Generative models for images

Bruno Galerne
bruno.galerne@univ-orleans.fr

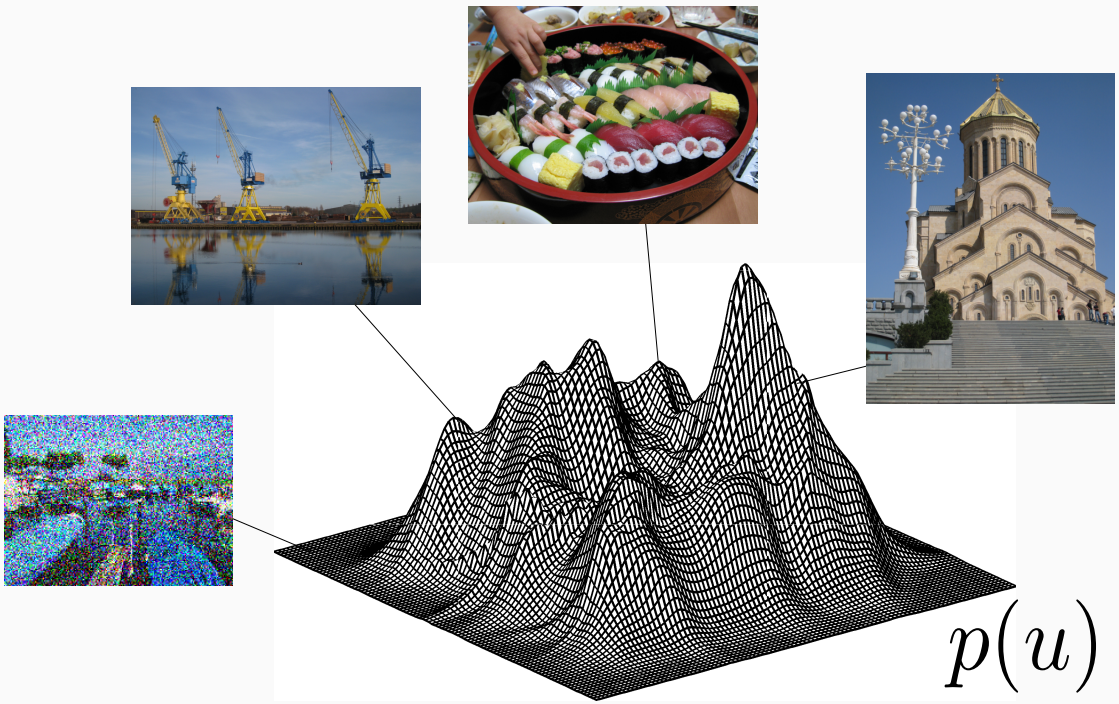
Summer school **Deep learning and applications**
IRMA, Université de Strasbourg
Wednesday August 31, 2022

Institut Denis Poisson
Université d'Orléans, Université de Tours, CNRS

Introduction on generative models

Generative models

1. Model and/or learn a distribution $p(u)$ on the space of images.



(source: Charles Deledalle)

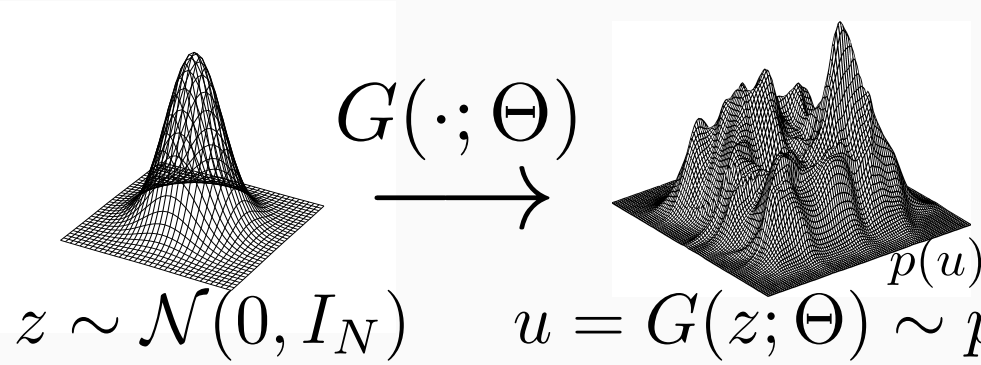
The images may represent:

- different instances of the same texture image.
- all images naturally described by a dataset of images.

2. Generate samples from this distribution.

Generative models

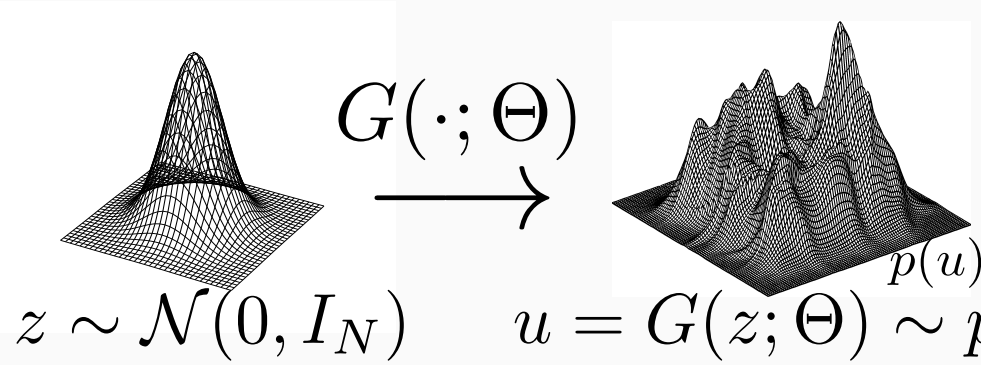
1. Model and/or learn a distribution $p(u)$ on the space of images.
2. Generate samples from this distribution.



- z is a generic source of randomness, often called the latent variable.
- If $G(\cdot;\Theta)$ is known, then $p = G(\cdot;\Theta)_{\#}\mathcal{N}(0, I_n)$ is the push-forward of the latent distribution.

Generative models

1. Model and/or learn a distribution $p(u)$ on the space of images.
2. Generate samples from this distribution.



- z is a generic source of randomness, often called the latent variable.
- If $G(\cdot;\Theta)$ is known, then $p = G(\cdot;\Theta)_{\#}\mathcal{N}(0, I_n)$ is the push-forward of the latent distribution.

The generator $G(\cdot;\Theta)$ can be:

- A deterministic function (e.g. convolution operator),
- A neural network with learned parameter,
- An iterative optimization algorithm (gradient descent,...),
- A stochastic sampling algorithm (e.g. MCMC, Langevin diffusion,. . .).

Generative models for images: Plan of the course

Part I: Introduction

Part II: Variational AutoEncoders (VAE)

1. Autoencoders
2. Deep latent variable models
3. Evidence lowerbound (ELBO) for training VAEs

Part III: Generative Adversarial Networks (GAN)

1. Introduction to GAN
2. Adversarial training and its issues
3. Wasserstein GAN (with recap on optimal transport)
4. Conditional GAN

Generative Adversarial Networks (GAN)

Generative Adversarial Networks (GAN)

Main references:

1. Original paper: (Goodfellow et al., 2014)
2. NIPS 2016 tutorial: (**?**)

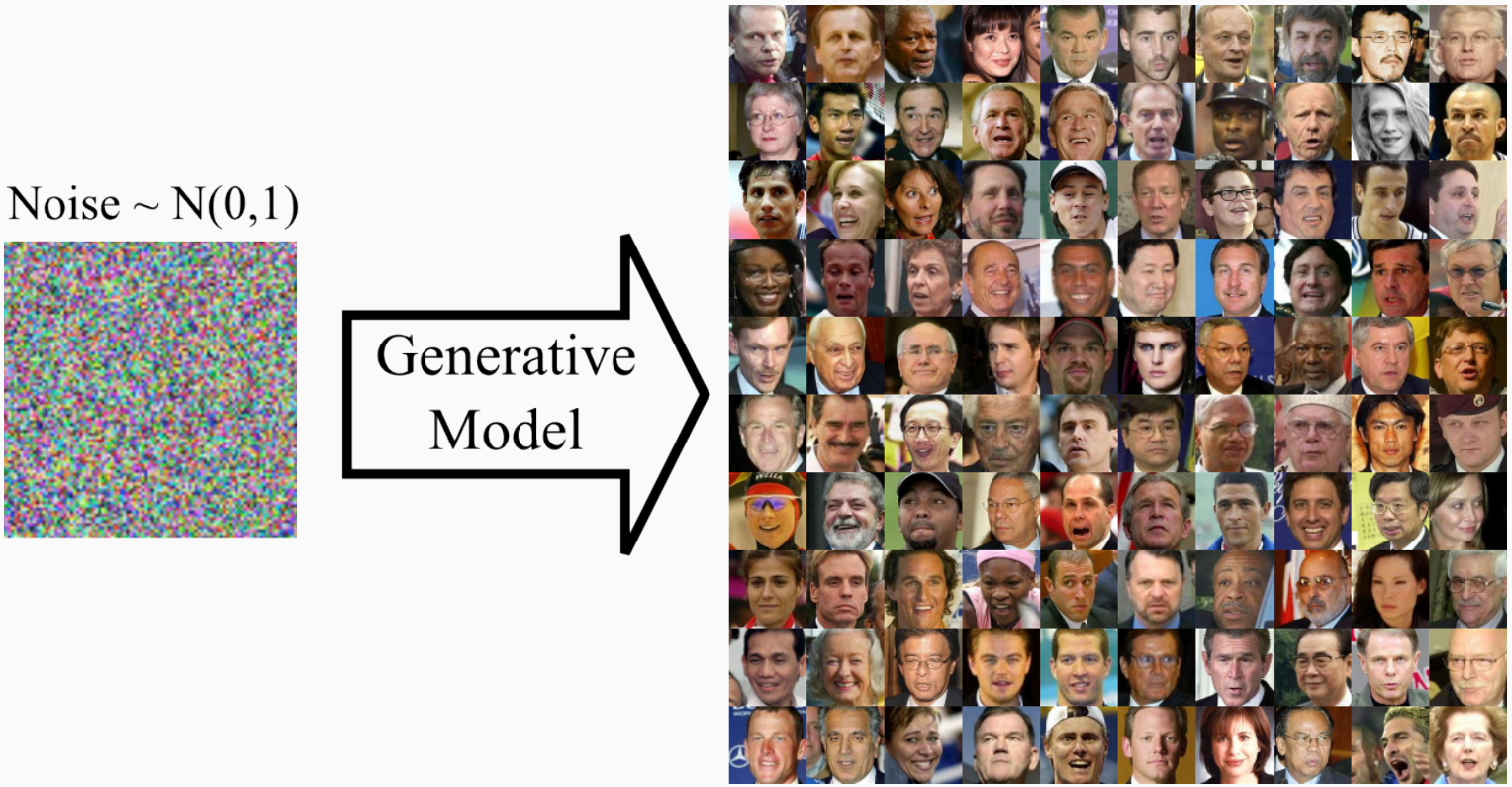
Credits:

Most of the slides from **Charles Deledalle's** course "UCSD ECE285 Machine learning for image processing" (30 $\times$ 50 minutes course)

`www.charles-deledalle.fr/`

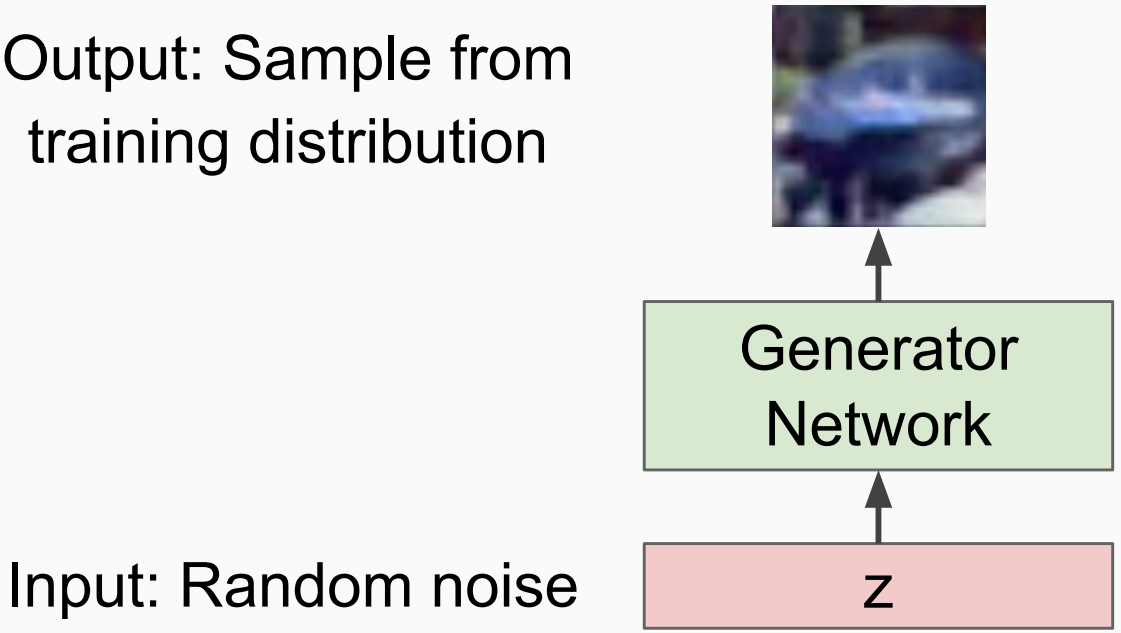
`https://www.charles-deledalle.fr/pages/teaching.php#learning`

Image generation – Beyond Gaussian models



Generative Adversarial Networks (GAN)

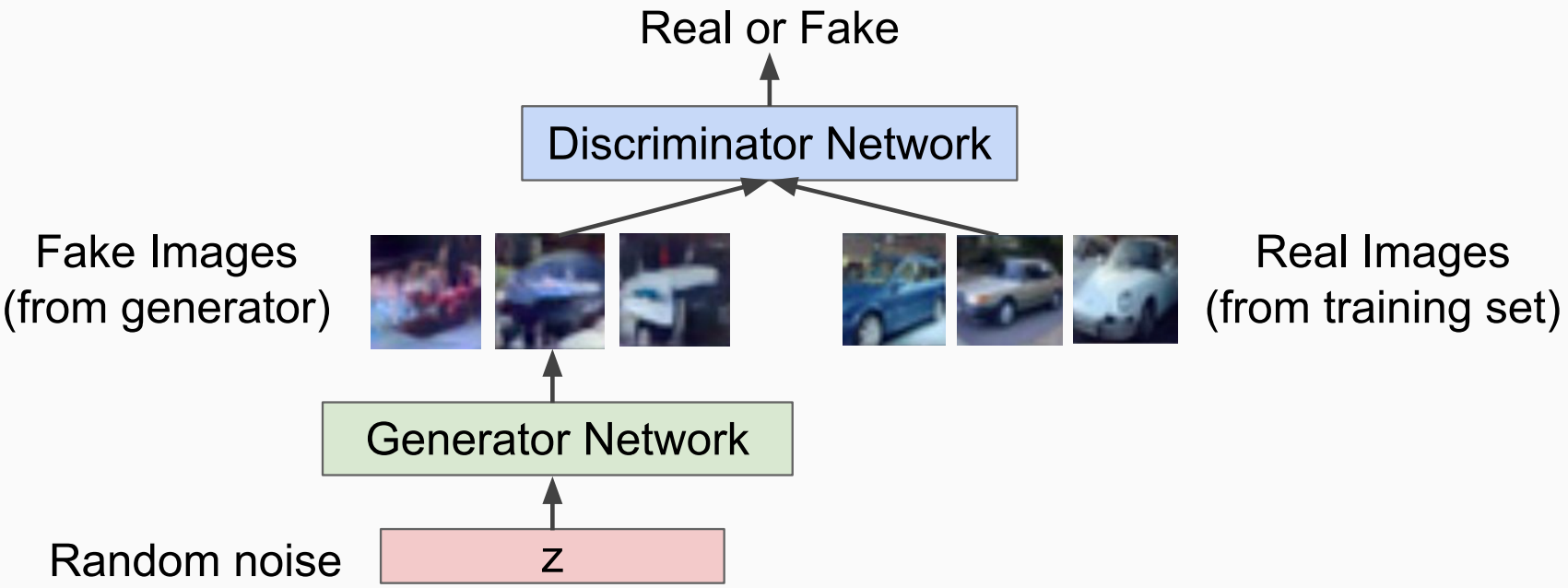
- **Goal:** design a complex model with high capacity able to map latent random noise vectors $z \in \mathbb{R}^k$ to a realistic image $x \in \mathbb{R}^d$.
- **Idea:** Take a deep neural network



- **What about the loss?** Measure if the generated image is **photo-realistic**.

Generative Adversarial Networks (GAN)

Define a loss measuring how much you can fool a classifier that has learned to distinguish between real and fake images.



- **Discriminator network:** Try to distinguish between real and fake images.
- **Generator network:** Fool the discriminator by generating realistic images.

Recap on binary classification

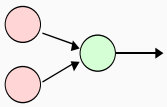
- Given a labeled dataset

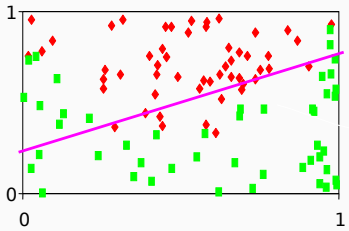
$$\mathcal{D} = \{(\boldsymbol{x}^{(i)}, y^{(i)}),\ i = 1, \ldots, N\} \subset \mathbb{R}^d \times \{0, 1\}$$

  with **binary labels** $y^{(i)} \in \{0, 1\}$ that corresponds to two classes C_0 and C_1.
- A **parametric classifier** $f_{\boldsymbol{\theta}} : \mathbb{R}^d \to [0, 1]$ outputs a probability such that

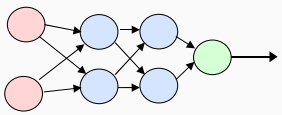
$$\boldsymbol{p} = f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \mathbb{P}(\boldsymbol{x} \in C_1) \quad \text{and} \quad 1 - \boldsymbol{p} = 1 - f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \mathbb{P}(\boldsymbol{x} \in C_0)$$

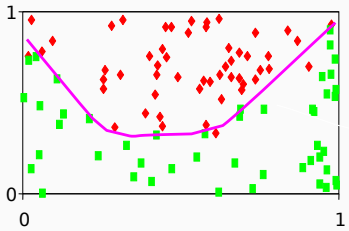
1 neuron



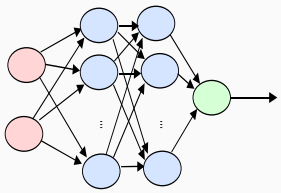


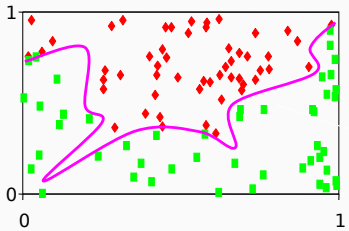
2+2+1 neurons





10+10+1 neurons





Estimated decision regions:
$\hat{C}_1 = \{\boldsymbol{x} \in \mathbb{R}^d,\ f_{\boldsymbol{\theta}}(\boldsymbol{x}) \geq \frac{1}{2}\}$ and
$\hat{C}_0 = \mathbb{R}^d \setminus \hat{C}_1$.

Complexity/capacity of the network

$\Rightarrow$

Trade-off between generalization and overfitting.

Recap on binary classification

- **Training:** Logistic regression for binary classification: Maximum likelihood of the dataset (opposite of binary crosss-entropy : `BCELoss` in PyTorch):

$$\max_{\boldsymbol{\theta}} \sum_{i=1}^{N} \left[y^{(i)} \log f_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}) + (1 - y^{(i)}) \log \left(1 - f_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}) \right) \right]$$

- For neural networks, the probability $f_{\boldsymbol{\theta}}$ is obtained using the **sigmoid function** $\sigma(t) = \dfrac{e^t}{1+e^t} = \dfrac{1}{1+e^{-t}}$ as the activation function of the last layer.
- Beware that $y^{(i)} = 0$ or 1 so only one term is non-zero.
- One could instead regroup the terms of the sum according to the label values:

$$\max_{\boldsymbol{\theta}} \sum_{\substack{(\boldsymbol{x}^{(i)}, y^{(i)}) \in \mathcal{D} \\ \text{s.t. } y^{(i)}=1}}^{N} \log f_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}) + \sum_{\substack{(\boldsymbol{x}^{(i)}, y^{(i)}) \in \mathcal{D} \\ \text{s.t. } y^{(i)}=0}}^{N} \log \left(1 - f_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}) \right)$$

Generative Adversarial Networks (GAN)

- **Discriminator network:** Consider two sets
  - $\mathcal{D}_{\text{real}}$: a dataset of n real images **(real = labeled with $y^{(i)} = 1$)**,
  - $\mathcal{D}_{\text{fake}}$: a dataset of m fake images $\boldsymbol{x} = G_{\theta_g}(z)$ **(fake = labeled with $y^{(i)} = 0$)**.

- **Goal:** Find the parameters θ_d of a binary classification network $\boldsymbol{x} \mapsto D_{\theta_d}(\boldsymbol{x})$ meant to classify real and fake images.
  Minimize the binary cross-entropy, or maximize its negation

$$\max_{\theta_d} \underbrace{\sum_{\boldsymbol{x}_{\text{real}} \in \mathcal{D}_{\text{real}}} \log D_{\theta_d}(\boldsymbol{x}_{\text{real}})}_{\substack{\text{force predicted labels to be 1} \\ \text{for real images}}} + \underbrace{\sum_{\boldsymbol{x}_{\text{fake}} \in \mathcal{D}_{\text{fake}}} \log(1 - D_{\theta_d}(\boldsymbol{x}_{\text{fake}}))}_{\substack{\text{force predicted labels to be 0} \\ \text{for fake images}}}$$

- **How:** Use gradient ascent with backprop (+ batch-normalization. . .).

Generative Adversarial Networks (GAN)

- **Generator network:** Consider a given discriminative model $\boldsymbol{x} \mapsto D_{\theta_d}(\boldsymbol{x})$ and consider $\mathcal{D}_{\text{rand}}$ a set of m random latent vectors.
- **Goal:** Find the parameters θ_g of a network $\boldsymbol{z} \mapsto G_{\theta_g}(\boldsymbol{z})$ generating images from random vectors $\boldsymbol{z}$ such that it fools the discriminator

$$\min_{\theta_g} \underbrace{\sum_{\boldsymbol{z} \in \mathcal{D}_{\text{rand}}} \log(1 - D_{\theta_d}(G_{\theta_g}(\boldsymbol{z})))}_{\substack{\text{force the discriminator to think that} \\ \text{our generated fake images are not fake (away from 0)}}} \tag{1}$$

or alternatively (works better in practice)

$$\max_{\theta_g} \underbrace{\sum_{\boldsymbol{z} \in \mathcal{D}_{\text{rand}}} \log D_{\theta_d}(G_{\theta_g}(\boldsymbol{z}))}_{\substack{\text{force the discriminator to think that} \\ \text{our generated fake images are real (close to 1)}}} \tag{2}$$

- **How:** Gradient descent for (1) or gradient ascent for (2) with backprop. . .

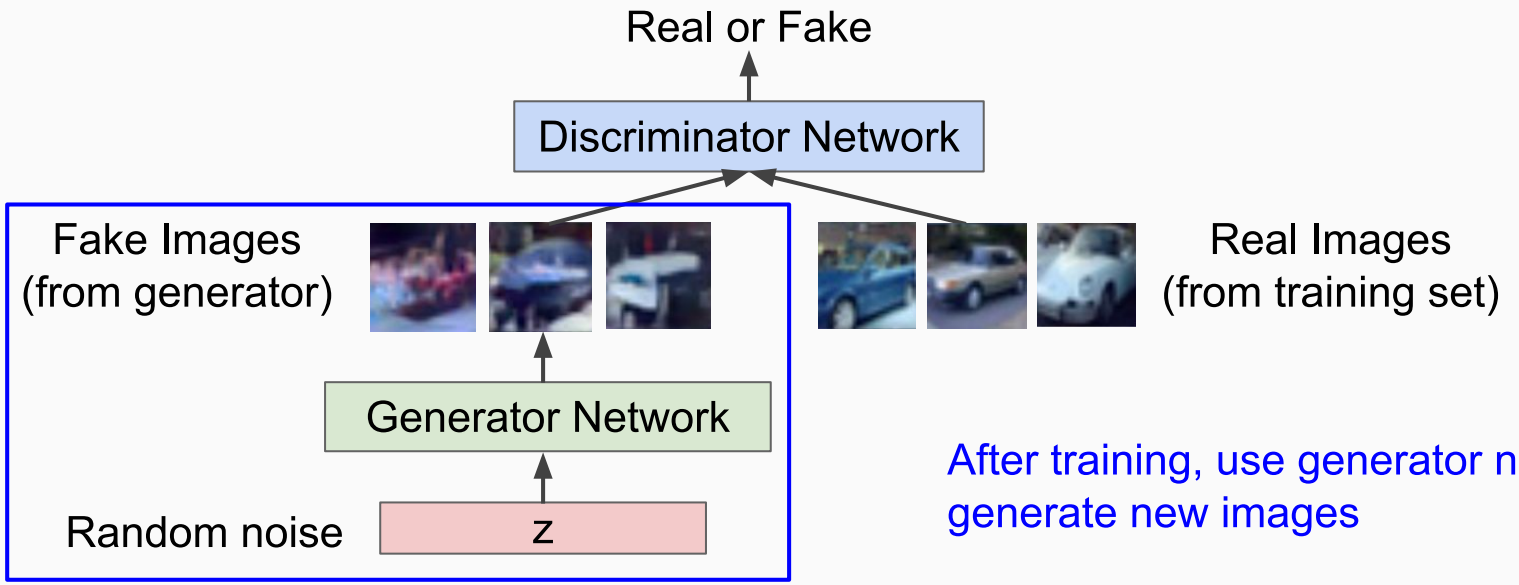
Generative Adversarial Networks (GAN)

- **Train both networks jointly**.
- Minimax loss in a two player game (each player is a network):

$$\min_{\theta_g} \max_{\theta_d} \sum_{\boldsymbol{x} \in \mathcal{D}_{\text{real}}} \log D_{\theta_d}(\boldsymbol{x}) + \sum_{z \in \mathcal{D}_{\text{rand}}} \log(1 - D_{\theta_d}(\underbrace{G_{\theta_g}(\boldsymbol{z})}_{\text{fake}}))$$

- **Algorithm:** Repeat until convergence
  1. Fix θ_g, update θ_d with one step of gradient ascent,
  2. Fix θ_d, update θ_g with one step of gradient descent for (1),
     (or one step of gradient ascent for (2).)

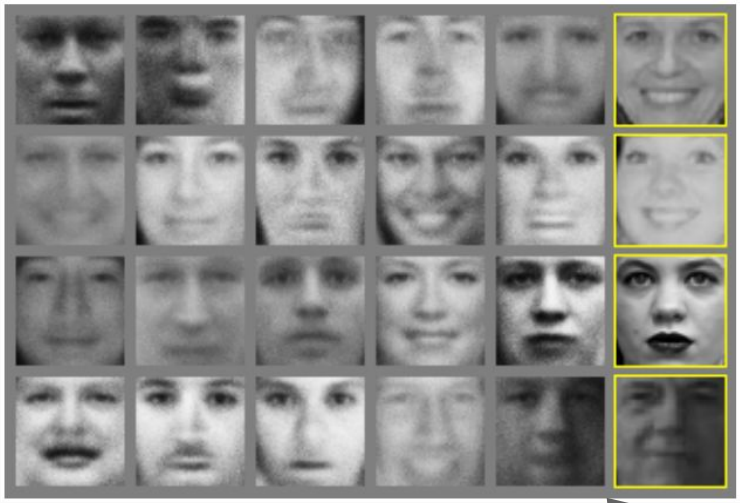
Generative Adversarial Networks (GAN)



After training, use generator network to generate new images

Generative Adversarial Networks (GAN)

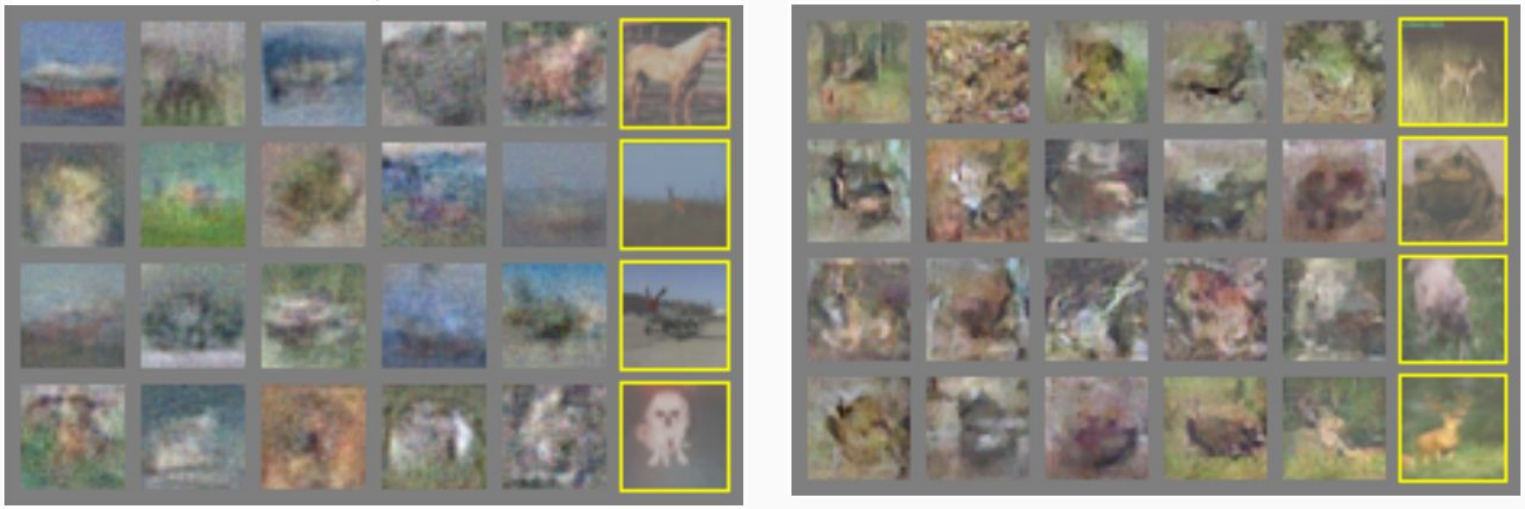
Generated samples



Nearest neighbor from training set

Generative Adversarial Networks (GAN)

Generated samples (CIFAR-10)

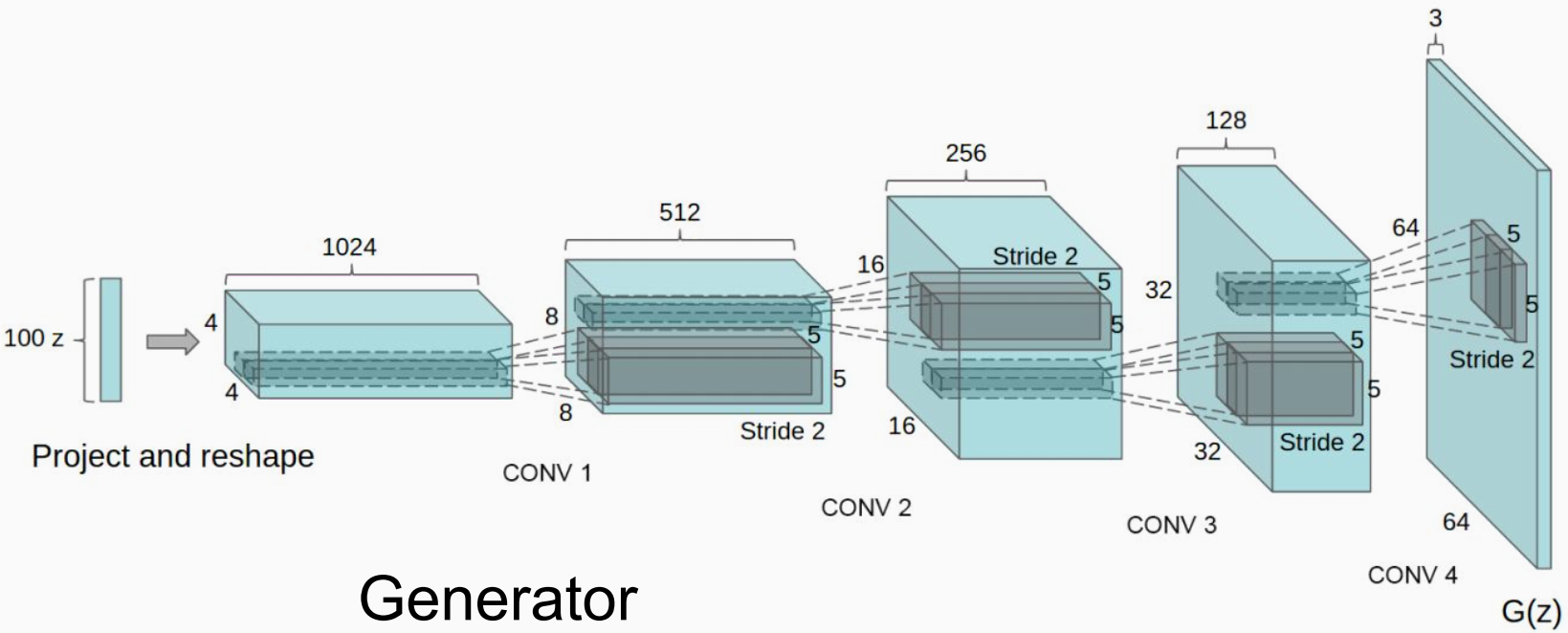


Nearest neighbor from training set

Convolutional GAN

(Radford et al., 2016)

- **Generator:** upsampling network with fractionally strided convolutions,
- **Discriminator:** convolutional network with strided convolutions.



Generator

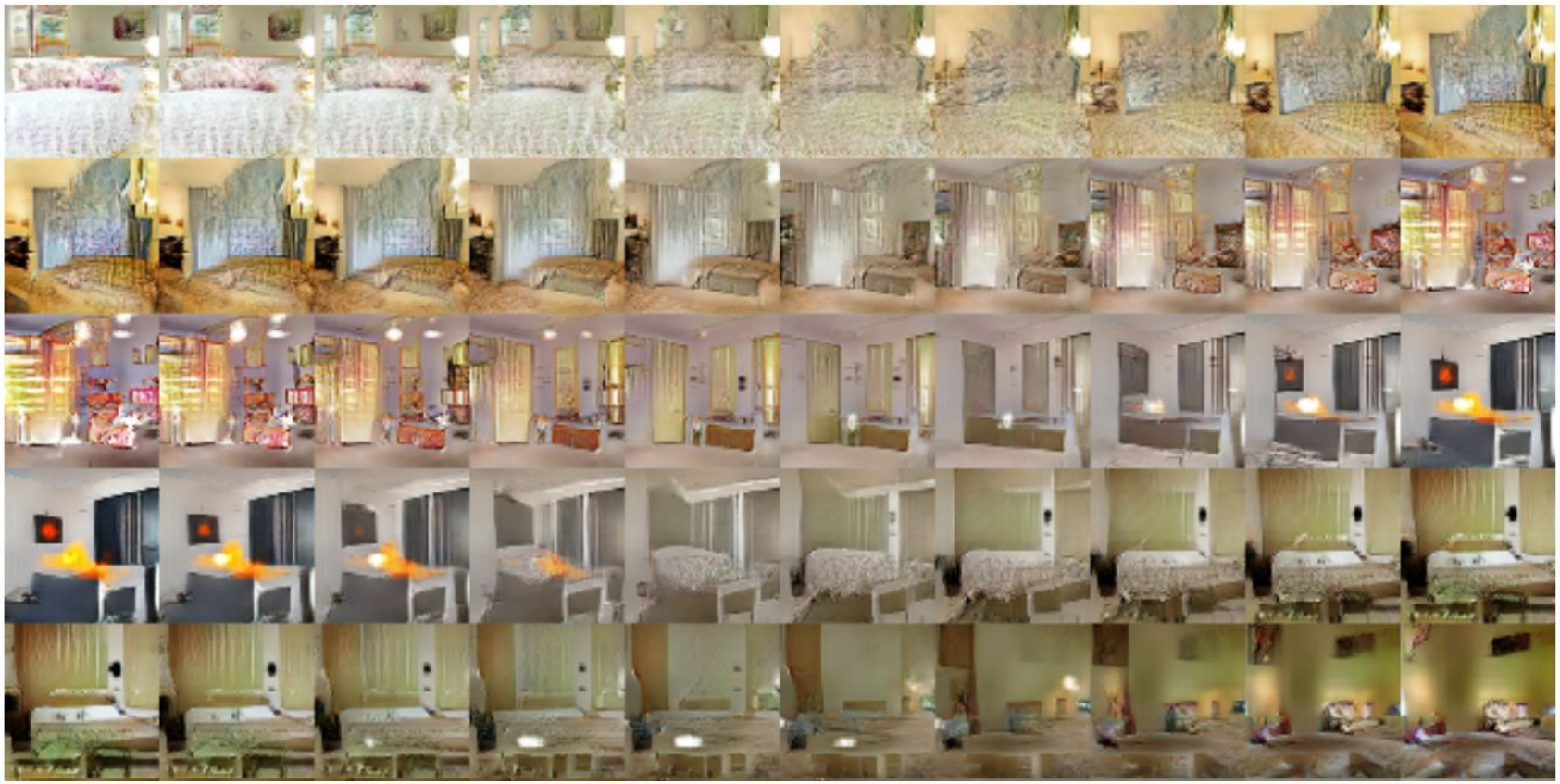
Convolutional GAN

(Radford et al., 2016)

Generations of realistic bedrooms pictures, from randomly generated latent variables.

Convolutional GAN

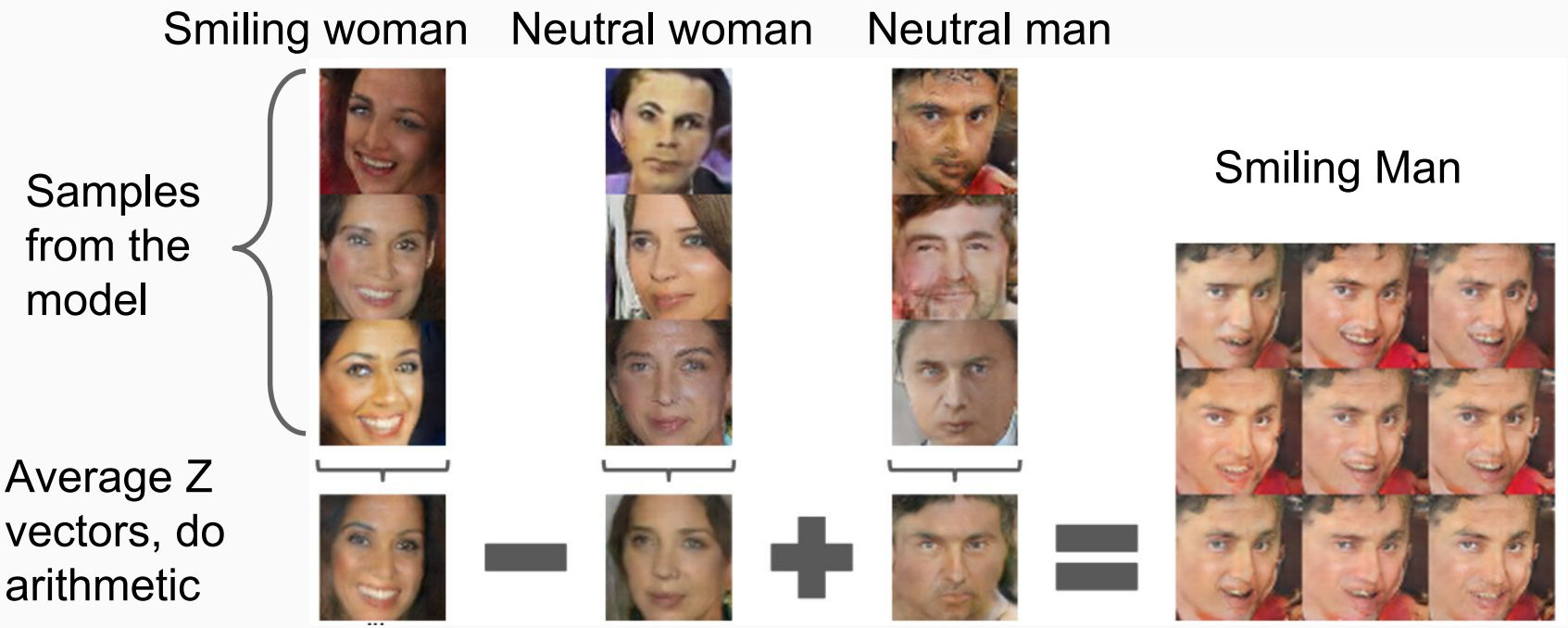
(Radford et al., 2016)



Interpolation in between points in latent space.

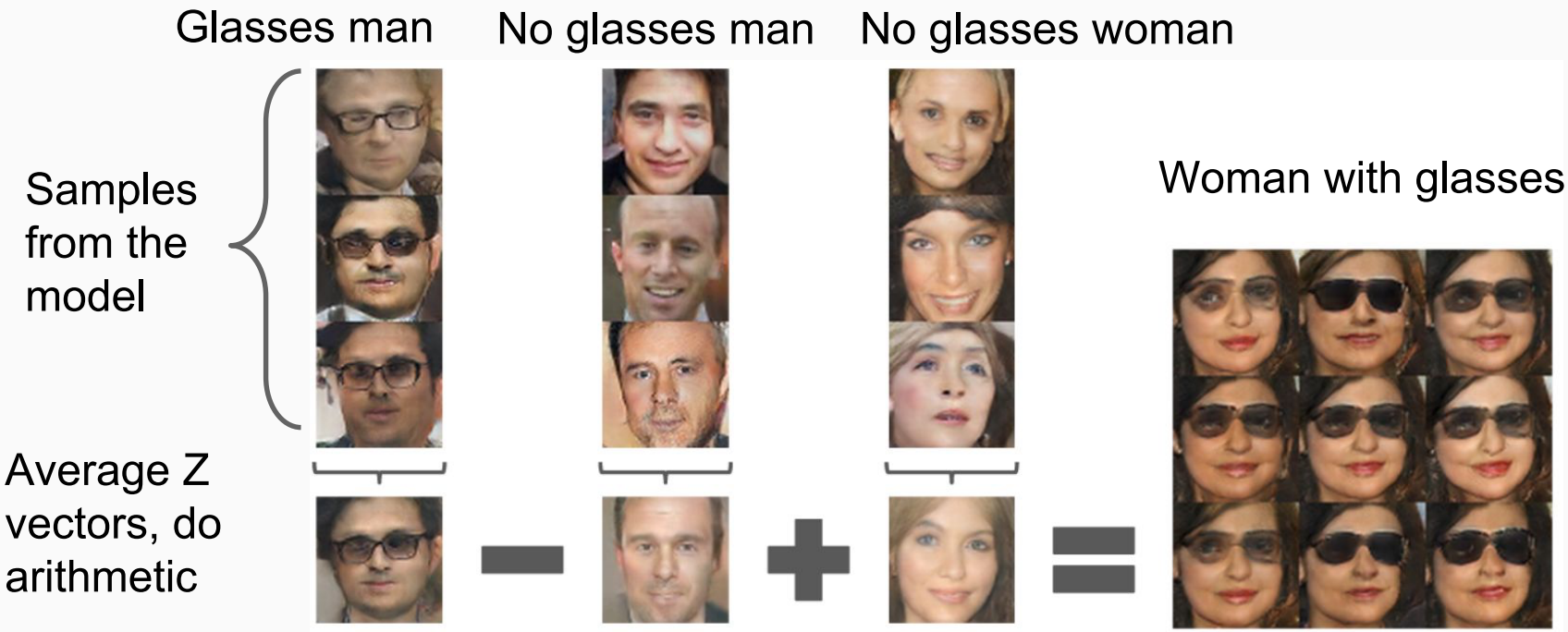
Convolutional GAN – Arithmetic

(Radford et al., 2016)



Convolutional GAN – Arithmetic

(Radford et al., 2016)



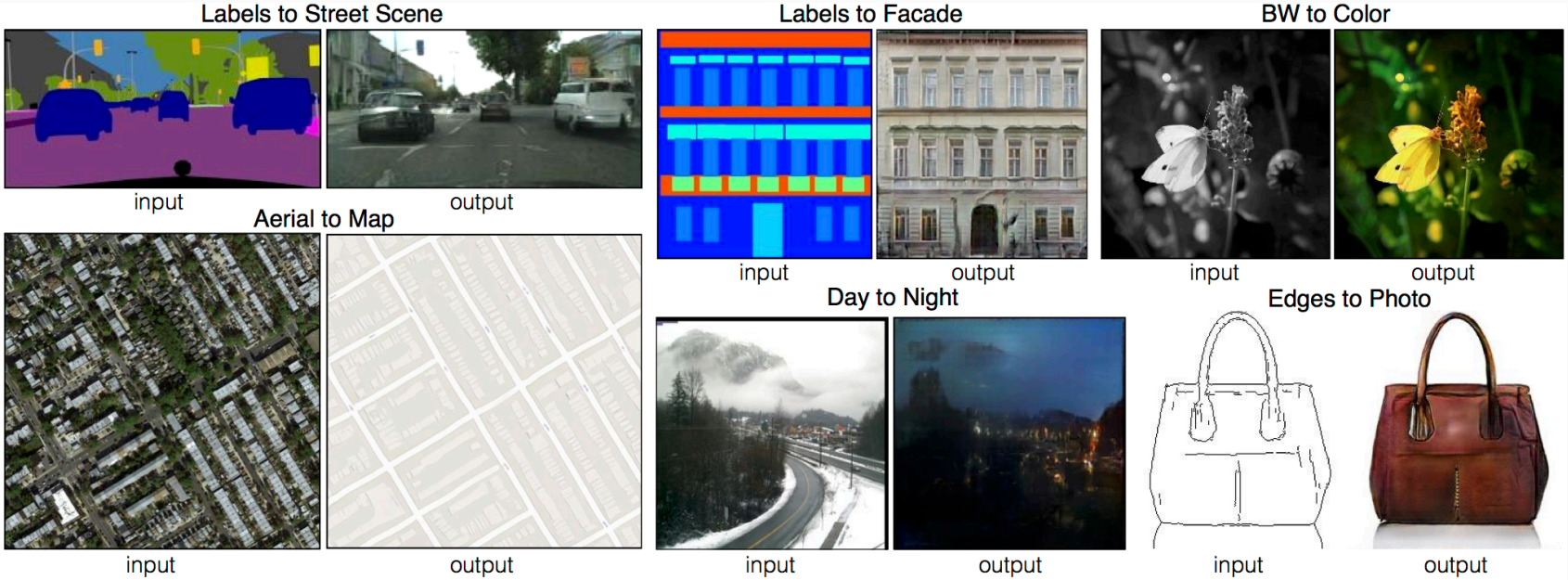
Generative Aversarial Networks: Style GAN (Karras et al., 2019)

Image size:
1024× 1024 px
(source: Karras et al.)

Conditional GAN

Pix2pix: Image-to-Image Translation with Conditional Adversarial Nets

(Isola et al., 2017)



- GAN conditioned on input image.
- Generator: U-net architecture
- Discriminator: Patch discriminator applied to each patch
- Opens the way for new creative tools

(source: Isola et al.)

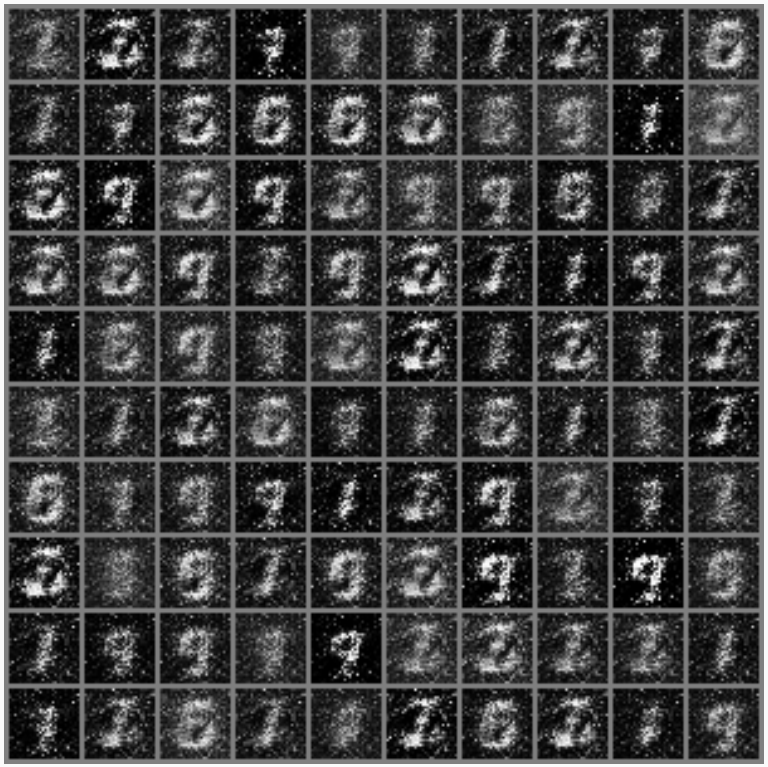
GAN Training

Training with MNIST (60 000 images)

- Adam optimizer
- Learning rate 0.0002 for both the discriminator and the generator

Real images:

Fake images, epoch 1:



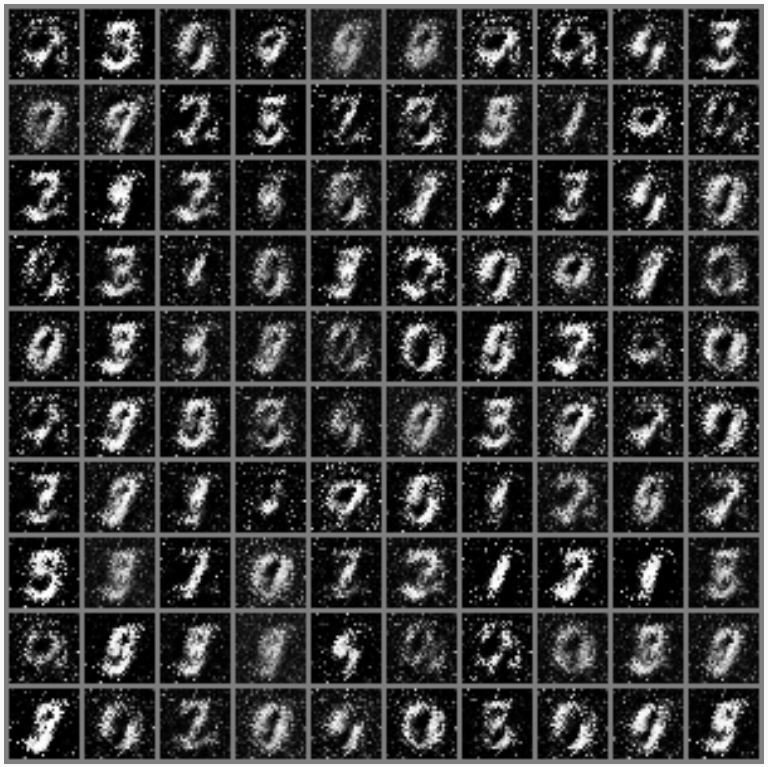
GAN Training

Training with MNIST (60 000 images)

- Adam optimizer
- Learning rate 0.0002 for both the discriminator and the generator

Real images:

Fake images, epoch 2:



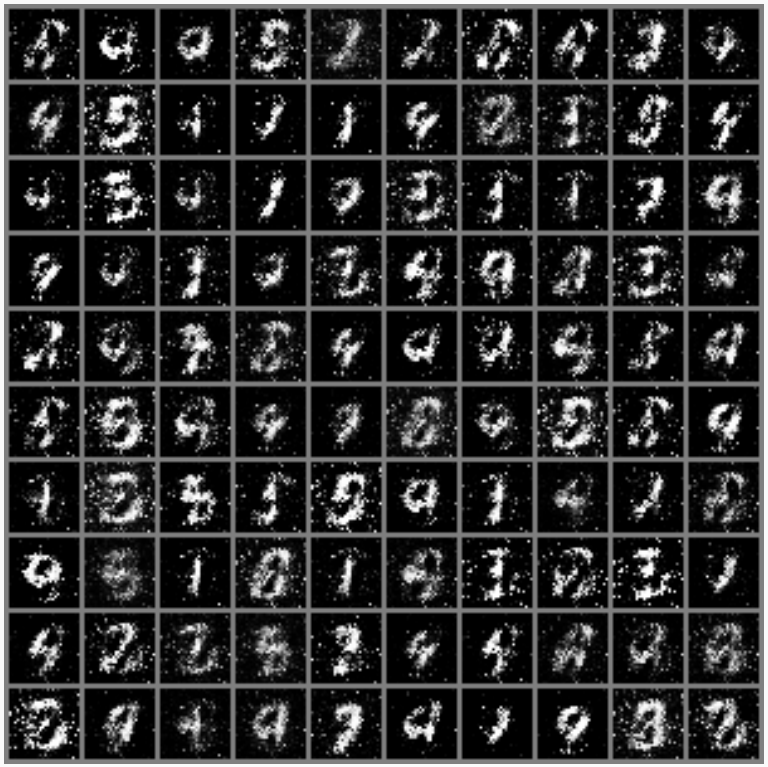
GAN Training

Training with MNIST (60 000 images)

- Adam optimizer
- Learning rate 0.0002 for both the discriminator and the generator

Real images:

Fake images, epoch 3:



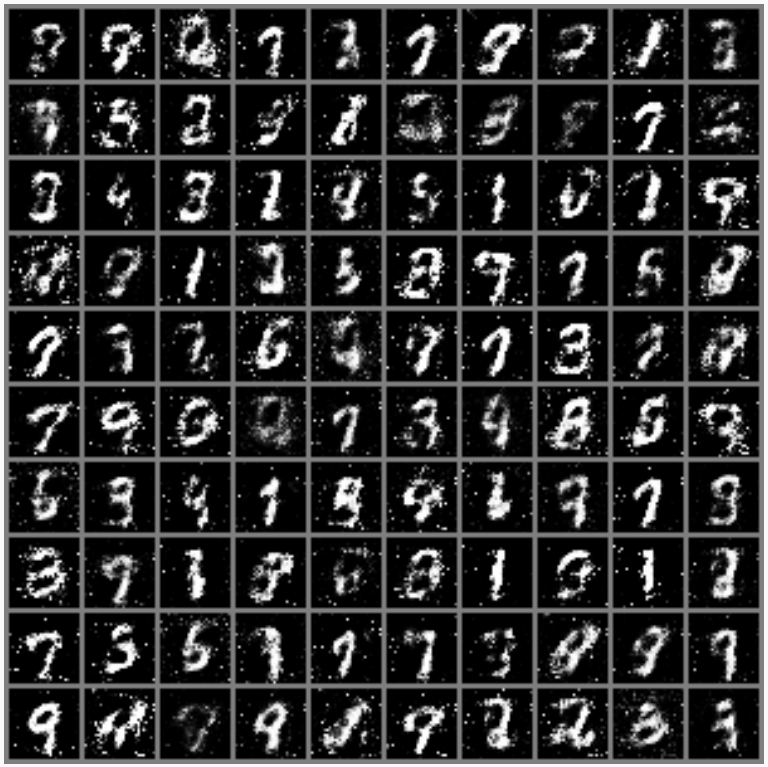
GAN Training

Training with MNIST (60 000 images)

- Adam optimizer
- Learning rate 0.0002 for both the discriminator and the generator

Real images:

Fake images, epoch 10:



GAN Training

Training with MNIST (60 000 images)

- Adam optimizer
- Learning rate 0.0002 for both the discriminator and the generator

Real images:

Fake images, epoch 100:

GAN Training

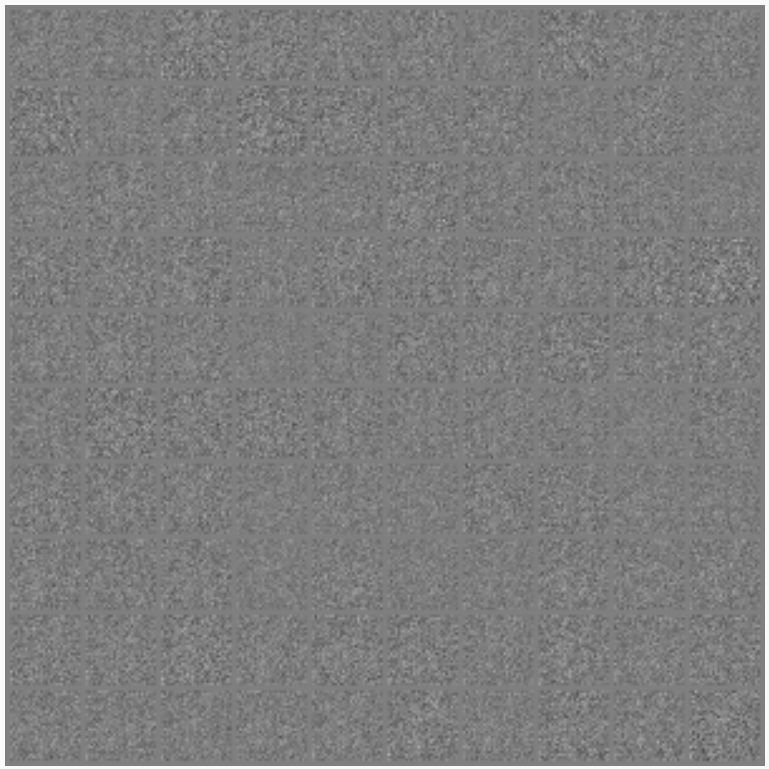
Training GANs is quite instable.

The generator can suffer *mode collapse*: It always produce the same image (one mode only).

Same as before **but with SGD**.

Real images:

Fake images, epoch 1:



GAN Training

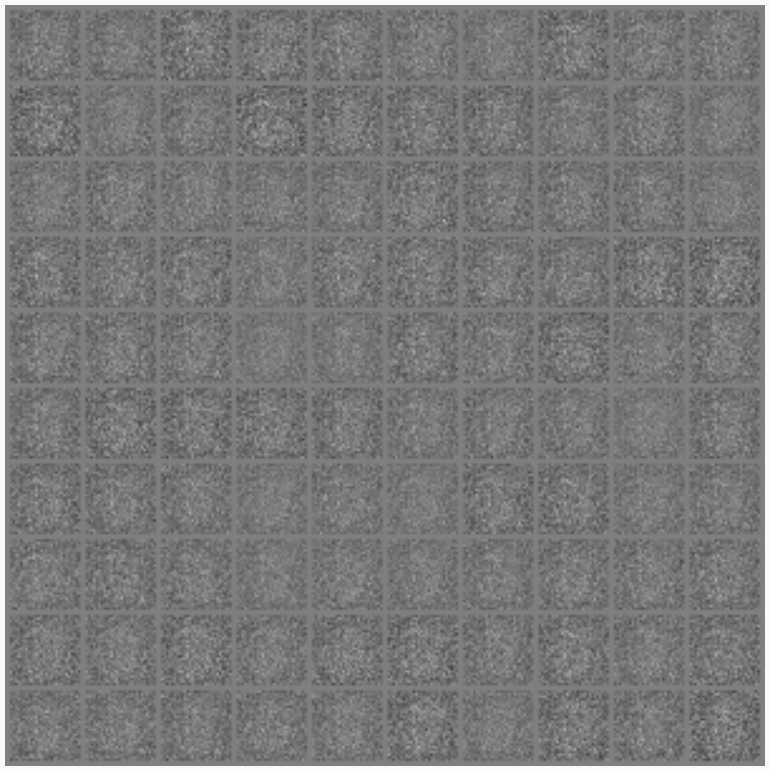
Training GANs is quite instable.

The generator can suffer *mode collapse*: It always produce the same image (one mode only).

Same as before **but with SGD**.

Real images:

Fake images, epoch 2:



GAN Training

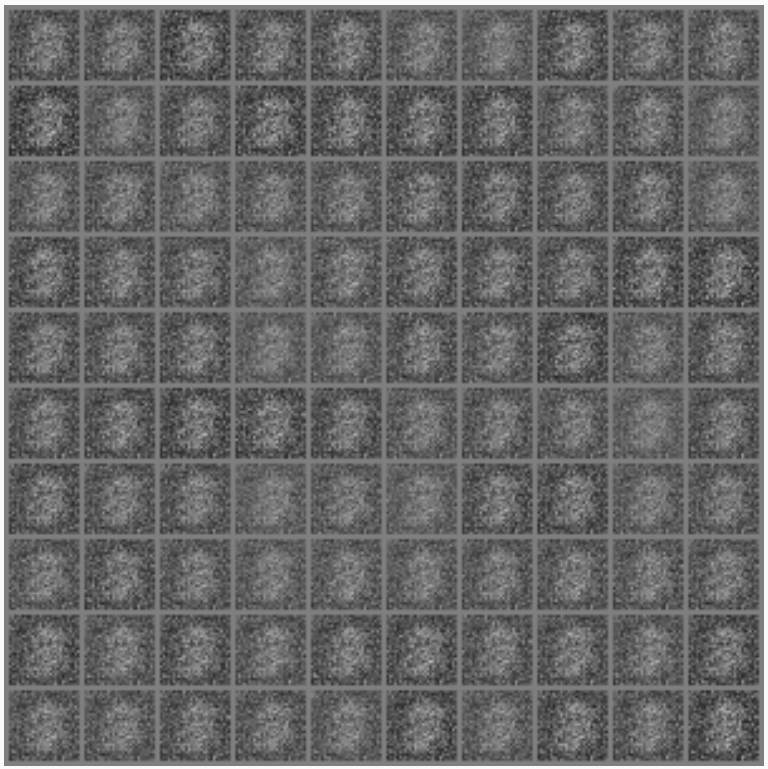
Training GANs is quite instable.

The generator can suffer *mode collapse*: It always produce the same image (one mode only).

Same as before **but with SGD**.

Real images:

Fake images, epoch 3:



GAN Training

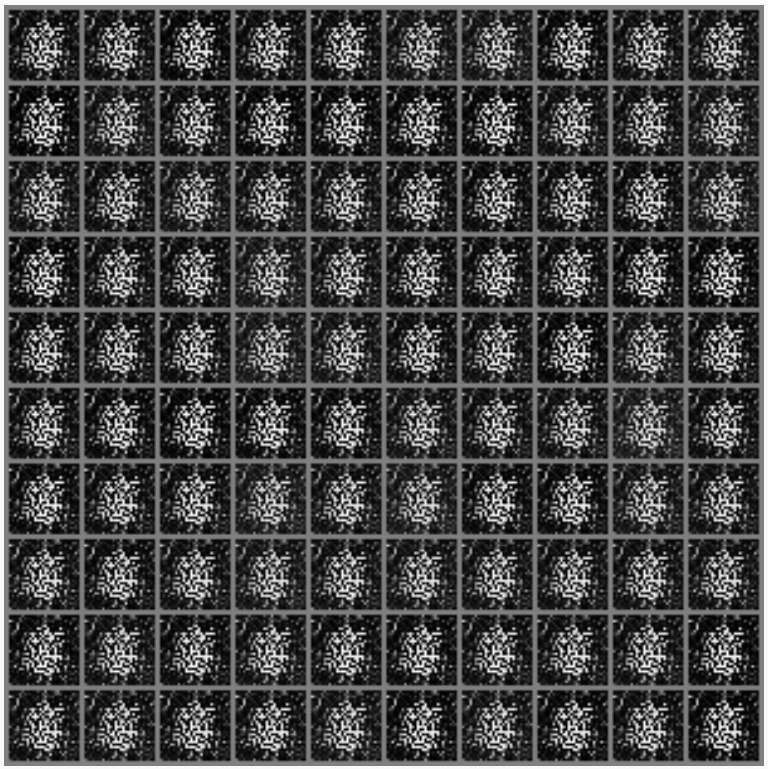
Training GANs is quite instable.

The generator can suffer *mode collapse*: It always produce the same image (one mode only).

Same as before **but with SGD**.

Real images:

Fake images, epoch 10:



GAN Training

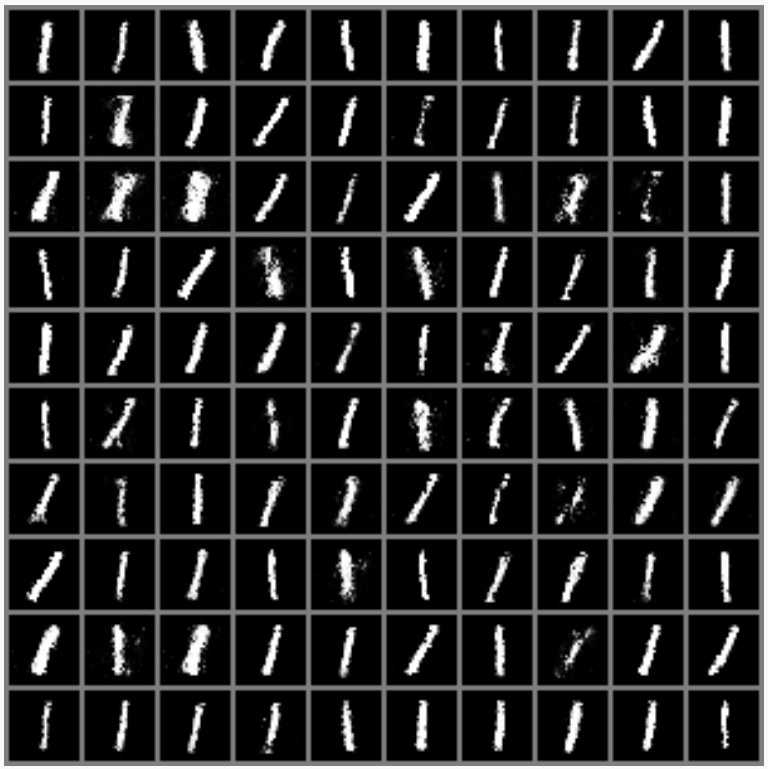
Training GANs is quite instable.

The generator can suffer *mode collapse*: It always produce the same image (one mode only).

Same as before **but with SGD**.

Real images:

Fake images, epoch 100:



GAN Training

Some heuristics inspired by optimal transport theory have been proposed and called **Wassertein GAN** (Arjovsky et al., 2017) (Gulrajani et al., 2017).

Optimal transport

- The optimal transport theory provides mathematical tools to compare or interpolate between probability distributions.
- Given two probability distributions μ_0 and μ_1 in $\mathcal{P}_2(\mathbb{R}^d)$ (the set of probability measures with finite second moments on $\mathbb{R}^d$), and a positive cost function $c : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^+$

$$MK_c(\mu_0, \mu_1) := \inf_{\gamma \in \Pi(\mu_0, \mu_1)} \int_{\mathbb{R}^d \times \mathbb{R}^d} c(y_0, y_1) d\gamma(y_0, y_1),$$

where $\Pi(\mu_0, \mu_1)$ is the set of probability distributions γ on $\mathbb{R}^d \times \mathbb{R}^d$ with marginal distributions μ_0 and μ_1.

Proposition (Duality)

$$MK_c(\mu_0, \mu_1) = \sup_{\phi, \psi \in \Phi_c(\mu_0, \mu_1)} \int \phi d\mu_0 + \int \psi d\mu_1,$$

where

$$\Phi_c(\mu_0, \mu_1) = \{\phi, \psi \in C_b(\mathbb{R}^d) \textit{ s.t. } \forall x, y,\ \phi(x) + \psi(y) \leq c(x, y)\}.$$

Wasserstein distances: When using $c(x,y) = \|x-y\|^p$ one defines Wasserstein distances:

Definition
The *p-Wasserstein distance* W_p between μ_0 and μ_1 is defined as

$$W_p^p(\mu_0,\mu_1) := \inf_{Y_0\sim\mu_0; Y_1\sim\mu_1} \mathbb{E}\left(\|Y_0 - Y_1\|^p\right) = \inf_{\gamma\in\Pi(\mu_0,\mu_1)} \int_{\mathbb{R}^d\times\mathbb{R}^d} \|y_0 - y_1\|^p d\gamma(y_0,y_1).$$

1-Wasserstein distance and duality: See eg (Santambrogio, 2015)

For $p=1$, one has

$$W_1(\mu_0,\mu_1) = \sup_{\phi\in \mathrm{Lip}_1} \int \phi d\mu_0 - \int \phi d\mu_1 = \sup_{\phi\in \mathrm{Lip}_1} \mathbb{E}_{\boldsymbol{x}\sim\mu_0}(\phi(\boldsymbol{x})) - \mathbb{E}_{\boldsymbol{x}\sim\mu_1}(\phi(\boldsymbol{x}))$$

where

$$\mathrm{Lip}_1 = \{f : \mathbb{R}^d \to \mathbb{R},\ \text{s.t. } \forall x,y,\ |f(x)-f(y)| \le \|x-y\|\}.$$

Back to GANs:

- The role of the discriminator D is to differentiate the distribution $p_{\text{real}}(\boldsymbol{x})$ of real images from the distribution $p_{\text{gen}}(\boldsymbol{x})$ of generated images.
- Ideally, one would like to optimize the generator to minimize $W_1(p_{\text{real}}, p_{\text{gen}})$.
- However it is not possible to compute this Wasserstein distance W_1 because taking the $\sup$ over Lip_1 is not tractable.
- (Arjovsky et al., 2017) proposes to restrict Lip_1 to the set of Lip_1 functions that are parameterized with some neural network:

$$\begin{aligned} W_1(p_{\text{real}}, p_{\text{gen}}) &= \sup_{\phi \in \text{Lip}_1} \mathbb{E}_{\boldsymbol{x} \sim p_{\text{real}}}(\phi(\boldsymbol{x})) - \mathbb{E}_{\boldsymbol{x} \sim p_{\text{gen}}}(\phi(\boldsymbol{x})) \\ &\geq \sup_{D_{\theta_d} \in \text{Lip}_1} \mathbb{E}_{\boldsymbol{x} \sim p_{\text{real}}}(D_{\theta_d}(\boldsymbol{x})) - \mathbb{E}_{\boldsymbol{x} \sim p_{\text{gen}}}(D_{\theta_d}(\boldsymbol{x})) \end{aligned}$$

Wassertsein GAN

GAN (Vanilla):

$$\min_{\theta_g} \max_{\theta_d} \sum_{\boldsymbol{x} \in \mathcal{D}_{\text{real}}} \log D_{\theta_d}(\boldsymbol{x}) + \sum_{z \in \mathcal{D}_{\text{rand}}} \log(1 - D_{\theta_d}(G_{\theta_g}(\boldsymbol{z}))$$

Wassestein GAN:

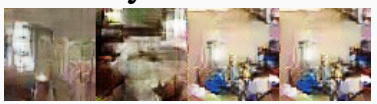
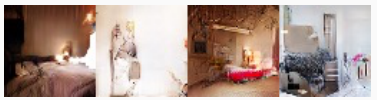
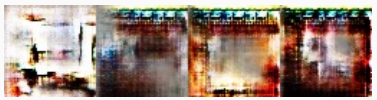
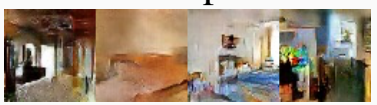
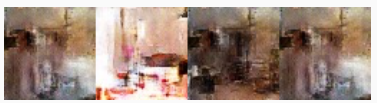
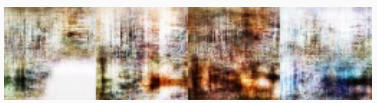
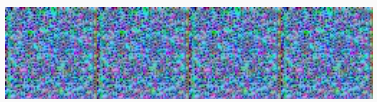
$$\min_{\theta_g} \max_{\substack{\theta_d \text{ s.t.} \\ D_{\theta_d} \in \text{Lip}_1}} \sum_{\boldsymbol{x} \in \mathcal{D}_{\text{real}}} D_{\theta_d}(\boldsymbol{x}) - \sum_{z \in \mathcal{D}_{\text{rand}}} D_{\theta_d}(G_{\theta_g}(\boldsymbol{z}))$$

- We just got read of the $\log$... but have a constrained optimization.
- It is known that optimal potential ϕ satisfy $\|\nabla_{\boldsymbol{x}} \phi(\boldsymbol{x})\| = 1$ (Santambrogio, 2015)
- (Gulrajani et al., 2017) propose to use this property by minimizing:

$$\min_{\theta_g} \max_{\theta_d} \sum_{\boldsymbol{x} \in \mathcal{D}_{\text{real}}} D_{\theta_d}(\boldsymbol{x}) - \sum_{z \in \mathcal{D}_{\text{rand}}} D_{\theta_d}(G_{\theta_g}(\boldsymbol{z})) + \lambda \sum_{\boldsymbol{x}_t} (\|\nabla_{\boldsymbol{x}} \phi(\boldsymbol{x}_t)\| - 1)^2$$

where each $\boldsymbol{x}_t$ is a point from a segment joining a real and a fake image.

Wassertsein GAN

DCGAN	LSGAN	WGAN (clipping)	WGAN-GP (ours)
Baseline (G: DCGAN, D: DCGAN)			
G: No BN and a constant number of filters, D: DCGAN			
G: 4-layer 512-dim ReLU MLP, D: DCGAN			
No normalization in either G or D			
Gated multiplicative nonlinearities everywhere in G and D			
$tanh$ nonlinearities everywhere in G and D			
101-layer ResNet G and D			

Wassertsein GAN

Wassertsein GAN using the gradient penalty is a more stable way to train deep convolutional generators/discriminators.

References

References

Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein generative adversarial networks. In Precup, D. and Teh, Y. W., editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 214–223. PMLR.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., and Weinberger, K., editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. (2017). Improved training of wasserstein gans. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

References ii

Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Karras, T., Laine, S., and Aila, T. (2019). A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Radford, A., Metz, L., and Chintala, S. (2016). Unsupervised representation learning with deep convolutional generative adversarial networks. In Bengio, Y. and LeCun, Y., editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Santambrogio, F. (2015). *Optimal Transport for Applied Mathematicians: Calculus of Variations, PDEs, and Modeling*. Birkhäuser.