma^2\mathbf{I}\right]^{-1} \mathbf{y}$$

$$\boldsymbol{\Sigma}^\star = \mathbf{K}_{\mathbf{X}^\star,\mathbf{X}^\star} - \mathbf{K}_{\mathbf{X}^\star,\mathbf{X}} \left[\mathbf{K}_{\mathbf{X},\mathbf{X}} + \sigma^2\mathbf{I}\right]^{-1} \mathbf{K}_{\mathbf{X},\mathbf{X}^\star}$$

Inference in GP

Given observed noisy data $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\}$, the joint probability over latent function values $\mathbf{f}$ and $\mathbf{f}^\star$ given $\mathbf{y}$ is

$$p([\mathbf{f}, \mathbf{f}^\star] \,|\, \mathbf{X}, \mathbf{X}^\star, \mathbf{y}, \boldsymbol{\theta}_\mathsf{K}, \sigma^2) \propto \overbrace{\mathcal{N}\left([\mathbf{f}, \mathbf{f}^\star] \,\middle|\, \mathbf{0}, \begin{bmatrix} \mathbf{K}_{\mathbf{X},\mathbf{X}} & \mathbf{K}_{\mathbf{X},\mathbf{X}^\star} \\ \mathbf{K}_{\mathbf{X}^\star,\mathbf{X}} & \mathbf{K}_{\mathbf{X}^\star,\mathbf{X}^\star} \end{bmatrix}\right)}^{\text{Prior}} \times \underbrace{\prod_{n=1}^{N} \mathcal{N}\left(y_n \,|\, f_n, \sigma^2\right)}_{\text{Likelihood}},$$

Classification Case:

$$p(y_n = 1 \,|\, f_n) = \frac{1}{1 + exp(-f_n)}$$

$$KL\Big[\overbrace{\mathcal{N}\left(\mathbf{f} \,|\, \mathbf{0}, \mathbf{K}_{\mathbf{X},\mathbf{X}}(\boldsymbol{\theta}_\mathsf{K})\right)}^{\text{Prior}} \times \overbrace{\prod_{n=1}^{N} p(y_n \,|\, f_n)}^{\text{exact likelihood}} \,\Big\|\, \underbrace{\mathcal{N}\left(\mathbf{f} \,|\, \mathbf{0}, \mathbf{K}_{\mathbf{X},\mathbf{X}}(\boldsymbol{\theta}_\mathsf{K})\right)}_{\text{Prior}} \times \underbrace{\prod_{n=1}^{N} \mathcal{N}\left(f_n \,|\, \tilde{\mu}_n, \tilde{\sigma}_n\right)}_{\text{approximation}}\Big]$$