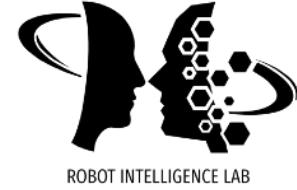


# Deep Learning and Applications

Generative Models 2

Sungjoon Choi, Korea University



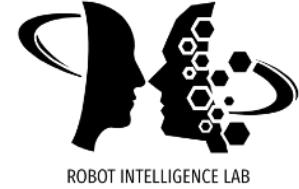
# Latent Variable Models

# Question



Is an **auto-encoder** a generative model?

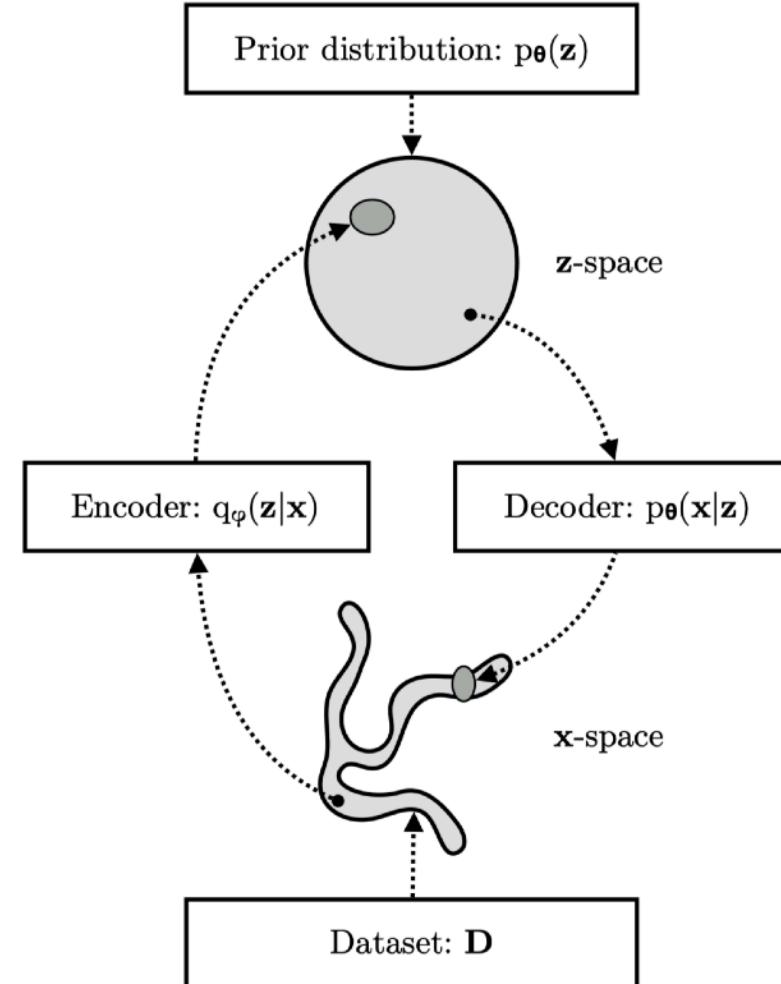
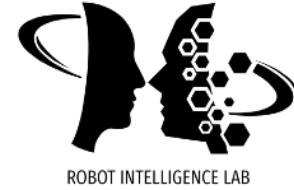
# Variational Auto-Encoder



- The objective is simple.

Maximize  $p_{\theta}(\mathbf{x})$

# Variational Auto-Encoder



# Variational Auto-Encoder



$$\begin{aligned}
 \underbrace{\log p_{\theta}(x)}_{\text{Maximum Likelihood Learning } \uparrow} &= \int_z \underbrace{q_{\phi}(z|x)}_{\text{For any } q_{\phi}} \log p_{\theta}(x) dz \\
 &= \mathbb{E}_{z \sim q_{\phi}(z|x)} \left[ \log \frac{p_{\theta}(x)p_{\theta}(z|x)}{p_{\theta}(z|x)} \right] \\
 &= \mathbb{E}_{z \sim q_{\phi}(z|x)} \left[ \log \frac{p_{\theta}(x, z)}{p_{\theta}(z|x)} \right] \\
 &= \mathbb{E}_{z \sim q_{\phi}(z|x)} \left[ \log \frac{p_{\theta}(x, z)q_{\phi}(z|x)}{p_{\theta}(z|x)q_{\phi}(z|x)} \right] \\
 &= \mathbb{E}_{z \sim q_{\phi}(z|x)} \left[ \log \frac{p_{\theta}(x, z)}{q_{\phi}(z|x)} \right] + \mathbb{E}_{z \sim q_{\phi}(z|x)} \left[ \log \frac{q_{\phi}(z|x)}{p_{\theta}(z|x)} \right] \\
 &= \underbrace{\mathbb{E}_{z \sim q_{\phi}(z|x)} \left[ \log \frac{p_{\theta}(x, z)}{q_{\phi}(z|x)} \right]}_{\text{ELBO } \uparrow} + \underbrace{D_{KL} (q_{\phi}(z|x) \| p_{\theta}(z|x))}_{\text{Variational Gap } \downarrow}
 \end{aligned}$$

# Variational Auto-Encoder



$$\underbrace{\mathbb{E}_{z \sim q_\phi(z|x)} \left[ \log \frac{p_\theta(x, z)}{q_\phi(z|x)} \right]}_{\text{ELBO } \uparrow} = \int \log \frac{p_\theta(x|z)p(z)}{q_\phi(z|x)} q_\phi(z|x) dz$$
$$= \underbrace{\mathbb{E}_{q_\phi(z|x)} [p_\theta(x|z)]}_{\text{Reconstruction Term}} - \underbrace{D_{KL} (q_\phi(z|x) \| p(z))}_{\text{Prior Fitting Term}}$$

# Variational Auto-Encoder

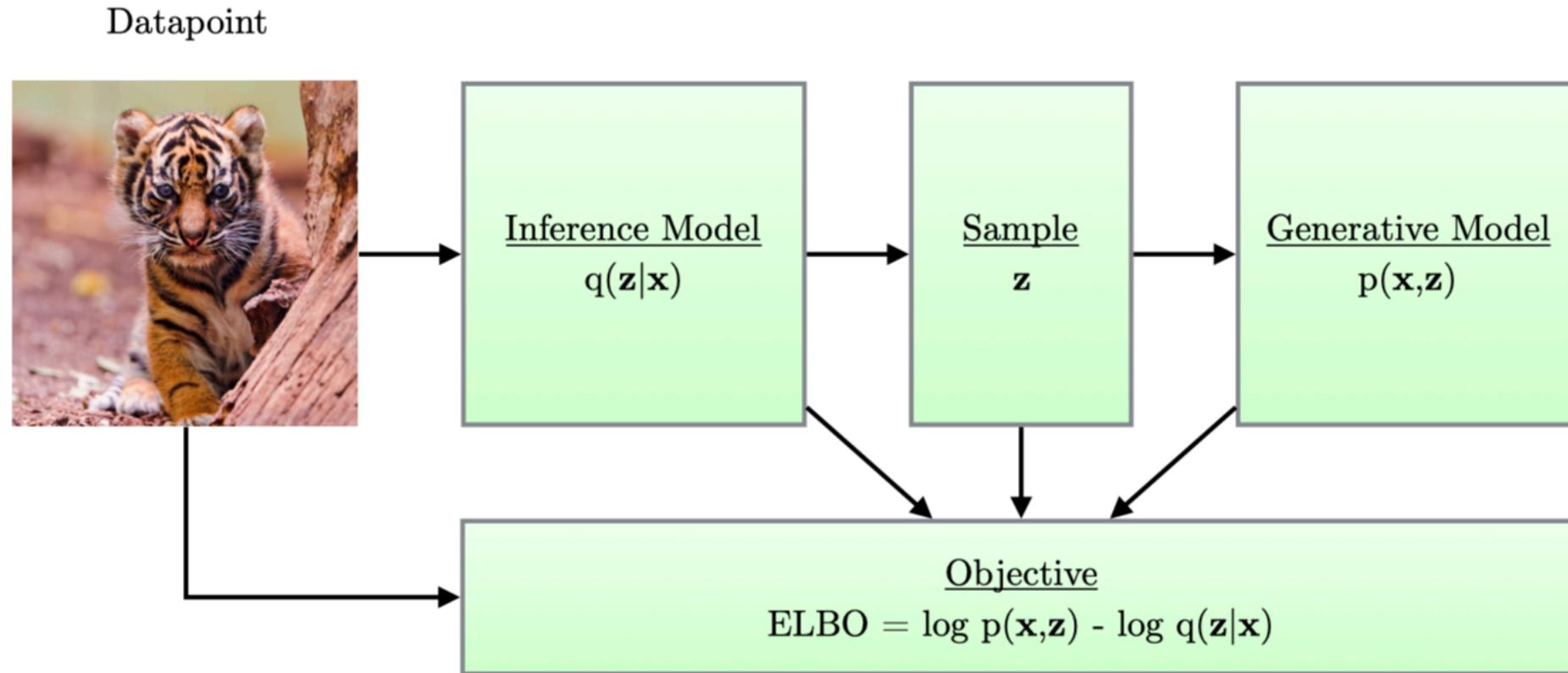
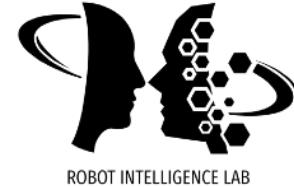
$$\begin{aligned}
 \underbrace{\mathbb{E}_{z \sim q_\phi(z|x)} \left[ \log \frac{p_\theta(x, z)}{q_\phi(z|x)} \right]}_{\text{ELBO } \uparrow} &= \int \log \frac{p_\theta(x|z)p(z)}{q_\phi(z|x)} q_\phi(z|x) dz \\
 &= \underbrace{\mathbb{E}_{q_\phi(z|x)} [p_\theta(x|z)]}_{\text{Reconstruction Term}} - \underbrace{D_{KL} (q_\phi(z|x) \| p(z))}_{\text{Prior Fitting Term}}
 \end{aligned}$$

- Key Limitation

- It is an **intractable** model (hard to evaluate likelihood).
- The prior fitting term should be differentiable, hence it is hard to use diverse latent prior distributions.
- In most cases, we use an isotropic Gaussian where we have a closed-form for the prior fitting term.

$$\underbrace{D_{KL} (q_\phi(z|x) \| \mathcal{N}(0, I))}_{\text{Prior Fitting Term}} = \frac{1}{2} \sum_{i=1}^D (\sigma_{z_i}^2 + \mu_{z_i}^2 - \log(\sigma_{z_i}^2) - 1)$$

# Variational Auto-Encoder



# Reparametrization Trick

- Want to compute a gradient with respect to  $\phi$  of

$$E_{q(z;\phi)}[r(z)] = \int q(z; \phi) r(z) dz$$

where  $z$  is continuous.

- Suppose  $q(z; \phi) = \mathcal{N}(\mu, \sigma^2)$  is Gaussian with parameters  $\phi = (\mu, \sigma)$ . These are equivalent ways of sampling:

- Sample  $z \sim q_\phi(z)$
- Sample  $\epsilon \sim \mathcal{N}(0,1)$ , then  $z = \mu + \sigma\epsilon = g(\epsilon; \phi)$

- Using this equivalence, we compute the expectation in two ways:

$$E_{z \sim q(z;\phi)}[r(z)] = E_{\epsilon \sim \mathcal{N}(0,1)}[r(g(\epsilon; \phi))] = \int p(\epsilon) r(\mu + \sigma\epsilon) d\epsilon$$

$$\nabla_\phi E_{q(z;\phi)}[r(z)] = \nabla_\phi E_\epsilon[r(g(\epsilon; \phi))] = E_\epsilon[\nabla_\phi r(g(\epsilon; \phi))]$$

- Easy to approximate the gradient by

$$E_\epsilon[\nabla_\phi r(g(\epsilon; \phi))] \approx \frac{1}{k} \sum_k \nabla_\phi r(g(\epsilon^k; \phi)) \text{ where } \epsilon^k \sim \mathcal{N}(0,1)$$

# Amortizaiton

$$\max_{\theta} l(\theta; \mathcal{D}) \geq \max_{\theta, \phi^1, \phi^2, \dots} \sum_{x^i \in \mathcal{D}} \mathcal{L}(x^i; \theta, \phi^i)$$

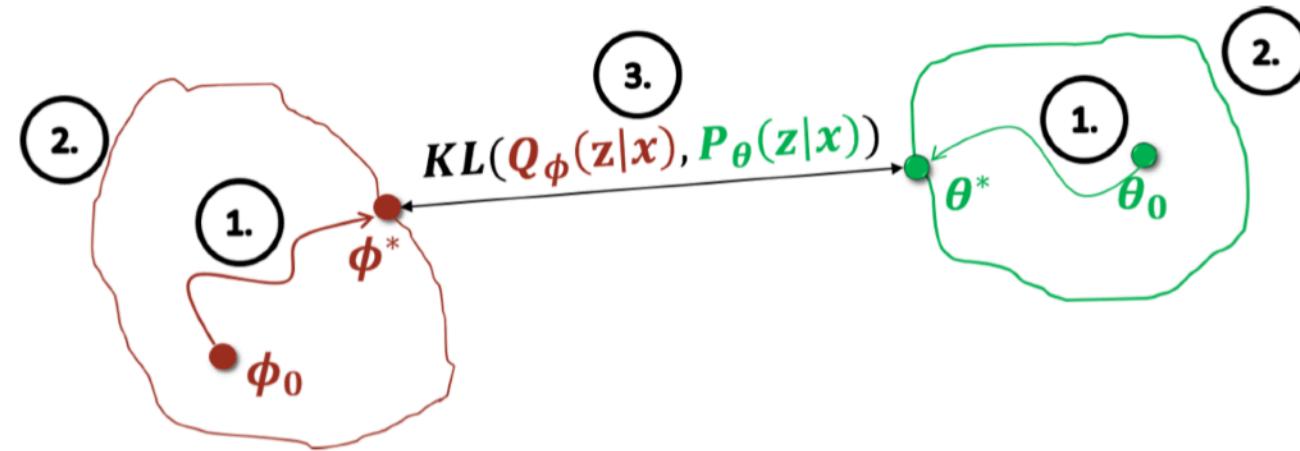
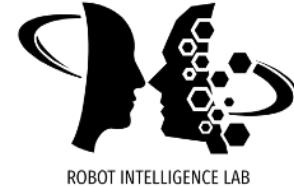
- So far, we have used a set of variational parameters (e.g.,  $\mu$  and  $\sigma$ )  $\phi^i$  for **each** data point  $x^i$ . However it does not scale to large datasets.
- **Amortization**: Now we learn a **single** parametric function  $f_\lambda$  that maps each  $x$  to a set of (good) variational parameters (i.e.,  $f_\lambda : x^i \mapsto \phi^i$ )
  - For example, if  $q(z|x^i)$  are Gaussians with different means and variances, we learn a single neural network  $f_\lambda$  mapping  $x^i$  to  $(\mu^i, \sigma^i)$ .
  - This is an encoder network in VAE.
  - We approximate the posteriors  $q(z|x^i)$  using the distribution  $q_\lambda(z|x)$ .

# Summary of Latent Variable Models



1. Combine simple models to get a more flexible one (e.g., mixture of Gaussians)
2. Directed model permits ancestral sampling (efficient generation):
  1.  $z \sim p(z)$
  2.  $x \sim p(x|z; \theta)$
3. However, log-likelihood is generally intractable, hence learning is difficult.
4. Joint learning of a (decoder) model ( $\theta$ ) and an amortized inference component ( $\phi$ ) to achieve tractability via ELBO optimization
5. Latent representation for any  $x$  can be inferred via  $q_\phi(z|x) = q(z; f_\lambda(x))$  (aka an encoder model).

# Research Directions



1. Better optimization techniques
2. More expressive approximating (variational) families
3. Alternate loss functions

# Adversarial Auto-encoder

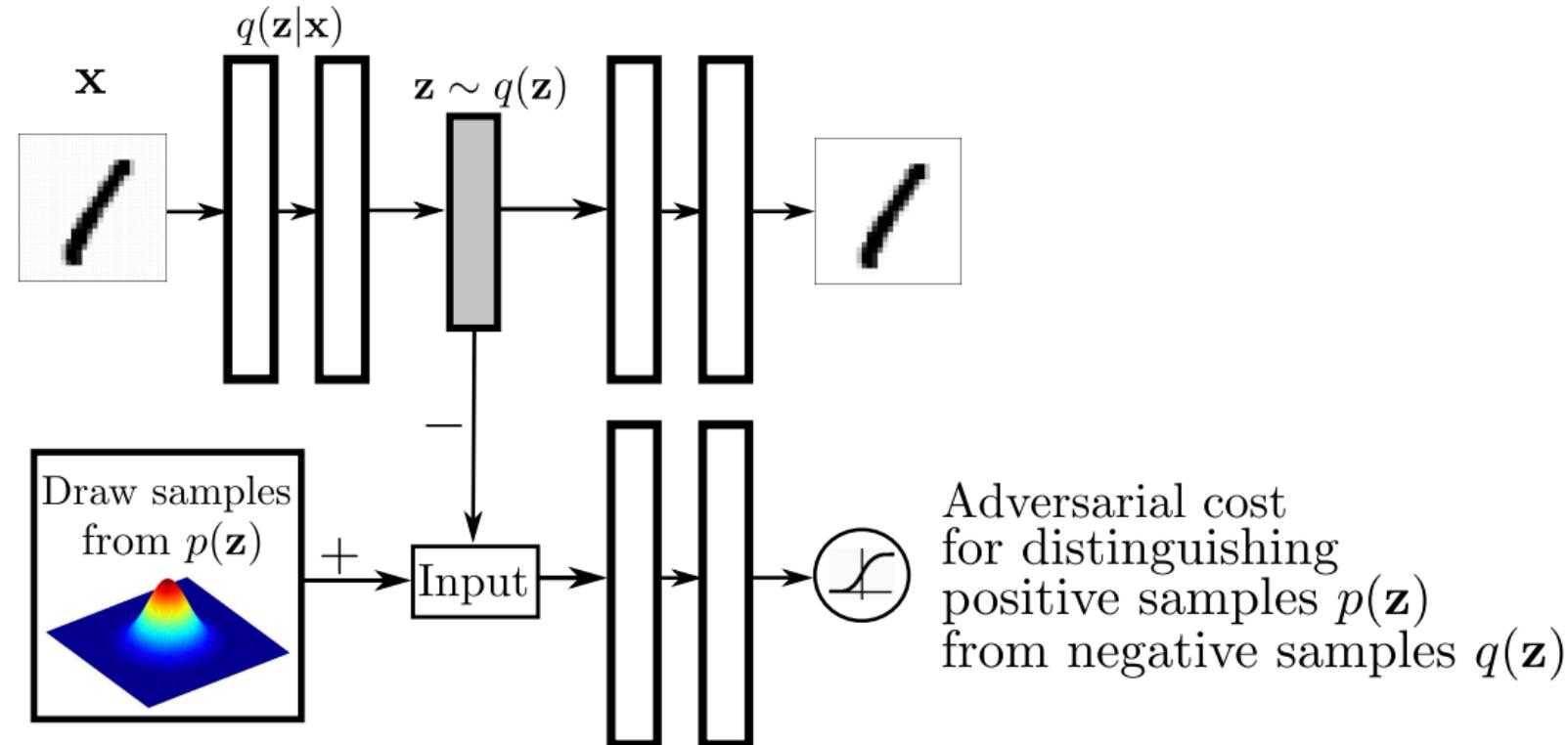


- Let  $x$  be the input and  $z$  be the latent vector.
- Let  $p(z)$  be the prior distribution we want to impose on the latent vectors,  $\underline{q(z|x)}$  be an encoding distribution, and  $p(x|z)$  be the decoding distribution.
- Also let  $p_d(x)$  be the data distribution, and  $p(x)$  be the model distribution.
- The **aggregated posterior distribution  $q(z)$**  is defined by the encoder function  $q(z|x)$ :

$$q(z) = \int_x q(z|x)p_d(x)dx$$

- The adversarial encoder (AAE) is an autoencoder that is **regularized by matching** the aggregated posterior,  $q(z)$ , to an arbitrary prior,  $p(z)$ .
- To achieve this, an adversarial network is attached on top of the latent vector of the autoencoder that guides  $q(z)$  to match  $p(z)$ .

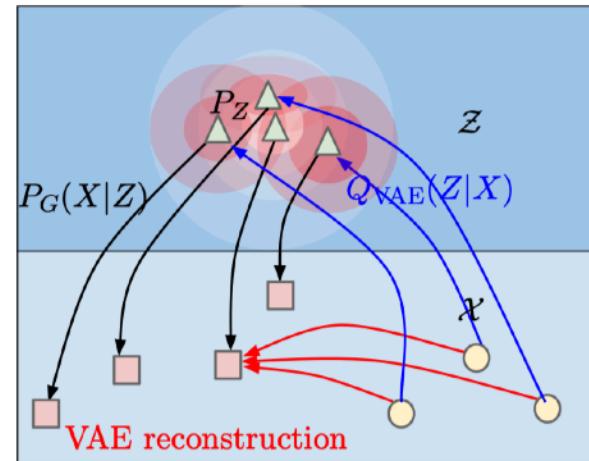
# Adversarial Auto-encoder



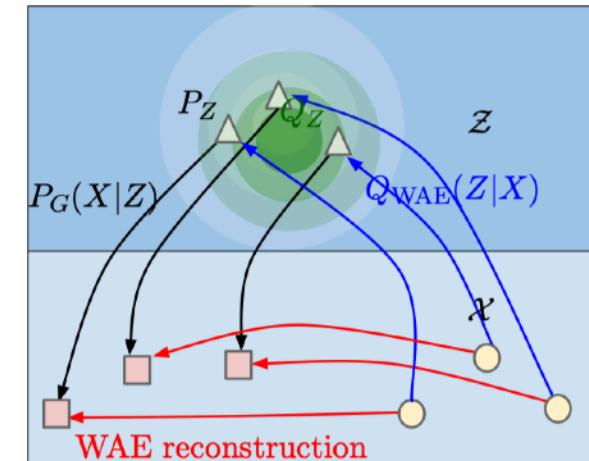
- AAE allows to use any arbitrary latent prior distributions that we can sample.

# Wasserstein Auto-encoders

(a) VAE



(b) WAE



- Both **VAE** and **WAE** minimize two terms: the reconstruction cost and the regularizer penalizing discrepancy between  $P_Z$  and distribution induced by the encoder  $Q$ .
- VAE forces  $Q(Z|X = x)$  to match  $P_Z$  for all the different input examples  $x$  drawn from  $P_X$ .
- This is illustrated on picture (a), where every single red ball is forced to match  $P_Z$  depicted as the white shape. Red balls start **intersecting**, which leads to problems with reconstruction.
- In contrast, WAE forces the continuous mixture  $Q_Z = \int Q(Z|X)dP_X$  to match  $P_Z$ , as depicted with the green ball in picture (b). As a result, latent vectors of different examples get a chance to **stay far way** from each other, promoting a better reconstruction.

# Wasserstein Auto-encoders



- Optimal transport and its dual formulations
  - A rich class of divergences between probability distributions is induced by the optimal transport (OT) problem.

$$W_c(P_X, P_G) = \inf_{\Gamma \in \mathcal{P}(X \sim P_X, Y \sim P_G)} E_{(X,Y) \sim \Gamma}[c(X, Y)]$$

where  $c(x, y) : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}_+$  is any measurable cost function and  $\mathcal{P}(X \sim P_X, Y \sim P_G)$  is a set of all joint distributions of  $(X, Y)$  with marginals  $P_X$  and  $P_G$ , respectively.

- A particularly interesting case is when  $(\mathcal{X}, d)$  is a metric space and  $c(x, y) = d^p(x, y)$  for  $p \geq 1$ . In this case,  $W_c$  is called the  $p$ -Wasserstein distance.

# Wasserstein Auto-encoders

**Theorem 1.** For  $P_G$  as defined above with deterministic  $P_G(X|Z)$  and any function  $G: \mathcal{Z} \rightarrow \mathcal{X}$

$$\inf_{\Gamma \in \mathcal{P}(X \sim P_X, Y \sim P_G)} \mathbb{E}_{(X,Y) \sim \Gamma} [c(X, Y)] = \inf_{Q: Q_Z = P_Z} \mathbb{E}_{P_X} \mathbb{E}_{Q(Z|X)} [c(X, G(Z))],$$

where  $Q_Z$  is the marginal distribution of  $Z$  when  $X \sim P_X$  and  $Z \sim Q(Z|X)$ .



$$D_{\text{WAE}}(P_X, P_G) := \inf_{Q(Z|X) \in \mathcal{Q}} \mathbb{E}_{P_X} \mathbb{E}_{Q(Z|X)} [c(X, G(Z))] + \lambda \cdot \mathcal{D}_Z(Q_Z, P_Z), \quad (\text{WAE objective})$$

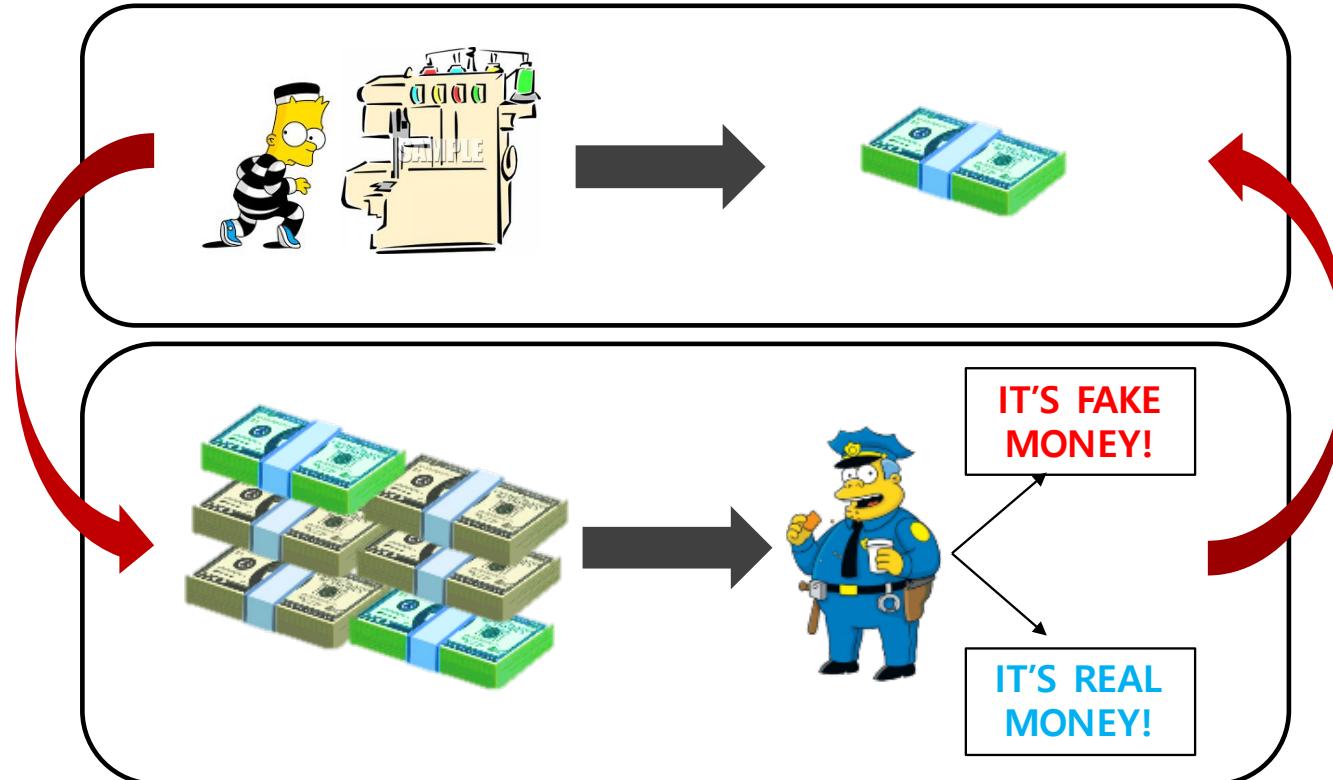
- What it means is that
  1. we can reformulate the **optimal transport problem** into the **WAE objective** using its dual formulations.
  2. The WAE objective contains a reconstruction loss and an arbitrary divergence between the aggregated posterior  $Q_Z = \int Q(Z|X) dP_X$  and the prior  $P_Z$ .
  3. If we use Jensen-Shannon divergence for  $D_Z$  (i.e.,  $D_Z(Q_Z, P_Z) = D_{JS}(Q_Z, P_Z)$ ), and use the adversarial training to estimate it, it becomes the adversarial auto-encoder.



# Generative Adversarial Network

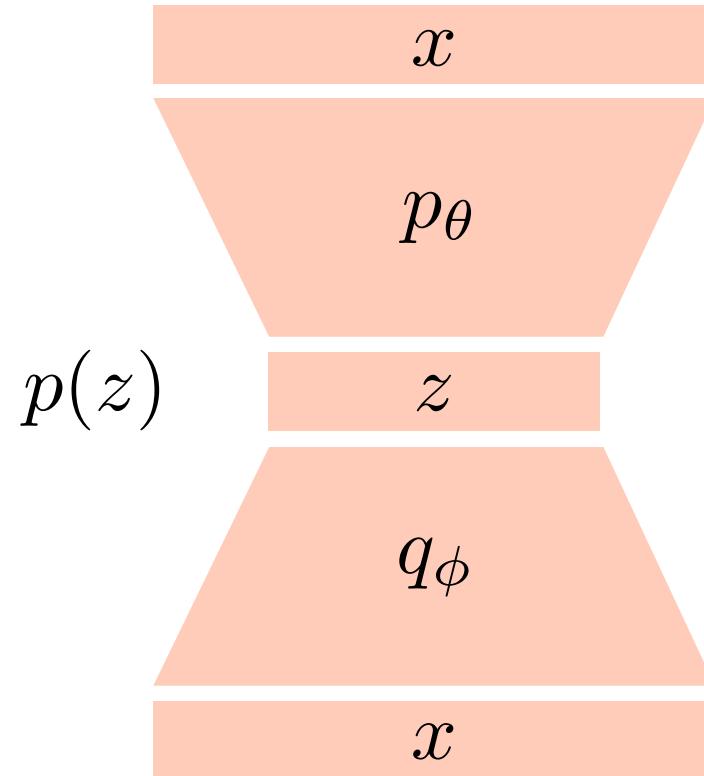
I. Goodfellow et al., "Generative Adversarial Networks", NIPS, 2014

# GAN

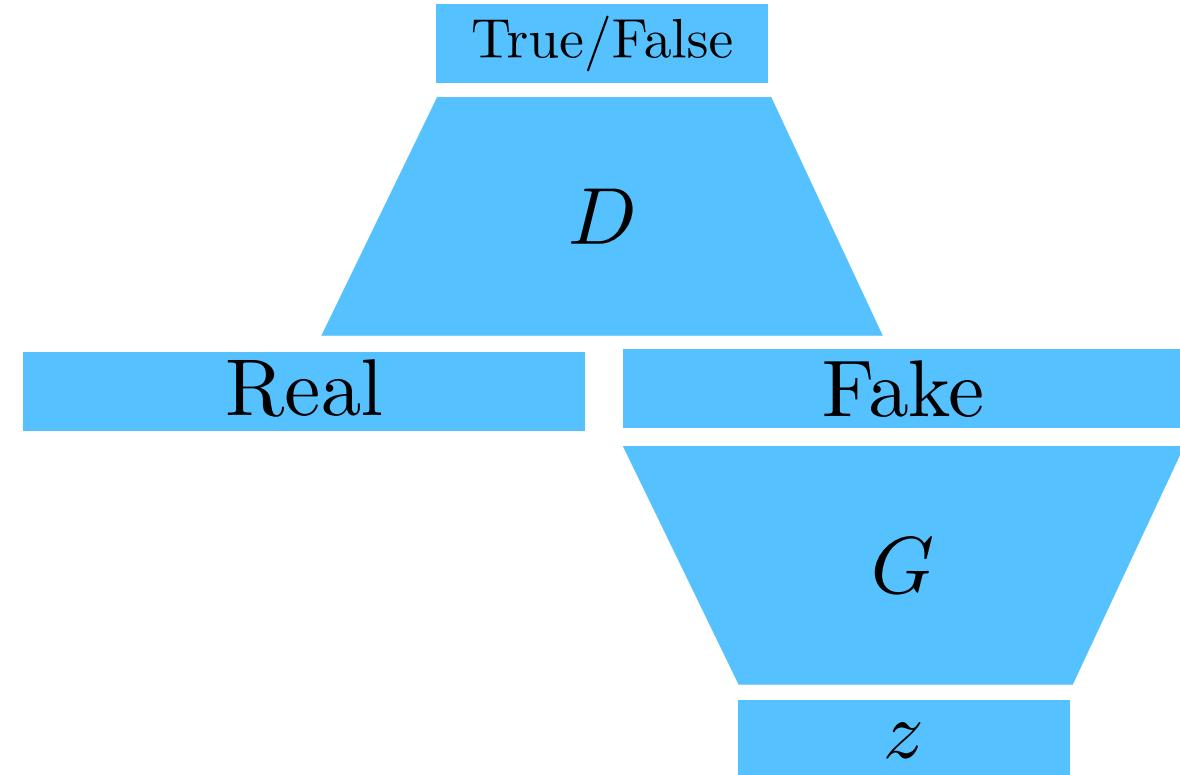


$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))].$$

# GAN vs. VAE



Variational Auto-encoder



Generative Adversarial Networks

# GAN Objective



- GAN is a two player minimax game between **generator** and **discriminator**.
  - **Discriminator** objective

$$\max_D V(G, D) = E_{\mathbf{x} \sim p_{\text{data}}} [\log D(\mathbf{x})] + E_{\mathbf{x} \sim p_G} [\log(1 - D(\mathbf{x}))]$$

- The optimal discriminator is

$$D_G^*(\mathbf{x}) = \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_G(\mathbf{x})}$$

# GAN Objective

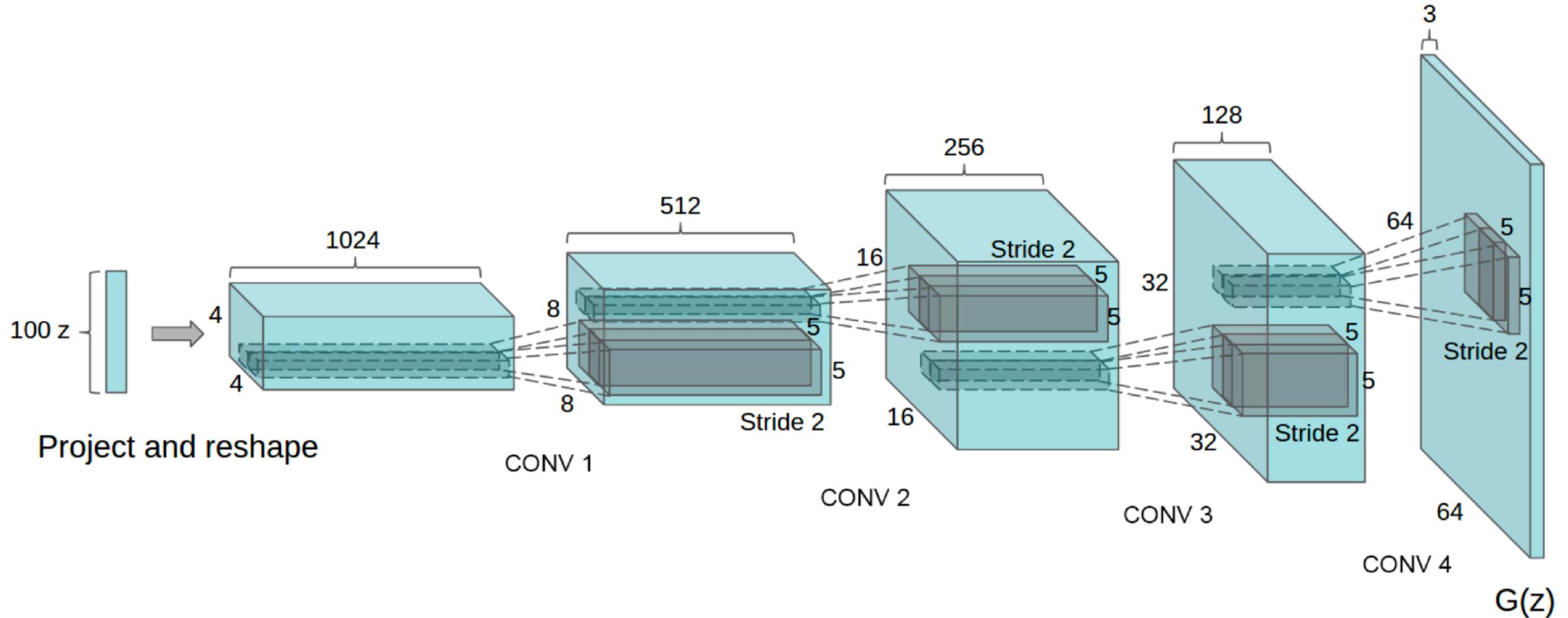
- GAN is a two player minimax game between **generator** and **discriminator**.
  - **Generator** objective

$$\min_G V(G, D) = E_{\mathbf{x} \sim p_{\text{data}}} [\log D(\mathbf{x})] + E_{\mathbf{x} \sim p_G} [\log(1 - D(\mathbf{x}))]$$

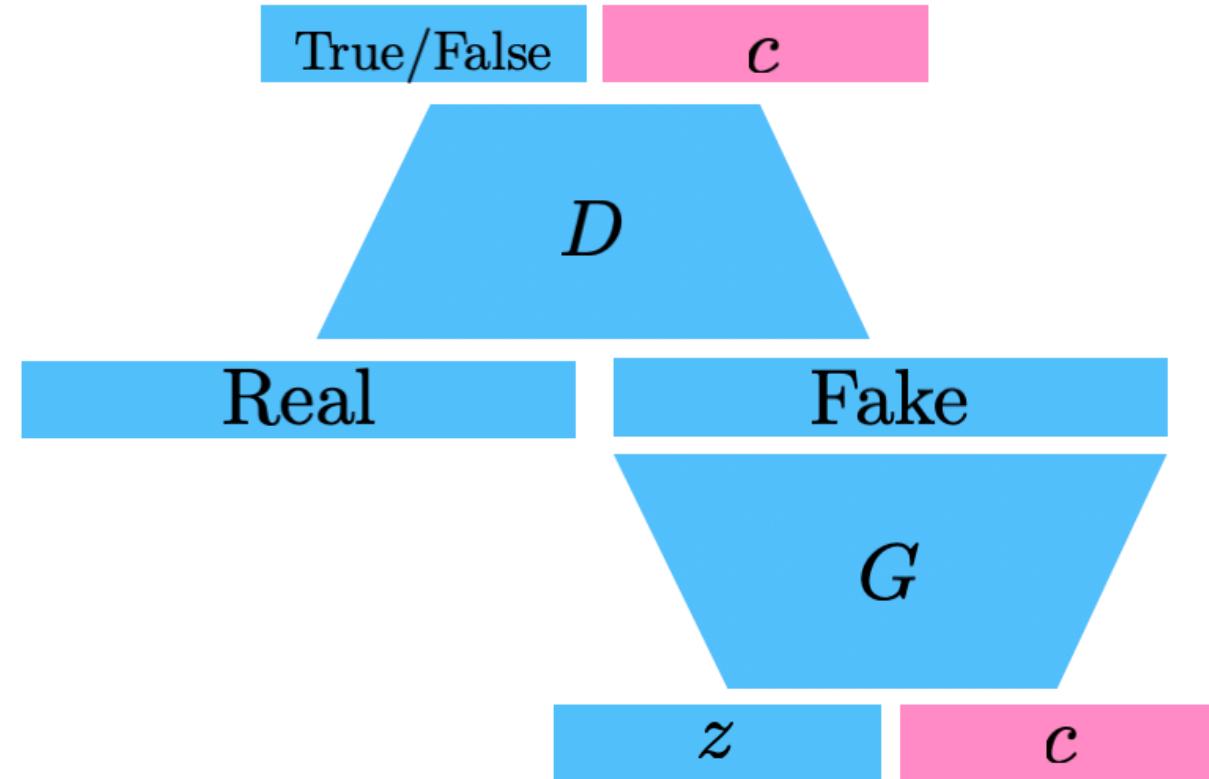
- Plugging in the **optimal discriminator**, we get

$$\begin{aligned}
 V(G, D_G^*(\mathbf{x})) &= E_{\mathbf{x} \sim p_{\text{data}}} \left[ \log \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_G(\mathbf{x})} \right] + E_{\mathbf{x} \sim p_G} \left[ \log \frac{p_G(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_G(\mathbf{x})} \right] \\
 &= E_{\mathbf{x} \sim p_{\text{data}}} \left[ \log \frac{p_{\text{data}}(\mathbf{x})}{\frac{p_{\text{data}}(\mathbf{x}) + p_G(\mathbf{x})}{2}} \right] + E_{\mathbf{x} \sim p_G} \left[ \log \frac{p_G(\mathbf{x})}{\frac{p_{\text{data}}(\mathbf{x}) + p_G(\mathbf{x})}{2}} \right] - \log 4 \\
 &= \underbrace{D_{KL} \left[ p_{\text{data}}, \frac{p_{\text{data}} + p_G}{2} \right] + D_{KL} \left[ p_G, \frac{p_{\text{data}} + p_G}{2} \right]}_{2 \times \text{Jenson-Shannon Divergence (JSD)}} - \log 4 \\
 &= 2D_{JSD}[p_{\text{data}}, p_G] - \log 4
 \end{aligned}$$

# DCGAN



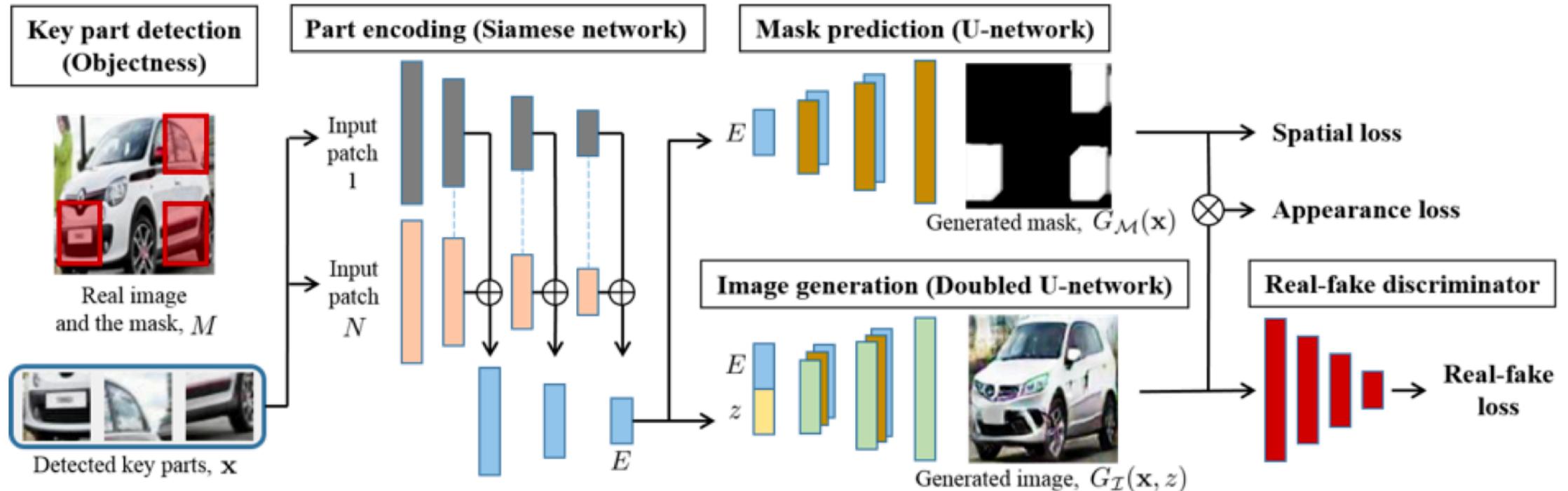
# Info-GAN



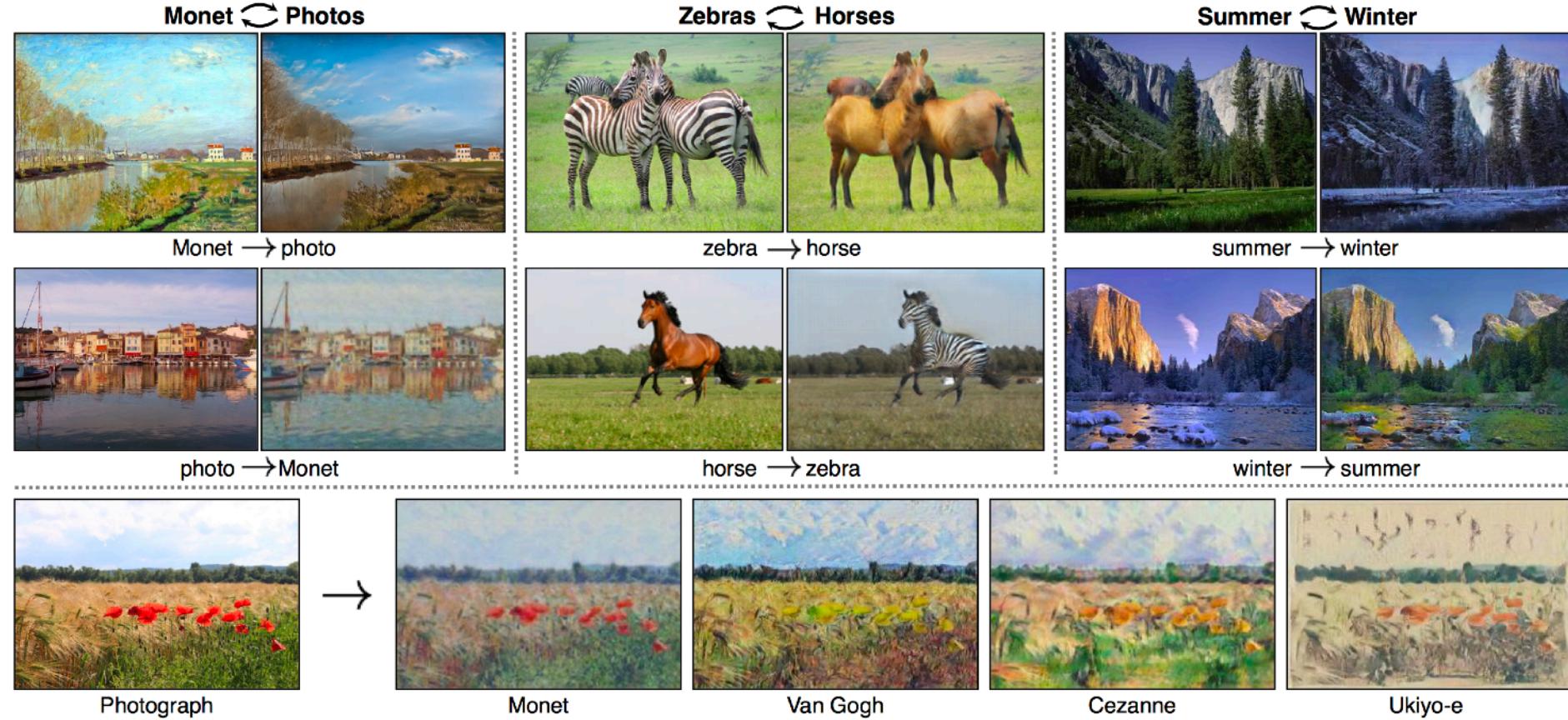
# Text2Image



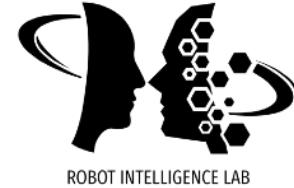
# Puzzle-GAN



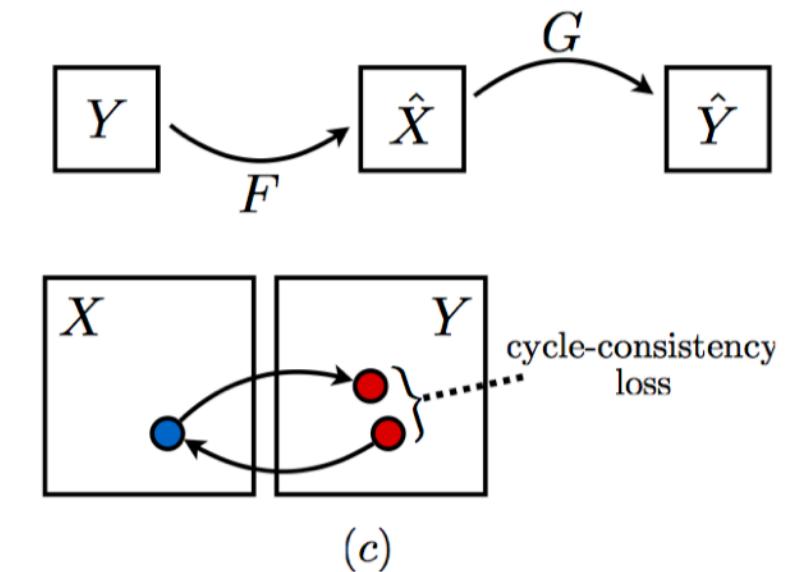
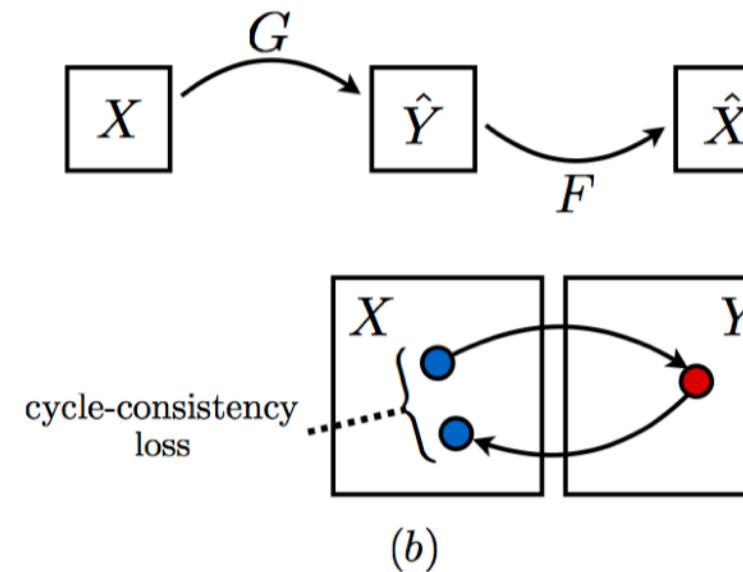
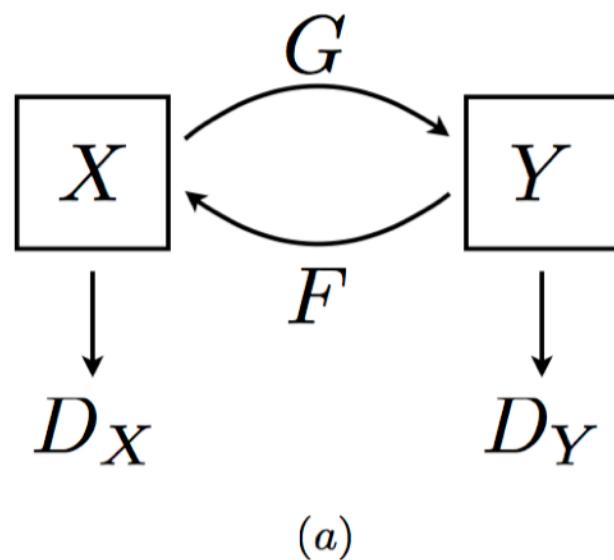
# CycleGAN



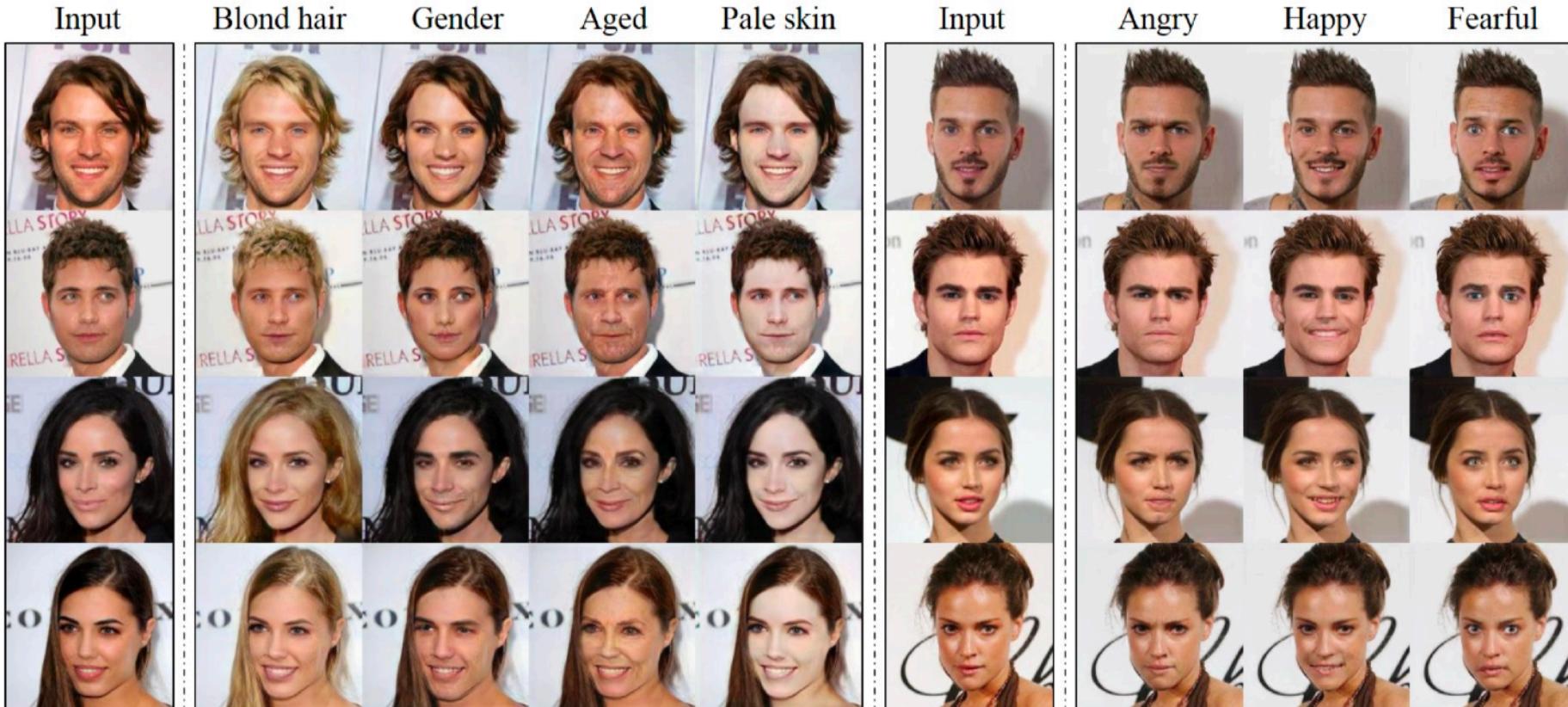
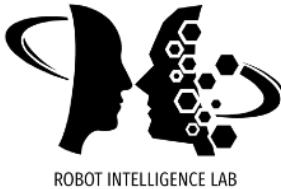
# CycleGAN



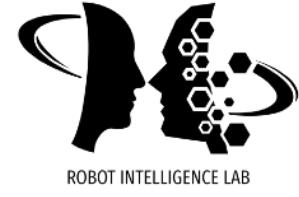
- Cycle-consistency Loss



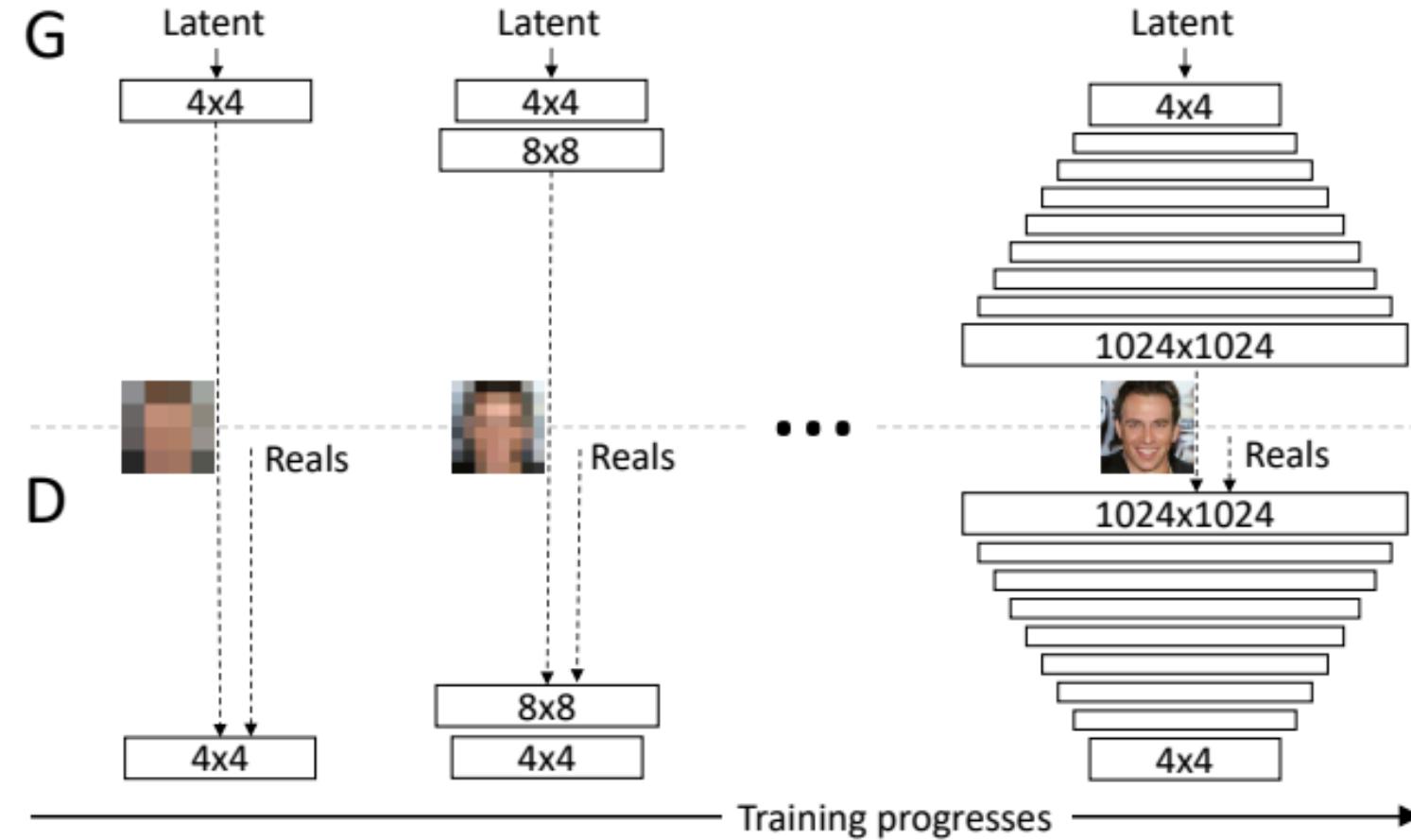
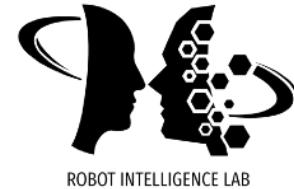
# Star-GAN



# Progressive-GAN

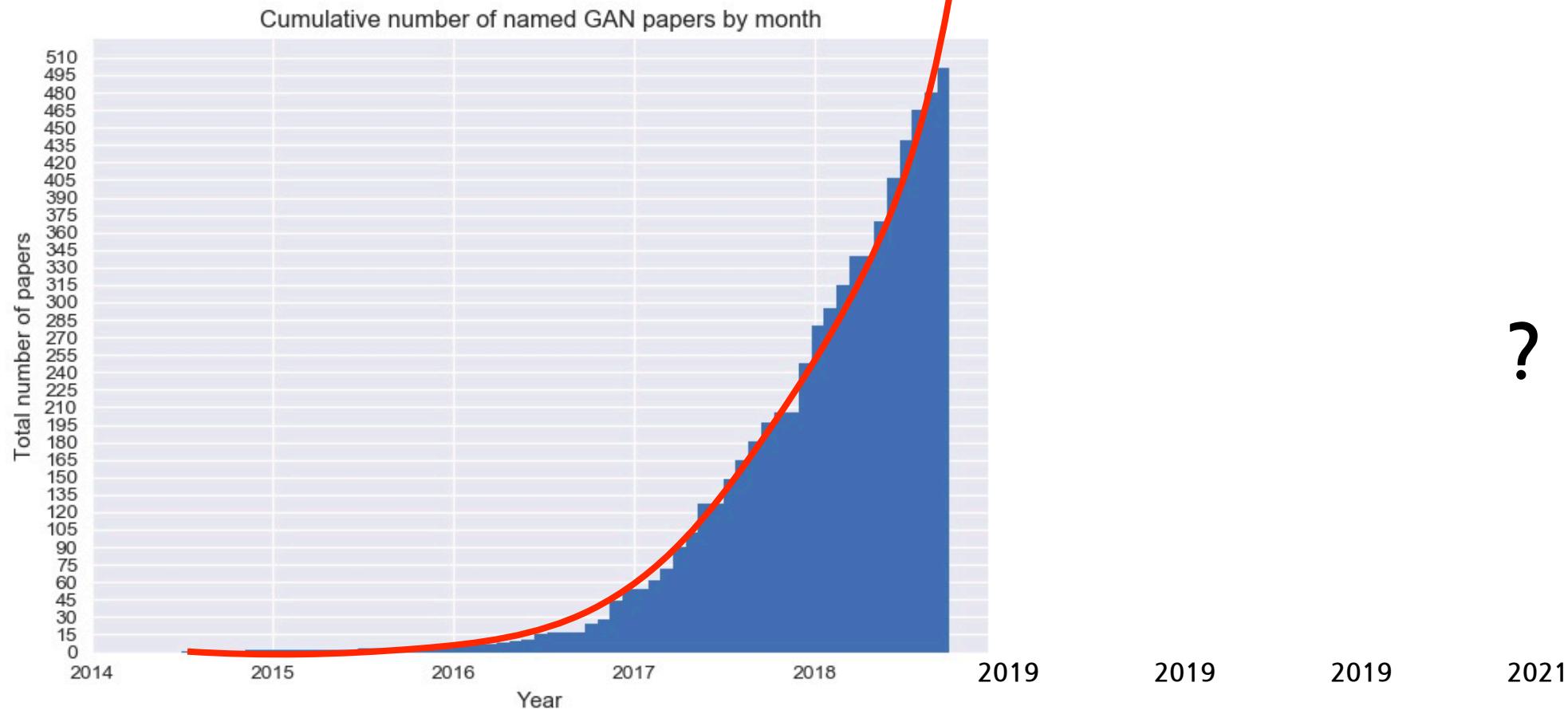


# Progressive-GAN



<https://arxiv.org/abs/1710.10196>

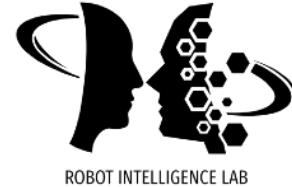
# and WAY MORE





# Normalizing Flow

# Normalizing Flow



- Limitations of existing work
  - Autoregressive models provide tractable likelihood but no direct mechanisms for learning features
  - Variational autoencoders can learn feature representations (via latent variables  $z$ ) but have intractable marginal likelihoods.
- Key question
  - Can we design a latent variable model with tractable likelihoods?
- Desirable properties of any model distribution:
  - Analytic density (explicit model)
  - Easy-to-sample
- Key idea
  - Map simple distributions (easy to sample and evaluate densities) to complex distributions (learned via data) using **change of variables**.

# Normalizing Flow

- Change of Variables Formula
  - Let  $Z$  be a uniform random variable  $\mathcal{U}[0,2]$  with density  $p_Z$ . What is  $p_Z(1)$ ?

# Normalizing Flow

- Change of Variables Formula
  - Let  $Z$  be a uniform random variable  $\mathcal{U}[0,2]$  with density  $p_Z$ . What is  $p_Z(1)$ ?
    - The answer is  $\frac{1}{2}$ .
  - Let  $X = 4Z$  and let  $p_X$  be its density. What is  $p_X(4)$ ?

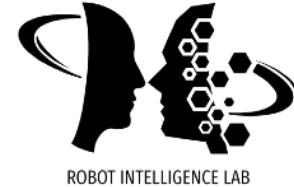
# Normalizing Flow

- Change of Variables Formula
  - Let  $Z$  be a uniform random variable  $\mathcal{U}[0,2]$  with density  $p_Z$ . What is  $p_Z(1)$ ?
    - The answer is  $\frac{1}{2}$ .
  - Let  $X = 4Z$  and let  $p_X$  be its density. What is  $p_X(4)$ ?
    - $p_X(4) = P(X = 4) = P(4Z = 4) = P(Z = 1) = p_Z(1) = 1/2$  ?

# Normalizing Flow

- Change of Variables Formula
  - Let  $Z$  be a uniform random variable  $\mathcal{U}[0,2]$  with density  $p_Z$ . What is  $p_Z(1)$ ?
    - The answer is  $\frac{1}{2}$ .
  - Let  $X = 4Z$  and let  $p_X$  be its density. What is  $p_X(4)$ ?
    - $p_X(4) = P(X = 4) = P(4Z = 4) = P(Z = 1) = p_Z(1) = 1/2$  ?
    - Unfortunately **Not**.
    - Clearly,  $X$  is uniform in  $[0,8]$ , so  $p_X(4) = 1/8$ .

# Normalizing Flow



- **Change of variables (1D case)**

- If  $X = f(Z)$  and  $f(\cdot)$  is monotone with inverse  $Z = f^{-1}(X) = h(X)$ , then

$$p_X(x) = p_Z(h(x)) |h'(x)|$$

- Previous example

- If  $X = 4Z$  and  $Z \sim \mathcal{U}[0,2]$ , what is  $p_X(4)$ ?
- Note that  $h(X) = X/4$  and  $h'(X) = 1/4$ .
- Then,  $p_X(4) = p_Z(1)h'(4) = 1/2 \times 1/4 = 1/8$ .

# Normalizing Flow

- **Generalized change of variables**

- The mapping between  $Z$  and  $X$ , given by  $f: \mathbb{R}^n \rightarrow \mathbb{R}^n$ , is invertible such that  $X = f(Z)$  and  $Z = f^{-1}(X)$ .

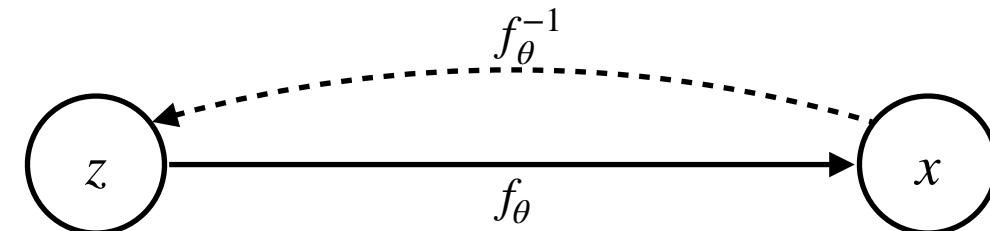
$$p_X(x) = p_Z(f^{-1}(x)) \left| \det \left( \frac{\partial f^{-1}(x)}{\partial x} \right) \right|$$

- Note 1:  $x$  and  $z$  need to be continuous and have the same dimension. For example, if  $x \in \mathbb{R}^n$  then  $z \in \mathbb{R}^n$ .
- Note 2: For any invertible matrix  $A$ ,  $\det(A^{-1}) = \det(A)^{-1}$ , then

$$p_X(x) = p_Z(z) \left| \det \left( \frac{\partial f(z)}{\partial z} \right) \right|^{-1}$$

# Normalizing Flow

- Normalizing flow models
  - Consider a directed, latent-variable model over observed variables  $X$  and latent variables  $Z$ .
  - In a **normalizing flow model**, the mapping between  $Z$  and  $X$ , given by  $f_\theta : \mathbb{R}^n \rightarrow \mathbb{R}^n$  is deterministic and invertible such that  $X = f_\theta(Z)$  and  $Z = f_\theta^{-1}(X)$



- Using change of variables, the marginal likelihood  $p(x)$  is given by

$$p_X(x; \theta) = p_Z(f_\theta^{-1}(x)) \left| \det \left( \frac{\partial f_\theta^{-1}(x)}{\partial x} \right) \right|$$

# Normalizing Flow

- Normalizing flow models
  - **Normalizing**: Change of variables gives a normalized density after applying an invertible transformation
  - **Flow**: Invertible transformations can be composed with each other

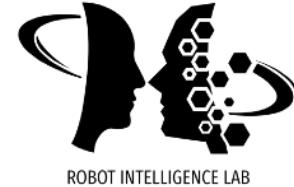
$$z_m \equiv f_\theta^m \circ \dots \circ f_\theta^1(z_0) = f_\theta^m(f_\theta^{m-1}(\dots(f_\theta^1(z_0)))) \equiv f_\theta(z_0)$$

- Start with a simple distribution for  $z_0$  (e.g., Gaussian)
- Apply a sequence of  $M$  invertible transformations  $x \equiv z_M$
- By change of variables

$$p_X(x; \theta) = p_Z(f_\theta^{-1}(x)) \prod_{m=1}^M \left| \det \left( \frac{\partial(f_\theta^m)^{-1}(z_m)}{\partial z_m} \right) \right|$$

(Note: determinant of product equals product of determinants)

# Planar flows



- Planar flow
- Invertible transformation

$$\mathbf{x} = f_{\theta}(\mathbf{z}) = \mathbf{z} + \mathbf{u}h(\mathbf{w}^T \mathbf{z} + b)$$

parametrized by  $\theta = (\mathbf{w}, \mathbf{u}, b)$  where  $h(\cdot)$  is a non-linearity.

- Absolute value of the determinant of the Jacobian is given by

$$\left| \det \frac{\partial f_{\theta}(\mathbf{z})}{\partial \mathbf{z}} \right| = \left| \det (I + h'(\mathbf{w}^T \mathbf{z} + b) \mathbf{u} \mathbf{w}^T) \right| = \left| 1 + h'(\mathbf{w}^T \mathbf{z} + b) \mathbf{u}^T \mathbf{w} \right|$$

(matrix determinant lemma)

# Planar flows



- Learning via **maximum likelihood** over the dataset  $\mathcal{D}$ .

$$\max_{\theta} \log p_X(\mathcal{D}; \theta) = \sum_{x \in \mathcal{D}} \log p_Z(f_{\theta}^{-1}(x)) + \log \left| \deg \left( \frac{\partial f_{\theta}^{-1}(x)}{\partial x} \right) \right|$$

- Exact likelihood evaluation via inverse transform  $\mathbf{x} \mapsto \mathbf{z}$  and change of variables formula
- Sampling via forward transformation  $\mathbf{z} \mapsto \mathbf{x}$

$$\mathbf{z} \sim p_Z(\mathbf{z}) \quad \mathbf{x} = f_{\theta}(\mathbf{z})$$

- Latent representations inferred via inverse transformation (in inference network required!)

$$\mathbf{z} = f_{\theta}^{-1}(\mathbf{x})$$

# Thank You



ROBOT INTELLIGENCE LAB