

# Generative Adversarial Networks (GAN)

# Generative Model

- Generative Adversarial Networks (GANs)
- Variational Autoencoders (VAEs)
  - Previous lectures
- Autoregressive models : PixelRNN and PixelCNN
  - <https://arxiv.org/abs/1601.06759>
  - <https://arxiv.org/abs/1606.05328>
  - <https://openreview.net/pdf?id=BJrFC6ceg>
- Good reference
  - [http://cs231n.stanford.edu/slides/2019/cs231n\\_2019\\_lecture13.pdf](http://cs231n.stanford.edu/slides/2019/cs231n_2019_lecture13.pdf)

# GAN

I. Goodfellow et al., “Generative Adversarial Nets”  
NIPS2014

---

# Generative Adversarial Nets

---

Ian J. Goodfellow, Jean Pouget-Abadie\*, Mehdi Mirza, Bing Xu, David Warde-Farley,  
Sherjil Ozair<sup>†</sup>, Aaron Courville, Yoshua Bengio<sup>‡</sup>

Département d'informatique et de recherche opérationnelle  
Université de Montréal  
Montréal, QC H3C 3J7

## Abstract

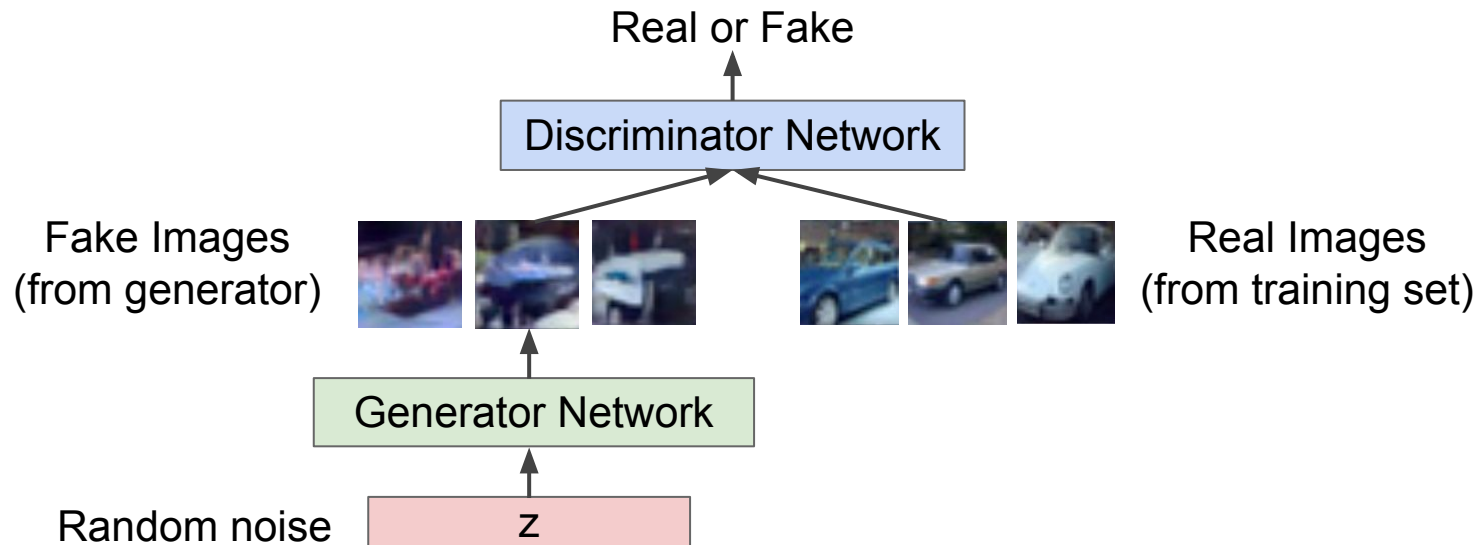
We propose a new framework for estimating generative models via an adversarial process, in which we simultaneously train two models: a generative model  $G$  that captures the data distribution, and a discriminative model  $D$  that estimates the probability that a sample came from the training data rather than  $G$ . The training procedure for  $G$  is to maximize the probability of  $D$  making a mistake. This framework corresponds to a minimax two-player game. In the space of arbitrary functions  $G$  and  $D$ , a unique solution exists, with  $G$  recovering the training data distribution and  $D$  equal to  $\frac{1}{2}$  everywhere. In the case where  $G$  and  $D$  are defined by multilayer perceptrons, the entire system can be trained with backpropagation. There is no need for any Markov chains or unrolled approximate inference networks during either training or generation of samples. Experiments demonstrate the potential of the framework through qualitative and quantitative evaluation of the generated samples.

# Training GANs: Two-player game

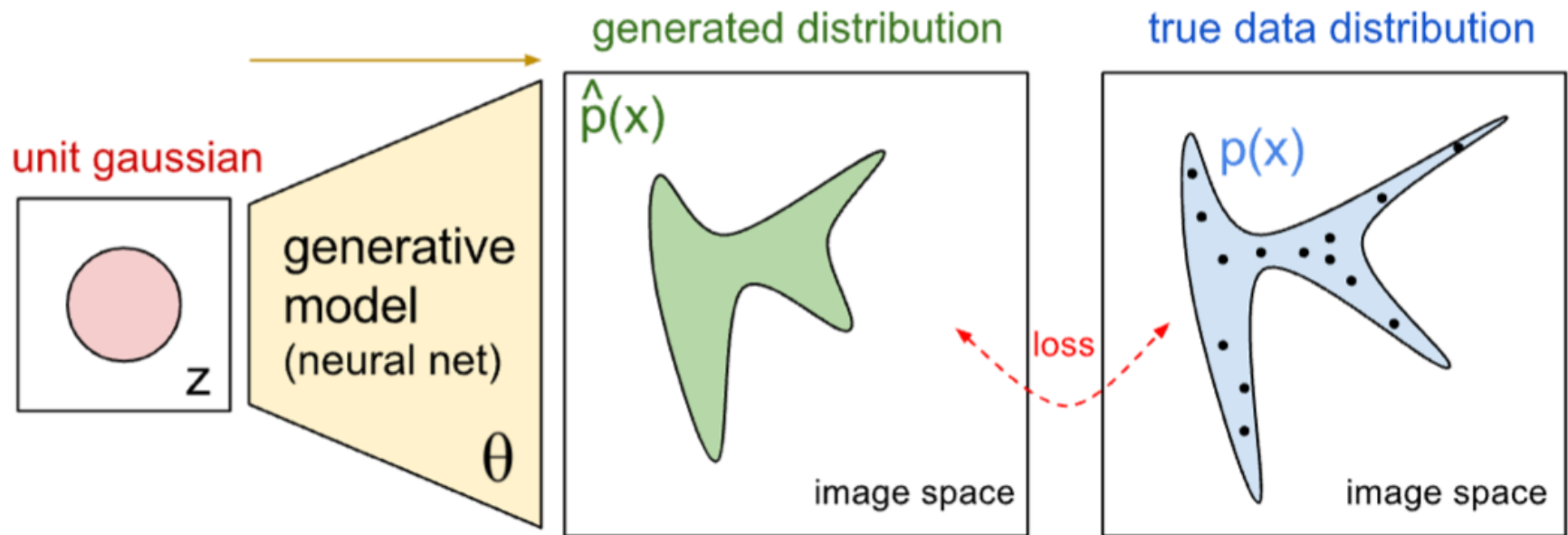
Ian Goodfellow et al., "Generative Adversarial Nets", NIPS 2014

**Generator network:** try to fool the discriminator by generating real-looking images

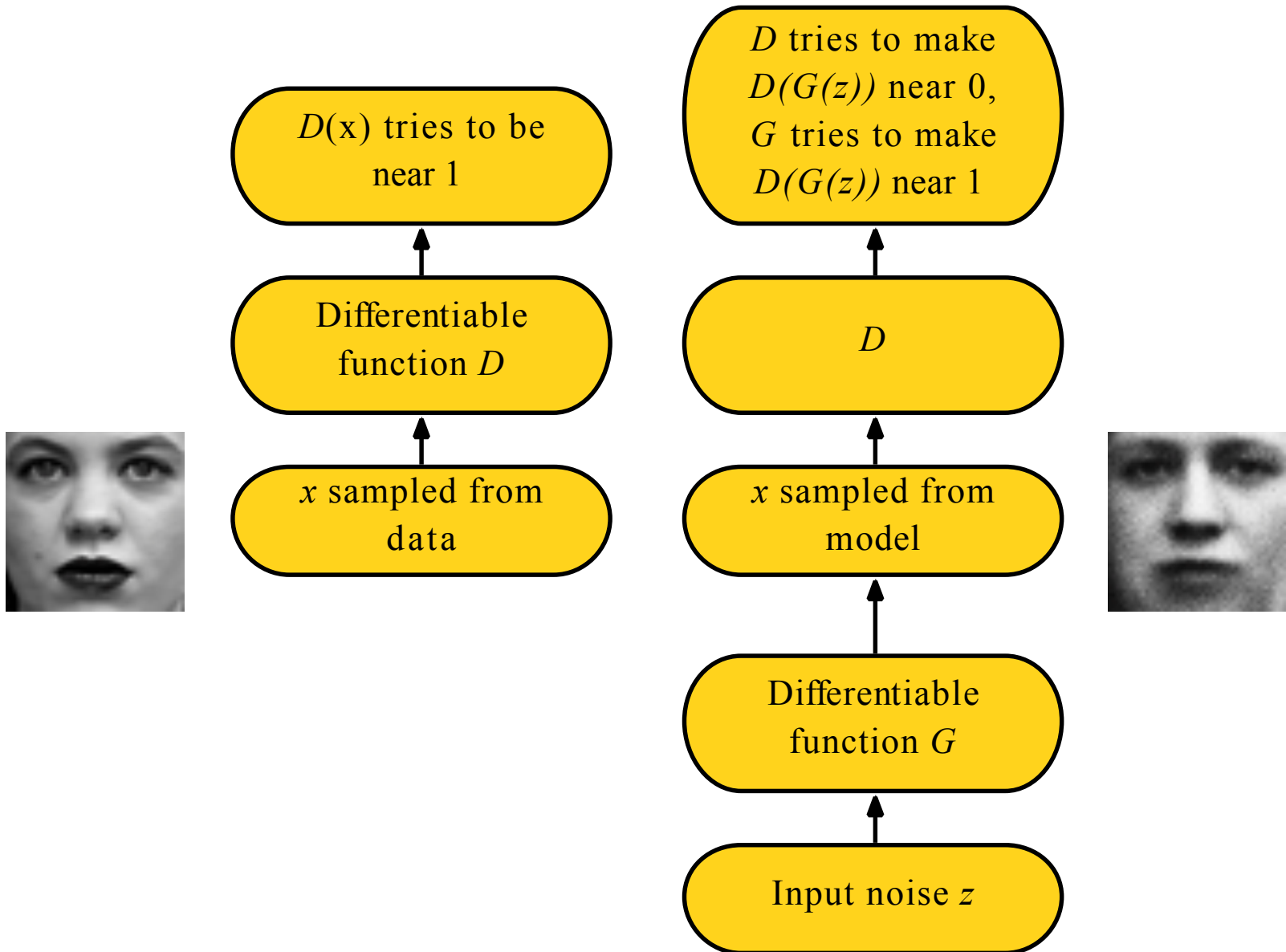
**Discriminator network:** try to distinguish between real and fake images



# Minimax Game



# Adversarial Nets Framework



---

**Algorithm 1** Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator,  $k$ , is a hyperparameter. We used  $k = 1$ , the least expensive option, in our experiments.

---

**for** number of training iterations **do**

**for**  $k$  steps **do**

- Sample minibatch of  $m$  noise samples  $\{\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(m)}\}$  from noise prior  $p_g(\mathbf{z})$ .
- Sample minibatch of  $m$  examples  $\{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}\}$  from data generating distribution  $p_{\text{data}}(\mathbf{x})$ .
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^m \left[ \log D(\mathbf{x}^{(i)}) + \log \left( 1 - D(G(\mathbf{z}^{(i)})) \right) \right].$$

**end for**

- Sample minibatch of  $m$  noise samples  $\{\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(m)}\}$  from noise prior  $p_g(\mathbf{z})$ .
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^m \log \left( 1 - D(G(\mathbf{z}^{(i)})) \right).$$

**end for**

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

---



# Jensen Shannon (JS)-Divergence

# Wasserstein GAN

Martin Arjovsky<sup>1</sup>, Soumith Chintala<sup>2</sup>, and Léon Bottou<sup>1,2</sup>

<sup>1</sup>Courant Institute of Mathematical Sciences

<sup>2</sup>Facebook AI Research

The *Earth-Mover* (EM) distance or Wasserstein-1

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y) \sim \gamma} [\|x - y\|] , \quad (1)$$

where  $\Pi(\mathbb{P}_r, \mathbb{P}_g)$  denotes the set of all joint distributions  $\gamma(x, y)$  whose marginals are respectively  $\mathbb{P}_r$  and  $\mathbb{P}_g$ . Intuitively,  $\gamma(x, y)$  indicates how much “mass” must be transported from  $x$  to  $y$  in order to transform the distributions  $\mathbb{P}_r$  into the distribution  $\mathbb{P}_g$ . The EM distance then is the “cost” of the optimal transport plan.

# *A Primer on* Optimal Transport

Marco Cuturi

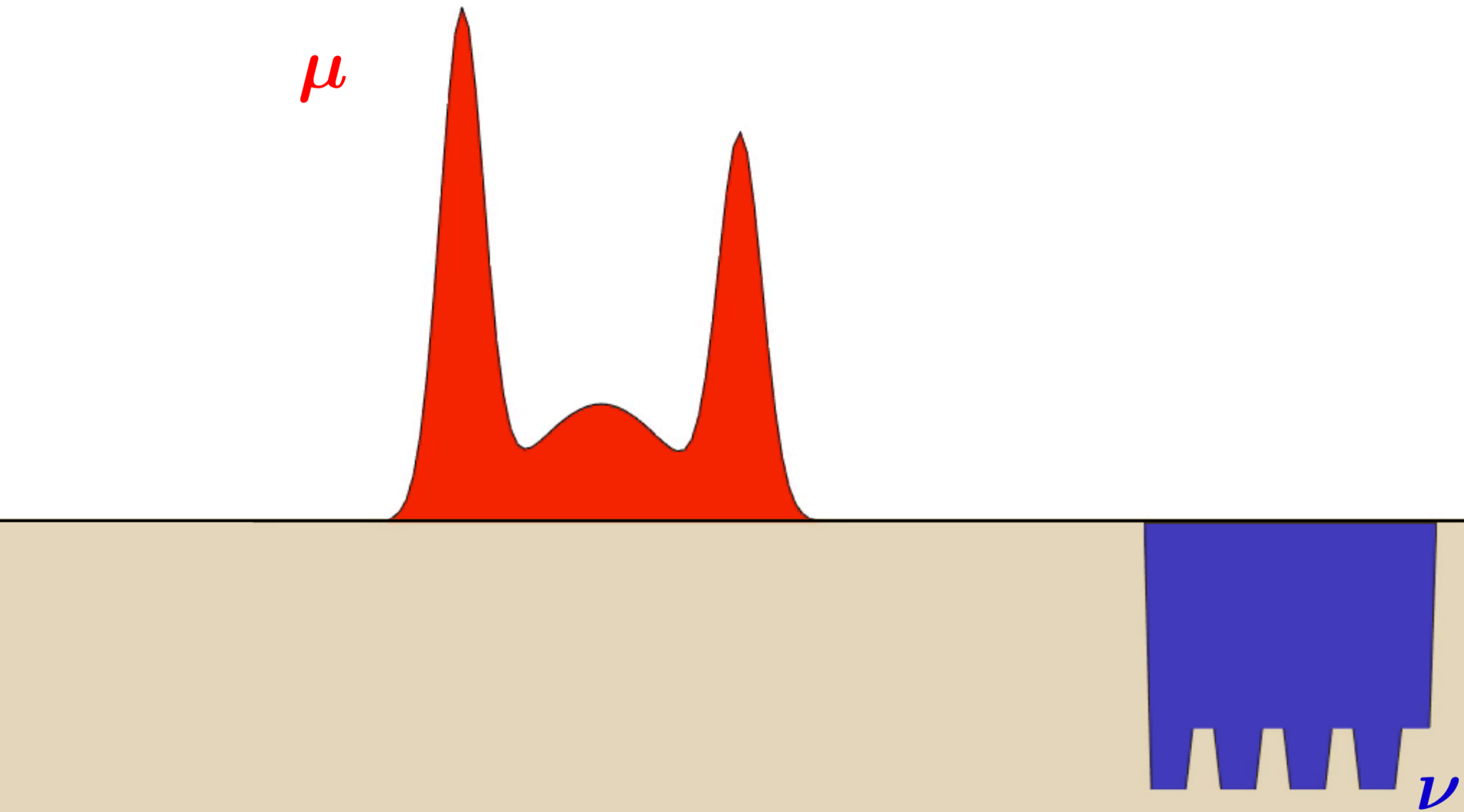


Justin Solomon



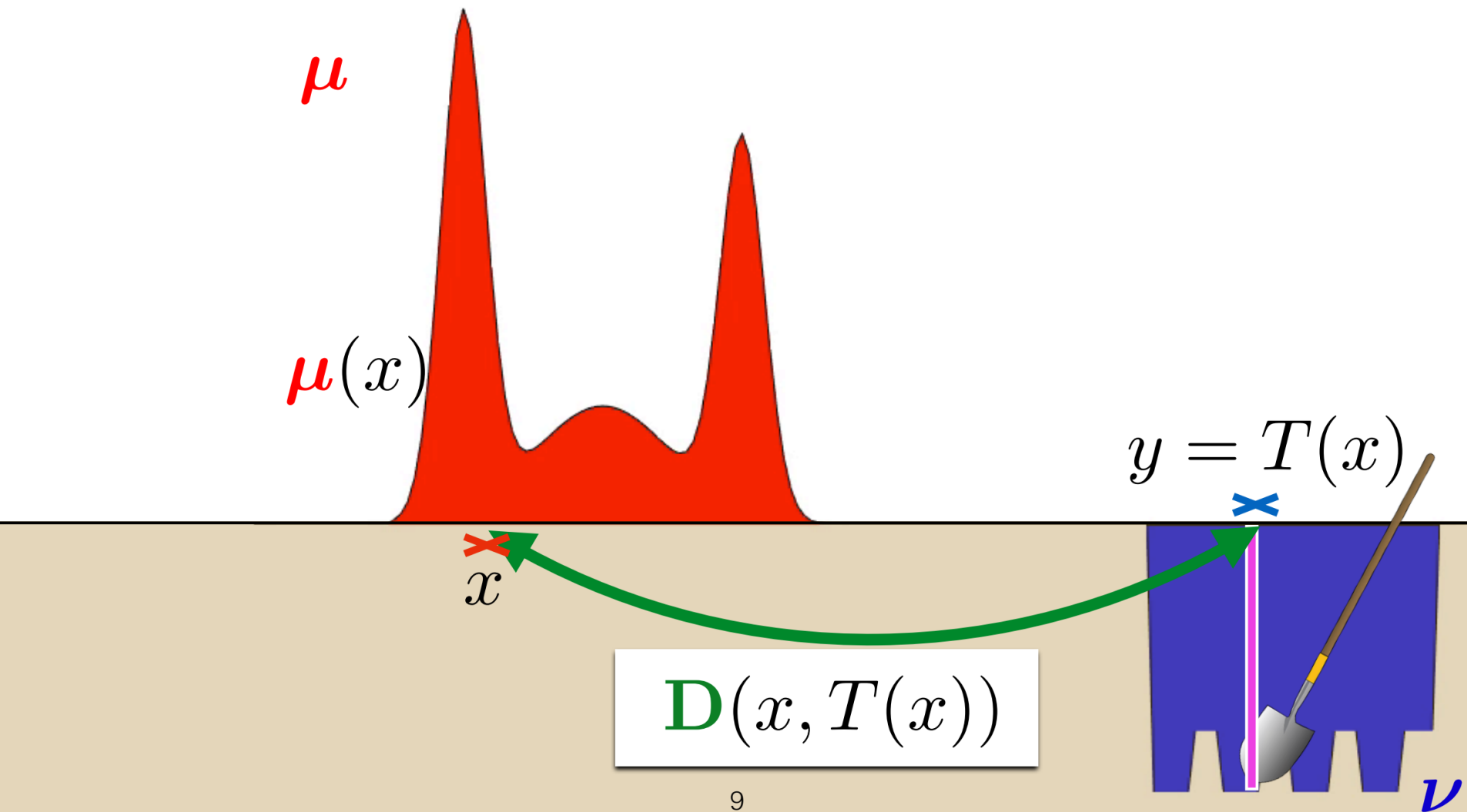
# Origins: Monge Problem

---



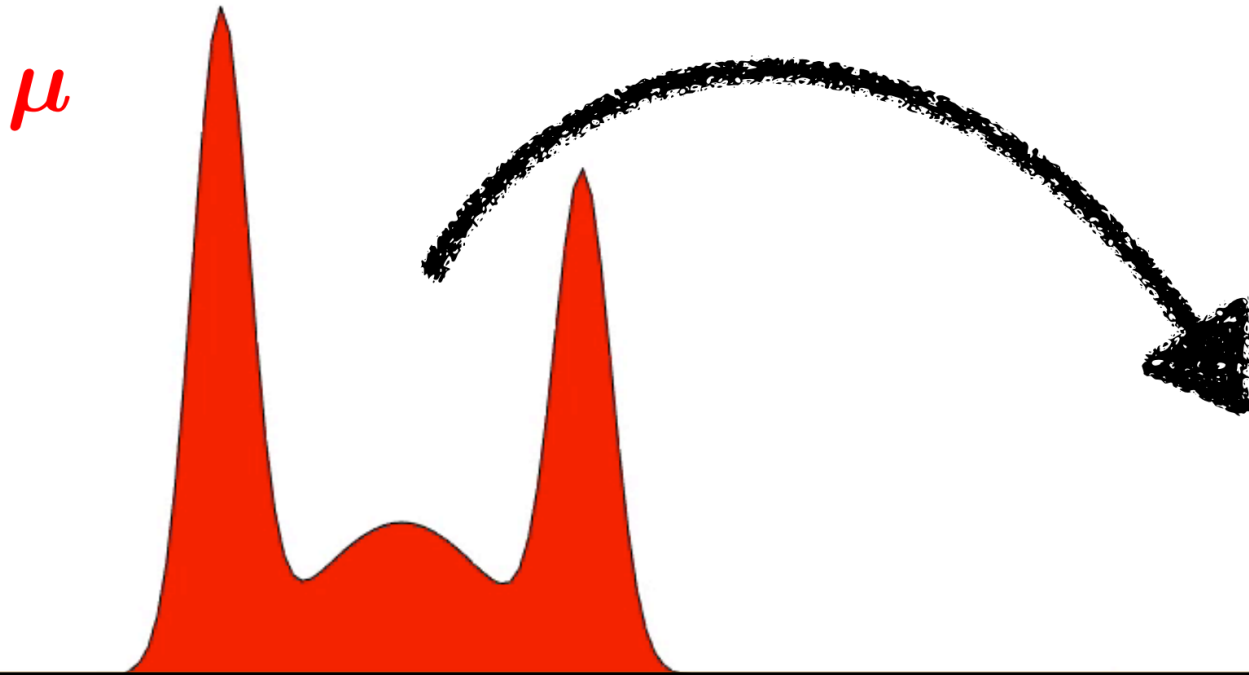
# Origins: Monge's Problem

*In 1781 however...*

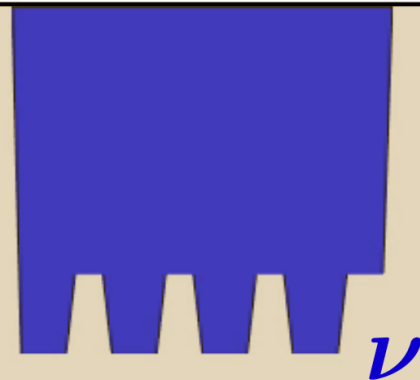


# Origins: Monge's Problem

$T$  must **push-forward** the red measure towards the blue



What  $T$  s.t.  $T_{\#}\mu = \nu$   
minimizes  $\int D(x, T(x))\mu(dx)$ ?



# JS-Divergence vs. W-distance

**Example 1** (Learning parallel lines). Let  $Z \sim U[0, 1]$  the uniform distribution on the unit interval. Let  $\mathbb{P}_0$  be the distribution of  $(0, Z) \in \mathbb{R}^2$  (a 0 on the x-axis and the random variable  $Z$  on the y-axis), uniform on a straight vertical line passing through the origin. Now let  $g_\theta(z) = (\theta, z)$  with  $\theta$  a single real parameter. It is easy to see that in this case,

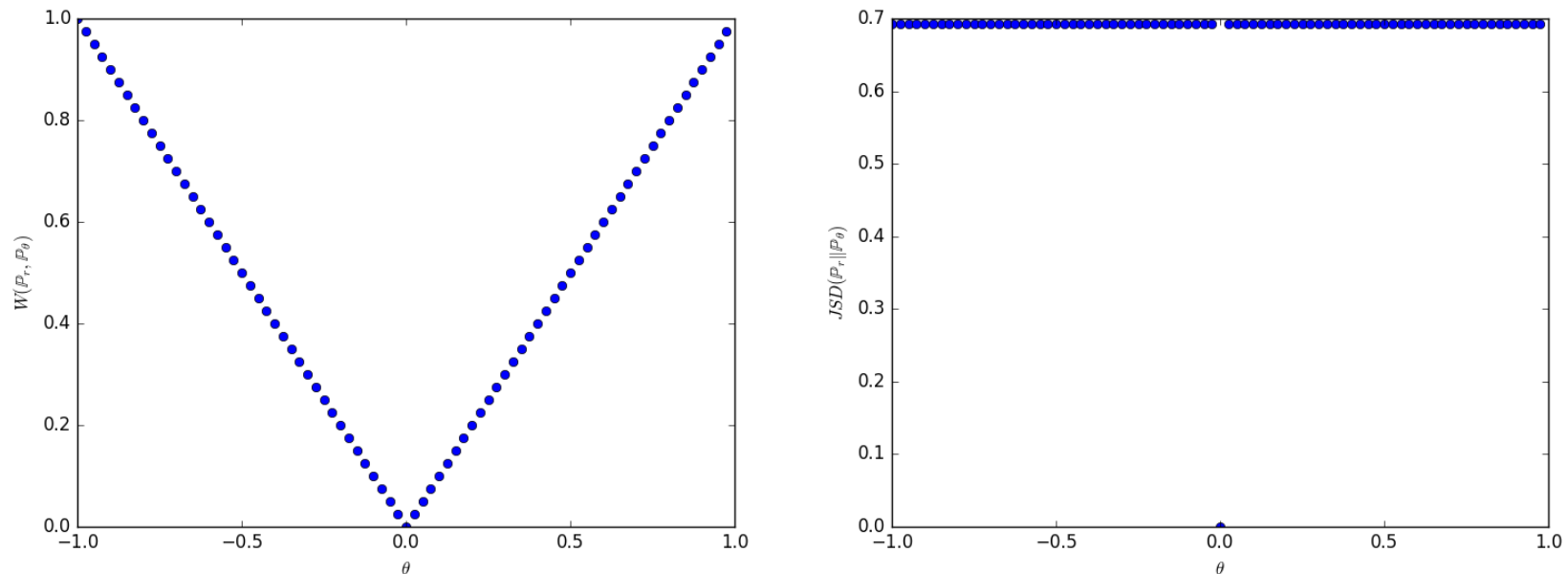


Figure 1: These plots show  $\rho(\mathbb{P}_\theta, \mathbb{P}_0)$  as a function of  $\theta$  when  $\rho$  is the EM distance (left plot) or the JS divergence (right plot). The EM plot is continuous and provides a usable gradient everywhere. The JS plot is not continuous and does not provide a usable gradient.

# Kantorovich-Rubinstein Duality

$$W(\mathbb{P}_r, \mathbb{P}_\theta) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x \sim \mathbb{P}_r} [f(x)] - \mathbb{E}_{x \sim \mathbb{P}_\theta} [f(x)]$$

---

**Algorithm 1** WGAN, our proposed algorithm. All experiments in the paper used the default values  $\alpha = 0.00005$ ,  $c = 0.01$ ,  $m = 64$ ,  $n_{\text{critic}} = 5$ .

---

**Require:** :  $\alpha$ , the learning rate.  $c$ , the clipping parameter.  $m$ , the batch size.  
 $n_{\text{critic}}$ , the number of iterations of the critic per generator iteration.

**Require:** :  $w_0$ , initial critic parameters.  $\theta_0$ , initial generator's parameters.

```
1: while  $\theta$  has not converged do
2:   for  $t = 0, \dots, n_{\text{critic}}$  do
3:     Sample  $\{x^{(i)}\}_{i=1}^m \sim \mathbb{P}_r$  a batch from the real data.
4:     Sample  $\{z^{(i)}\}_{i=1}^m \sim p(z)$  a batch of prior samples.
5:      $g_w \leftarrow \nabla_w \left[ \frac{1}{m} \sum_{i=1}^m f_w(x^{(i)}) - \frac{1}{m} \sum_{i=1}^m f_w(g_\theta(z^{(i)})) \right]$ 
6:      $w \leftarrow w + \alpha \cdot \text{RMSPProp}(w, g_w)$ 
7:      $w \leftarrow \text{clip}(w, -c, c)$ 
8:   end for
9:   Sample  $\{z^{(i)}\}_{i=1}^m \sim p(z)$  a batch of prior samples.
10:   $g_\theta \leftarrow -\nabla_\theta \frac{1}{m} \sum_{i=1}^m f_w(g_\theta(z^{(i)}))$ 
11:   $\theta \leftarrow \theta - \alpha \cdot \text{RMSPProp}(\theta, g_\theta)$ 
12: end while
```

---