Deep Learning:

Generative Adversarial Networks

Ozan Özdenizci

Institute of Theoretical Computer Science

ozan.ozdenizci@igi.tugraz.at

Deep Learning VO - WS 23/24

Lecture 10 - December 11th, 2023

Today

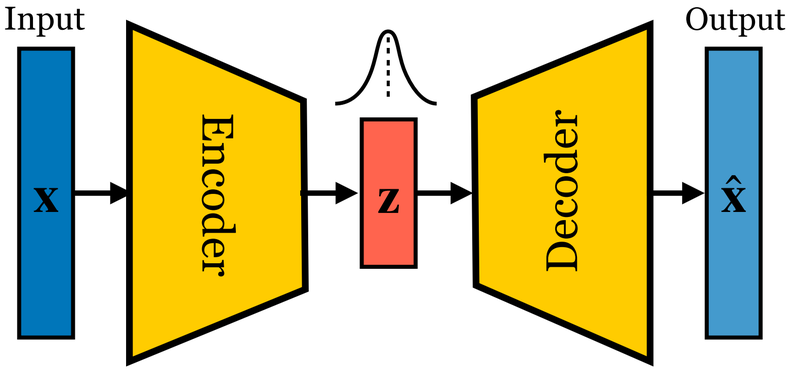
- [ ] Deep Generative Models
  - [x] Variational Autoencoders
  - [ ] Generative Adversarial Networks
    - [ ] Architecture & Training
    - [ ] Mode Collapse
    - [ ] Further Extensions & Variants

Ian Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014.

Recap: Deep Generative Models

- Given a list of data points $X = \langle \mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)} \rangle$ coming from a data distribution $p(\mathbf{x})$.
- **Generative model:** a model $p_\theta(\mathbf{x})$ for this distribution, from which we can sample.

Variational Autoencoders



Generative Adversarial Nets

Today

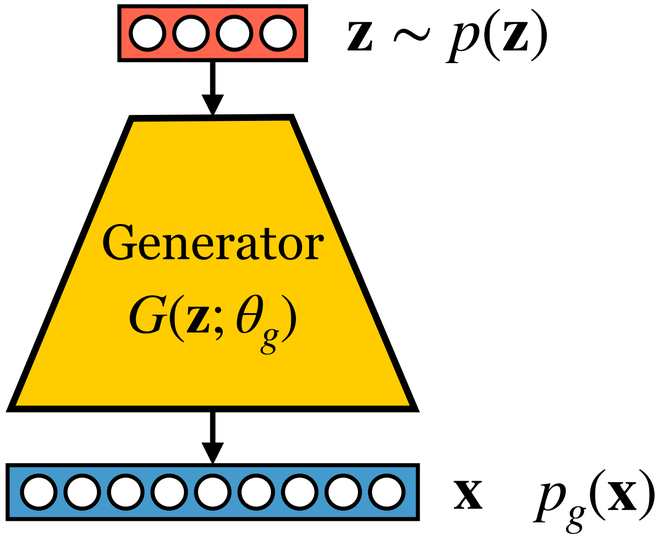
Generative Stochastic Networks

Uses feed-forward networks to generate samples.

Generator network: G

Network input: A random vector $\mathbf{z}$ from some simple prior distribution $p(\mathbf{z})$.

Network output: $G(\mathbf{z};\theta_g)$: A sample $\mathbf{x}$ from a complex distribution $p_g(\mathbf{x})$.



Generative Adversarial Networks (GANs)

Add a discriminator that tries to distinguish real data from generated data.

Generator network: G

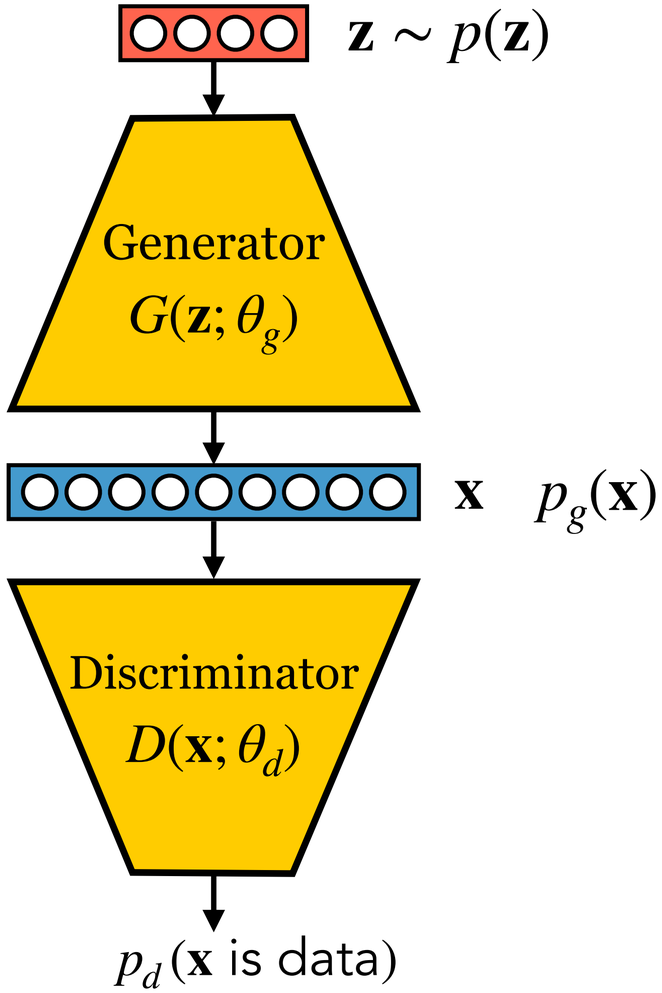
Network input: A random vector $\mathbf{z}$ from some simple prior distribution $p(\mathbf{z})$.

Network output: $G(\mathbf{z};\theta_g)$: A sample $\mathbf{x}$ from a complex distribution $p_g(\mathbf{x})$.

Discriminator network: D

Network input: $\mathbf{x}$, either coming from G or from $p_{data}(\mathbf{x})$

Network output: $D(\mathbf{x};\theta_d)$: probability $p(\mathbf{x}$ is an actual data point$)$



Training of GANs

Discriminator network: D is a classifier that performs binary classification.

$$\text{target} = \begin{cases} 1, & \text{if real data point} \\ 0, & \text{if generated} \end{cases}$$

Generator network: G is trained to minimize:

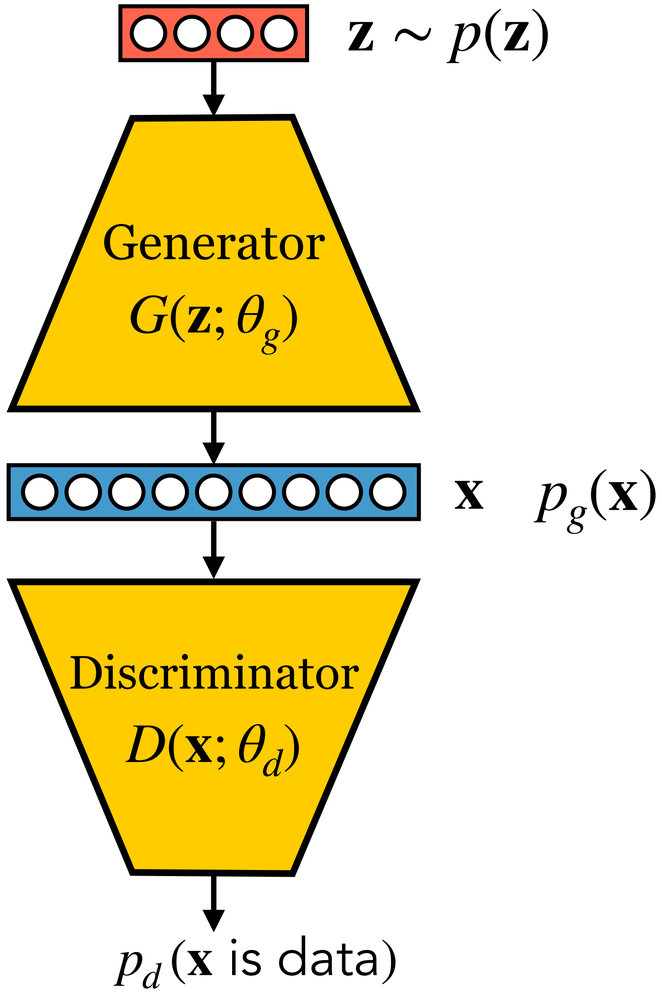
$$\log\left(1 - D(G(\mathbf{z}))\right)$$

We adjust G such that D thinks the *generated sample* comes from *real data*.

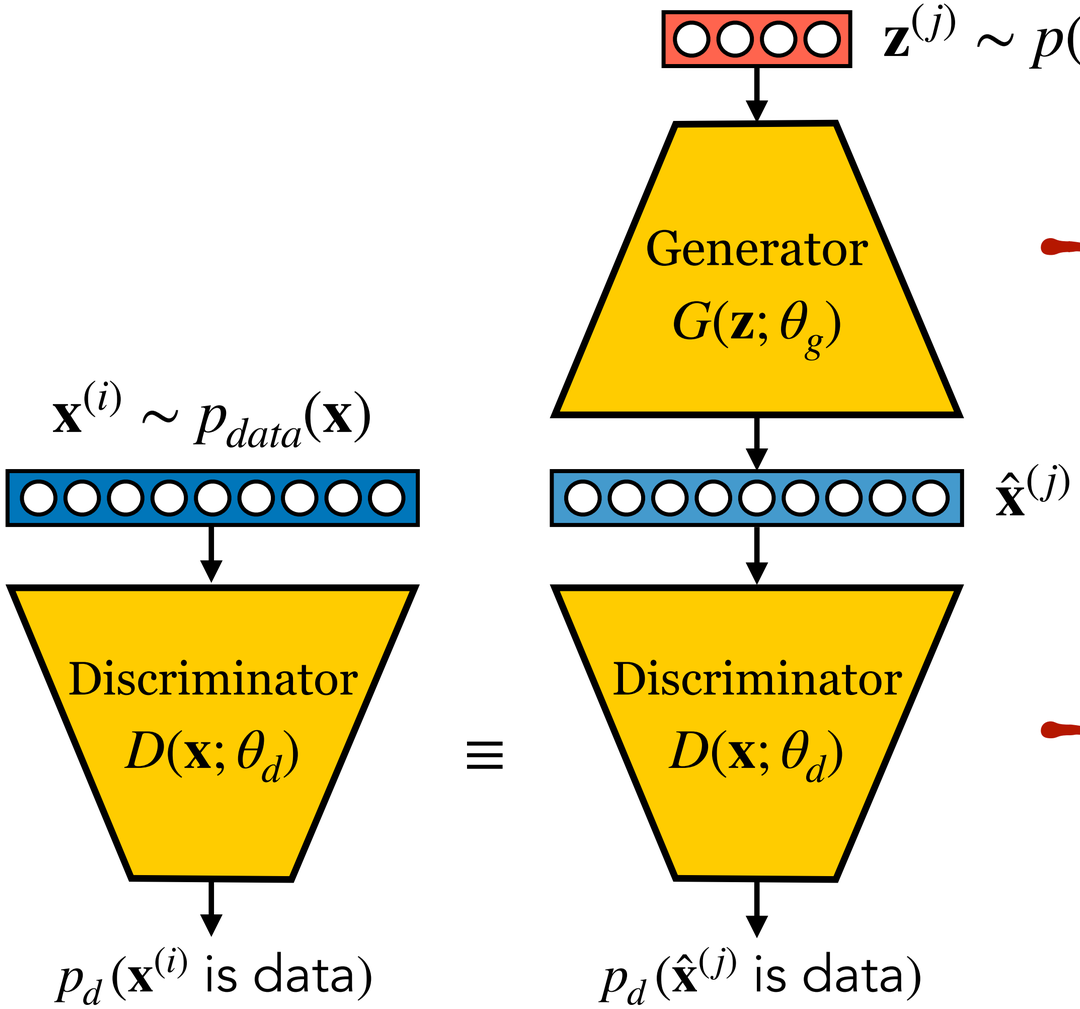
Two-player minimax game: G wants to minimize performance of D, and D wants to maximize its own performance.

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

Global optimum (given enough capacity): $p_g = p_{data}$



Training of GANs



$$\min_G \sum_j \left[\log\left(1 - D(G(\mathbf{z}^{(j)}))\right)\right]$$

$$\max_D \sum_i \left[\log D(\mathbf{x}^{(i)})\right] + \sum_j \left[\log\left(1 - D(\hat{\mathbf{x}}^{(j)})\right)\right]$$

recognize true samples (assign high prob.)

reject generated samples (assign low prob.)

Algorithm

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

for number of training iterations **do**

  for k steps **do**

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

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

  end for

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

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

end for

[Ian Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014.]

Algorithm

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

for number of training iterations **do**

  for k steps **do**

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

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

  end for

- D is optimized k times (typically $k \in \{1,\ldots,5\}$), for each optimization step of G.

end for

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

Algorithm

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

for number of training iterations **do**

for k steps **do**

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

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

end for

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

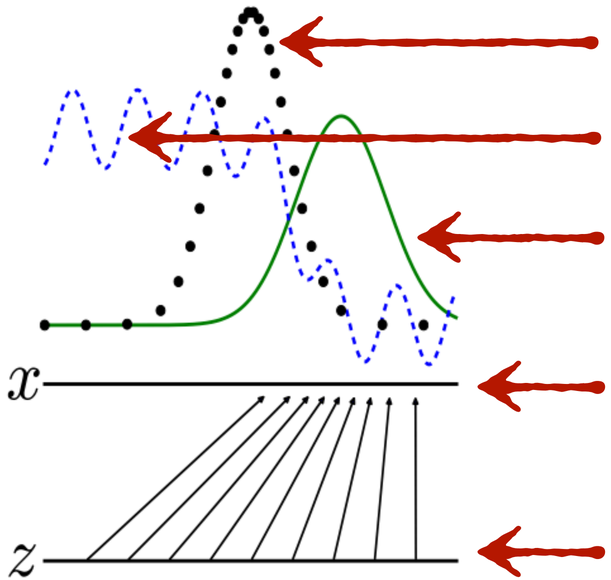
$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

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

Training of GANs: Basic Intuition

Example: Fitting a 1D Gaussian dist.



Samples from data generating distribution $p_{data}(x)$: black, dotted line

Discriminative distribution D : blue, dashed line

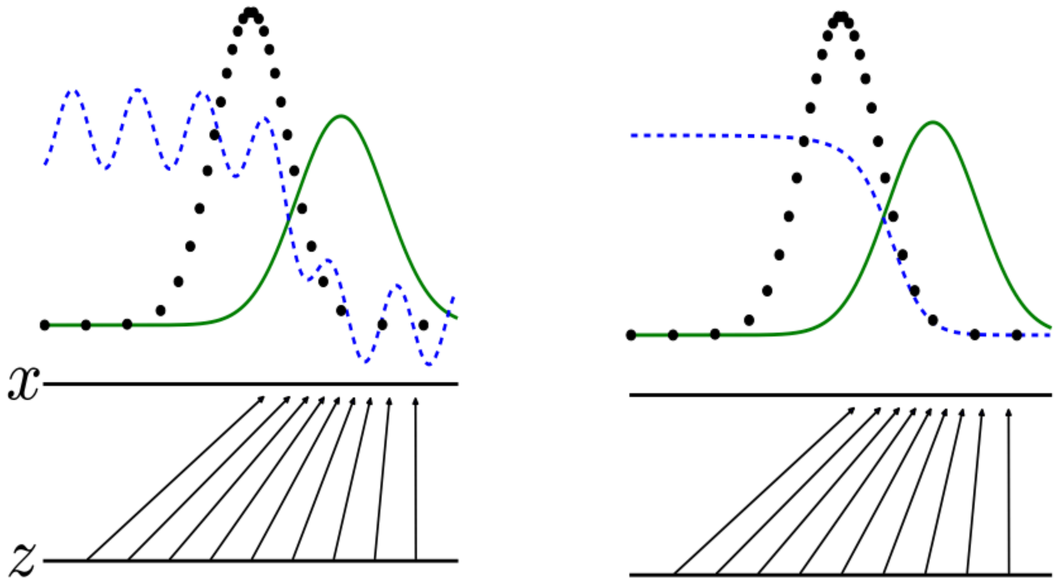
Generative distribution $p_g(x)$, G : green, solid line

Part of the domain of x with transformed samples via the mapping $x = G(z)$.

Lower horizontal line: domain where z is sampled (in this case uniformly).

Training of GANs: Basic Intuition

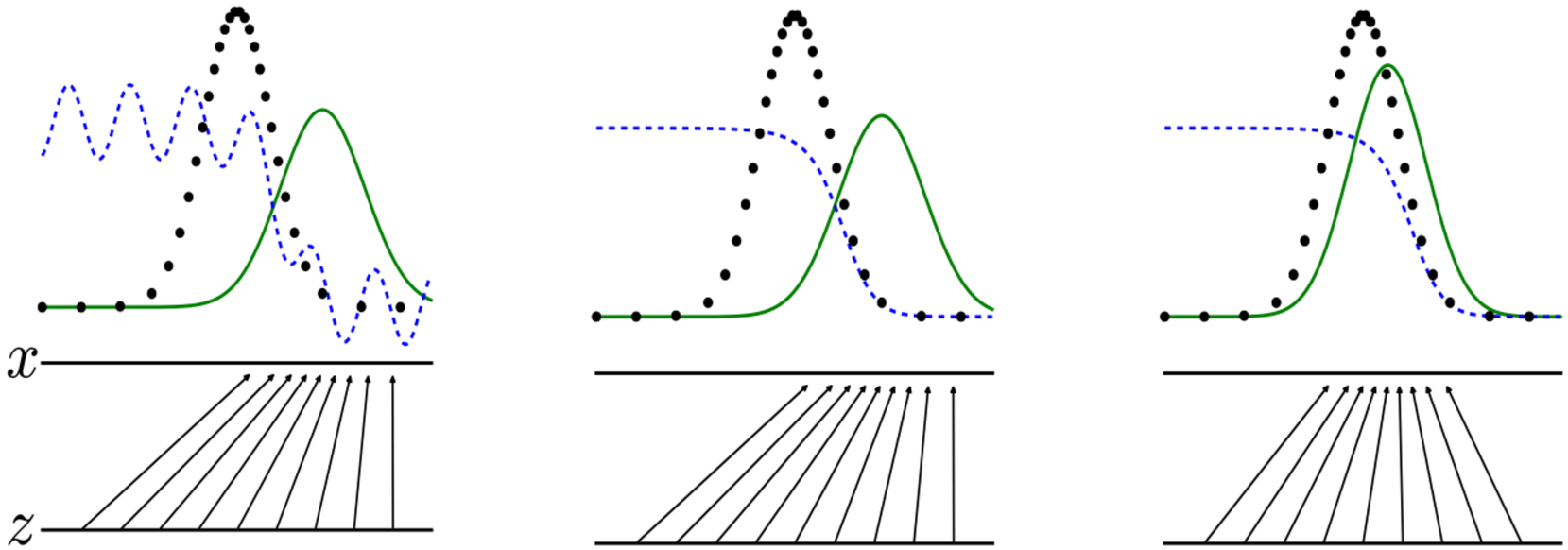
Example: Fitting a 1D Gaussian dist.



- In the inner loop, D is trained to discriminate real data, converging to $D^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)}$.

Training of GANs: Basic Intuition

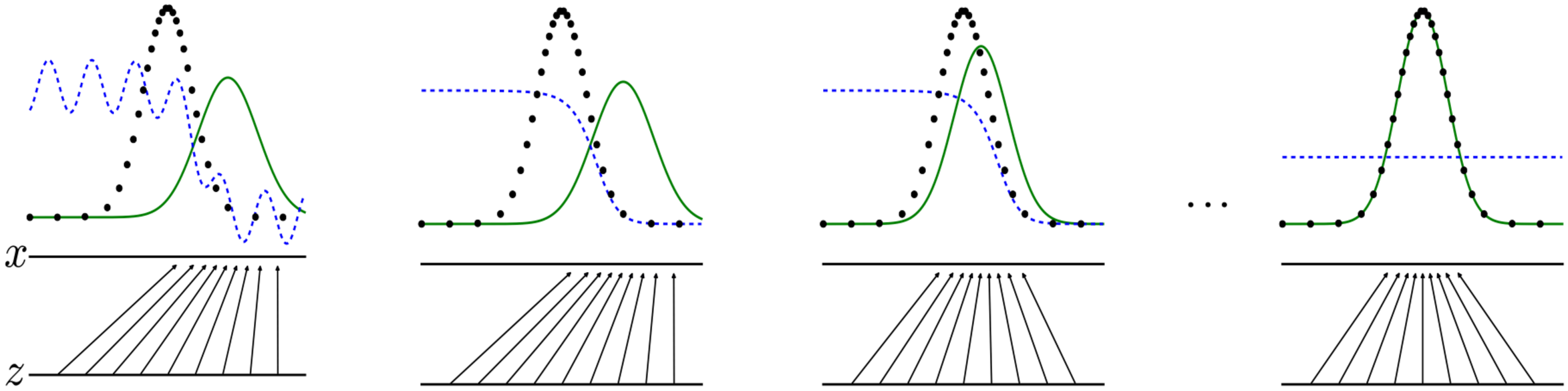
Example: Fitting a 1D Gaussian dist.



- In the inner loop, D is trained to discriminate real data, converging to $D^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)}$.
- After an update to G, gradient of D guides G to regions that are more likely to be classified as real data.

Training of GANs: Basic Intuition

Example: Fitting a 1D Gaussian dist.



- In the inner loop, D is trained to discriminate real data, converging to $D^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)}$.
- After an update to G, gradient of D guides G to regions that are more likely to be classified as real data.
- Given enough capacity, they reach to $p_g = p_{data}$ and $D(x) = \frac{1}{2}$.

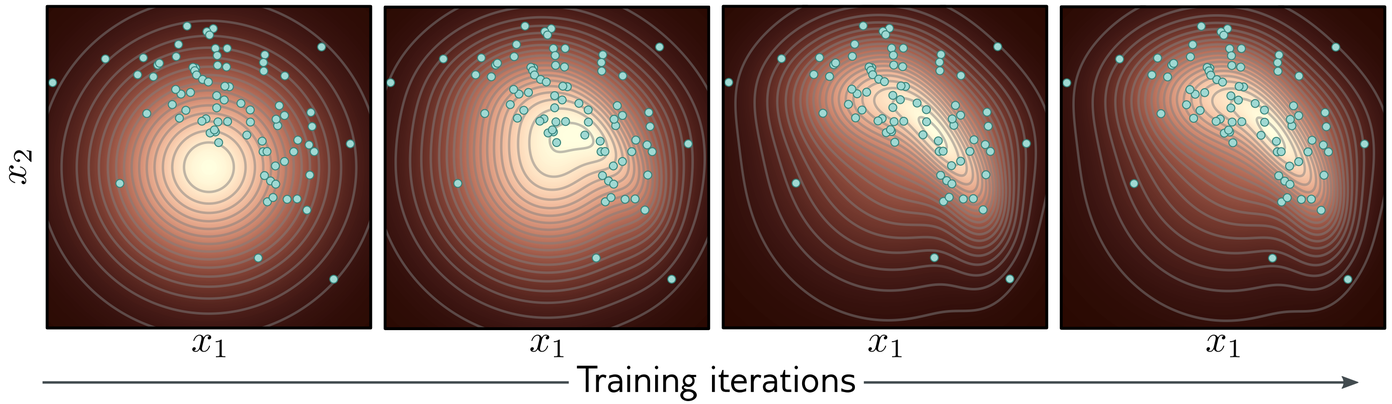
Today

- [ ] Deep Generative Models
  - [x] Variational Autoencoders
  - [ ] Generative Adversarial Networks
    - [x] Architecture & Training
    - [ ] Mode Collapse
    - [ ] Further Extensions & Variants

Comparison to Variational Autoencoders

Variational autoencoder:

- Aims to learn a probability distribution over the training data.
- Likelihood of real examples increase under this distribution (by maximizing VLB), which is then used to draw new samples.



Generative adversarial network:

- Avoids the issue of explicitly designing a tractable density function.
- A mechanism for generating samples. Training encourages samples to become less distinguishable from real examples.

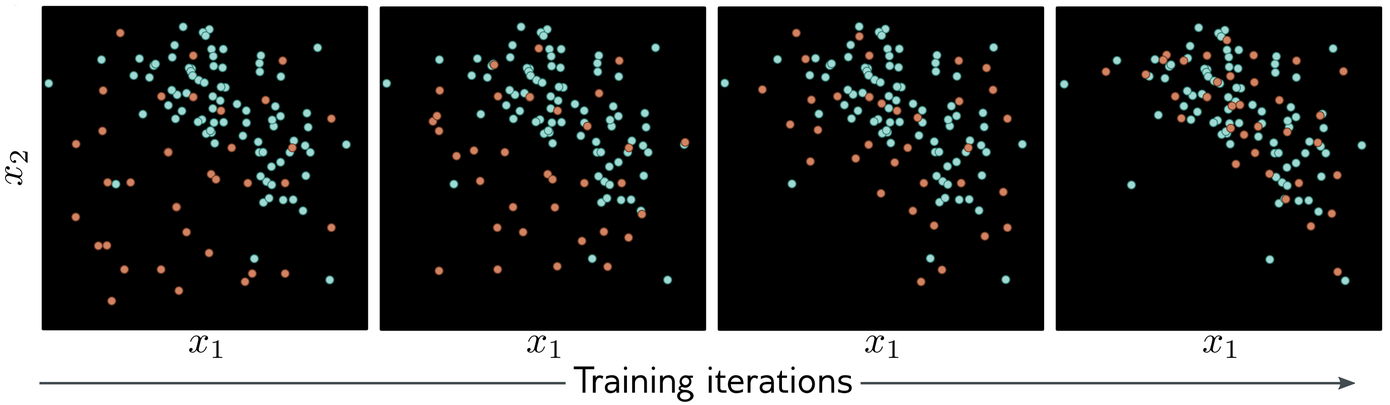
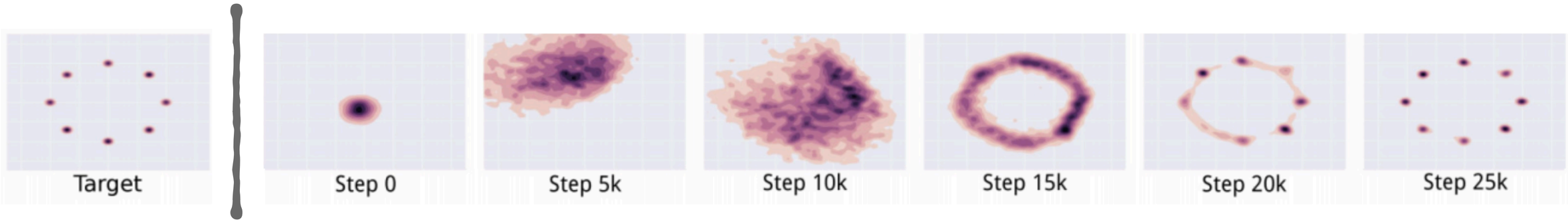


Illustration source: Simon J. D. Prince, "Understanding Deep Learning".

Mode Coverage

An example of "good mode coverage":

Here, the generative model is trained on data from a mixture of Gaussians.

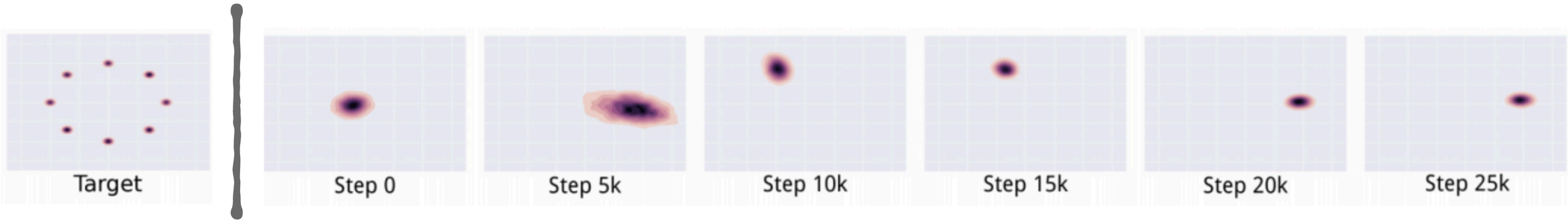


(Simple) GANs often struggle to achieve this.

Mode-Collapse

One main problem of GANs is the so-called “Mode-Collapse”:

This happens when many latent configurations are mapped to single or similar outputs in the generator.

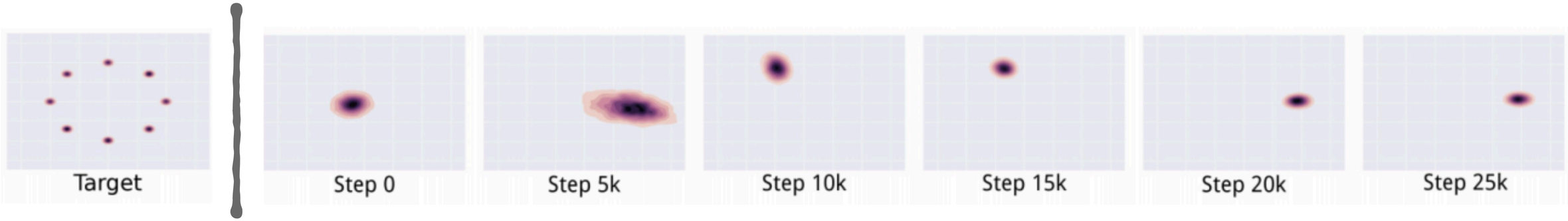


Mode-Collapse

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

One main problem of GANs is the so-called "Mode-Collapse":

This happens when many latent configurations are mapped to single or similar outputs in the generator.

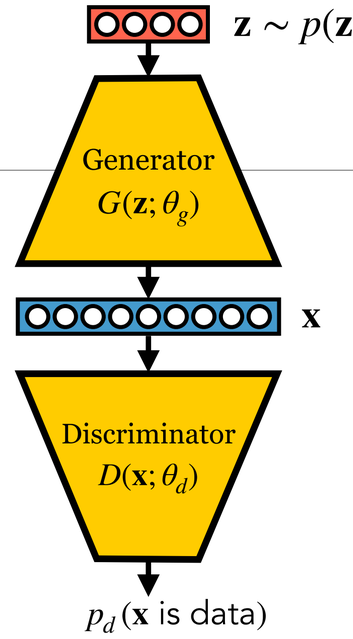


- The GAN switches between different modes during training when being trained on a mixture of Gaussians.
  - When the generator sticks to one out of *n* modes (of equal probability mass), it is *n* times more likely that an $\mathbf{x}$ from this mode comes from the generator (normalization of probabilities). Hence, the discriminator will predict that this $\mathbf{x}$ is fake.
- Result: Generating high quality samples with very limited variability, covering only a part of $p_{data}(\mathbf{x})$.

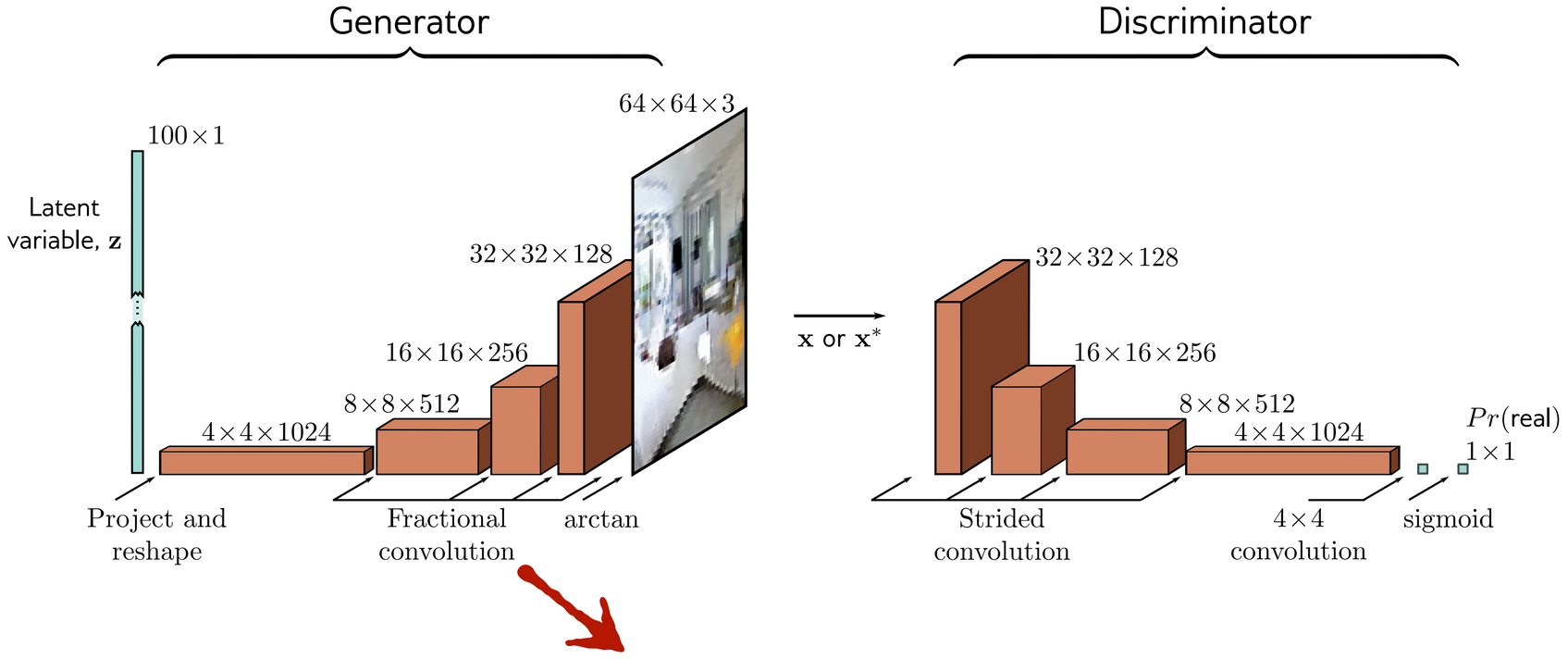
Today

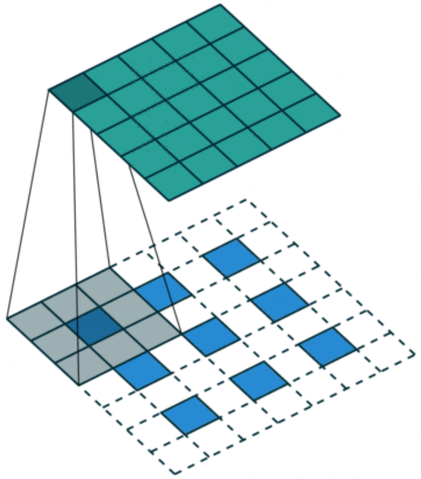
- [ ] Deep Generative Models
  - [x] Variational Autoencoders
  - [ ] Generative Adversarial Networks
    - [x] Architecture & Training
    - [x] Mode Collapse
    - [ ] Further Extensions & Variants

Deep Convolutional GANs (DCGAN)



Uses a convolutional architecture without (de)-pooling and with Batch-Norm.

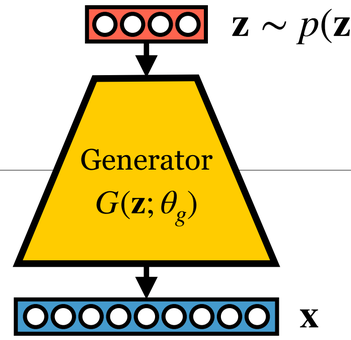




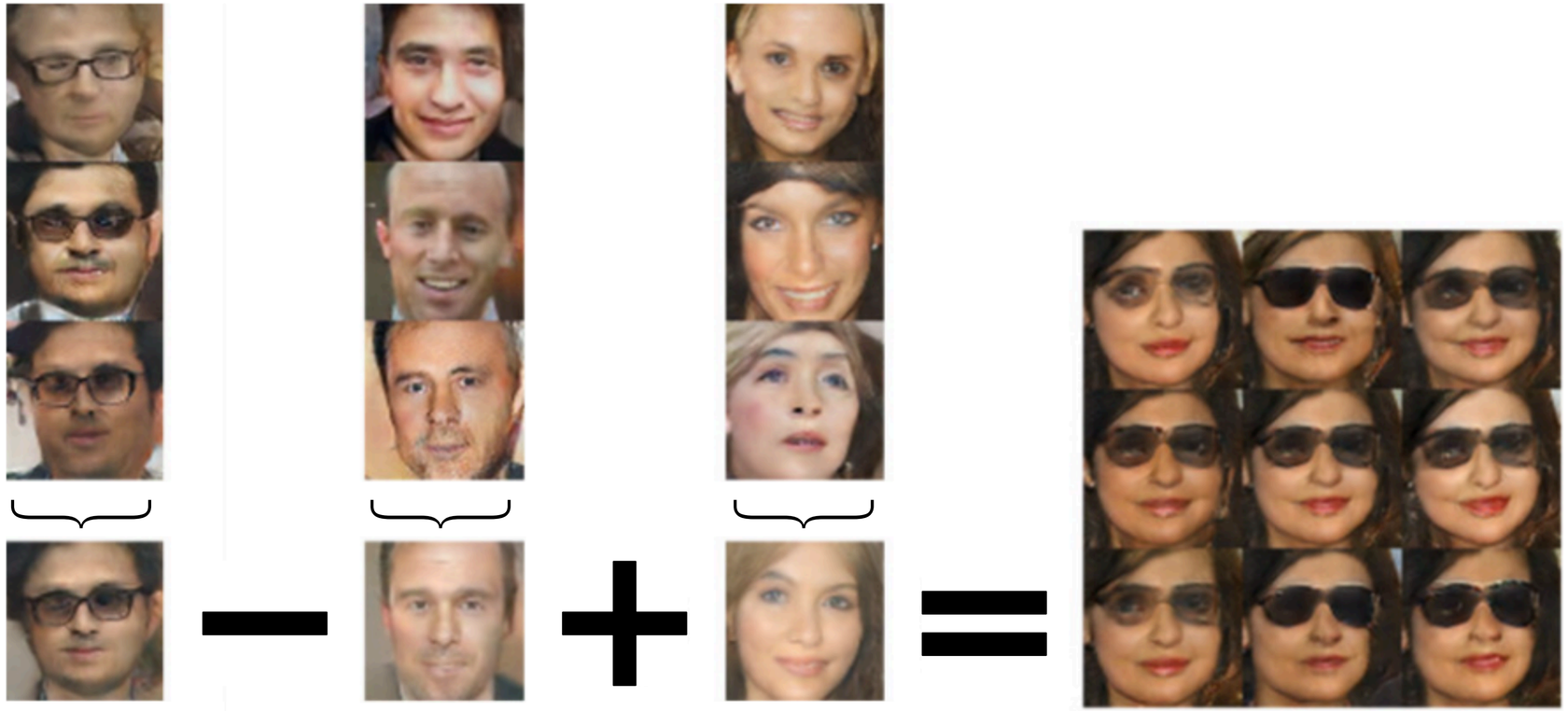
Fractionally-strided convolutions: Transposed convolutions with strides & adding empty pixels between the real ones.

[A. Radford, L. Metz, & S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks." ICLR 2016.]

Vector Arithmetics in Latent Space



It turns out, semantic arithmetics can be performed in latent space of GANs.

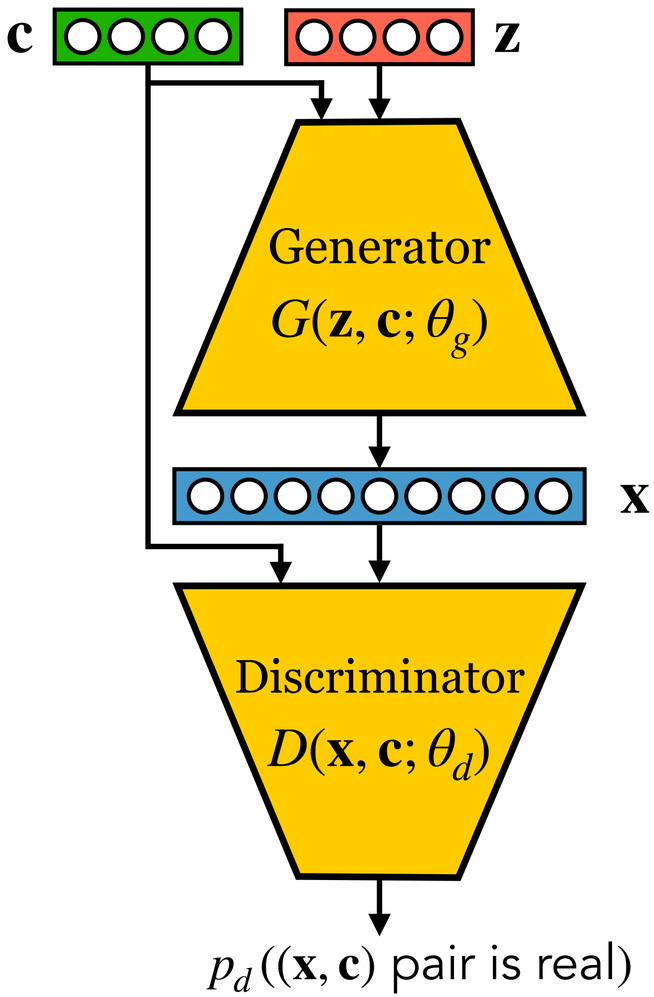


[A. Radford, L. Metz, & S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks." ICLR 2016.]

Conditional GANs

One can use an additional (condition) information as an extra input to the generator.

The discriminator also gets this information.



An example:

- A (standard) GAN trained on CIFAR-10 can generate samples from all categories, but we can not specify the category we want.
- **Conditional GAN:** We can generate cat images when given the condition "cat".

The condition input can be:

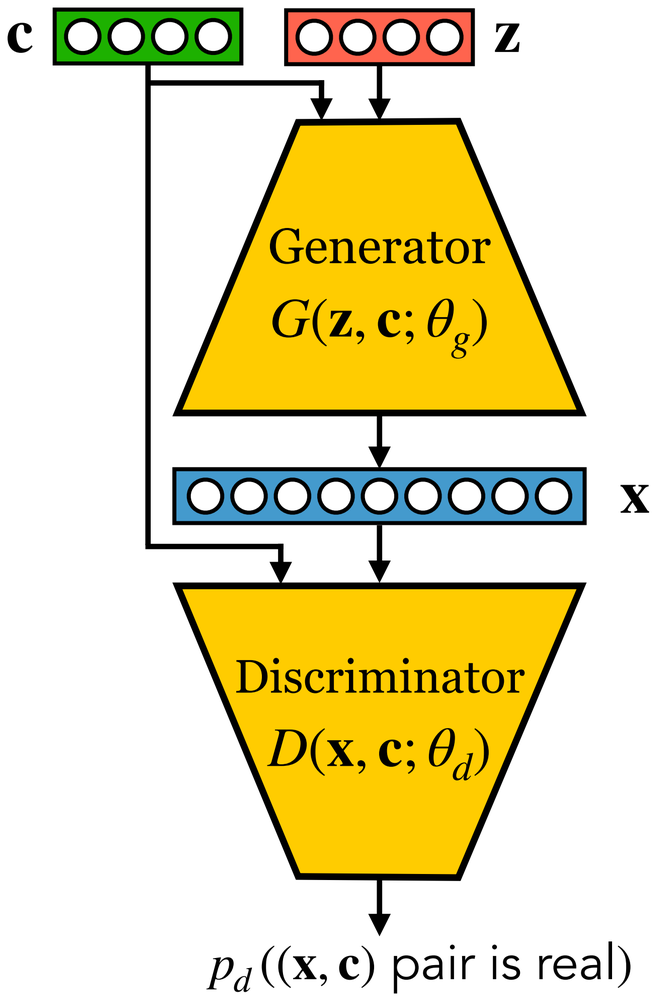
- the class/label of the image to be generated,
- an attribute vector for an image (e.g., male or female face)
- a different image, …

[M. Mirza & S. Osindero (2014). "Conditional generative adversarial nets." *arXiv preprint arXiv:1411.1784.*]

Conditional GANs

One can use an additional (condition) information as an extra input to the generator.

The discriminator also gets this information.



This allows for many different applications of GANs:

- text-to-image translation
- image-to-image translation
- style transfer, photo colorization
- transforming photos from summer to winter, or day to night, …

[M. Mirza & S. Osindero (2014). "Conditional generative adversarial nets." *arXiv preprint arXiv:1411.1784*.]

Wasserstein GANs

Original GANs can have problems with convergence, stability of learning, or mode collapse.

- In the original GAN, the discriminator essentially minimizes the KL-divergence between the posterior distribution and the target distribution.
- Wasserstein GANs use the **Wasserstein Distance**.
  - This simply amounts to a different loss with better properties.
  - Discriminator provides a better and more stable learning signal for the generator.
- Here, one has to assure that the weights don't become too large to avoid instabilities.

[M. Arjovsky, S. Chintala, & L. Bottou, "Wasserstein generative adversarial networks." ICML 2017.]

Wasserstein GANs

$$\min_G \max_D \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}\left[D(\mathbf{x})\right] - \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}\left[D(G(\mathbf{z}))\right]$$

Original GANs can have problems with convergence, stability of learning, or mode collapse.

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: : α, the learning rate. c, the clipping parameter. m, the batch size. n_{critic}, the number of iterations of the critic per generator iteration.

Require: : w_0, initial critic parameters. θ_0, initial generator's parameters.

while θ has not converged **do**
  for $t = 0, ..., n_{\text{critic}}$ **do**
    Sample $\{x^{(i)}\}_{i=1}^{m} \sim \mathbb{P}_r$ a batch from the real data.
    Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
    $g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^{m} f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))\right]$
    $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$
    $w \leftarrow \text{clip}(w, -c, c)$
  end for
  Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
  $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))$
  $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$
end while

[M. Arjovsky, S. Chintala, & L. Bottou, "Wasserstein generative adversarial networks." ICML 2017.]

Example Applications: Image Generation

GANs can produce photorealistic images.

[Goodfellow et al., "Generative Adversarial Nets", NIPS 2014.]

Progress on generating "faces":

2014 2015 2016 2017

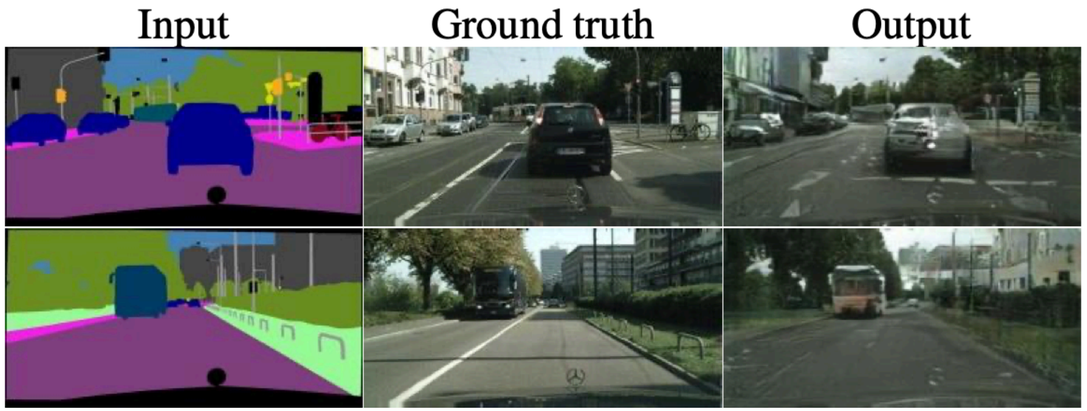
An example generated by StyleGAN based on portraits:

StyleGAN [Karras et al., CVPR 2019]

StyleGAN2 - http://thispersondoesnotexist.com/

Example Applications: Image-to-Image Translation

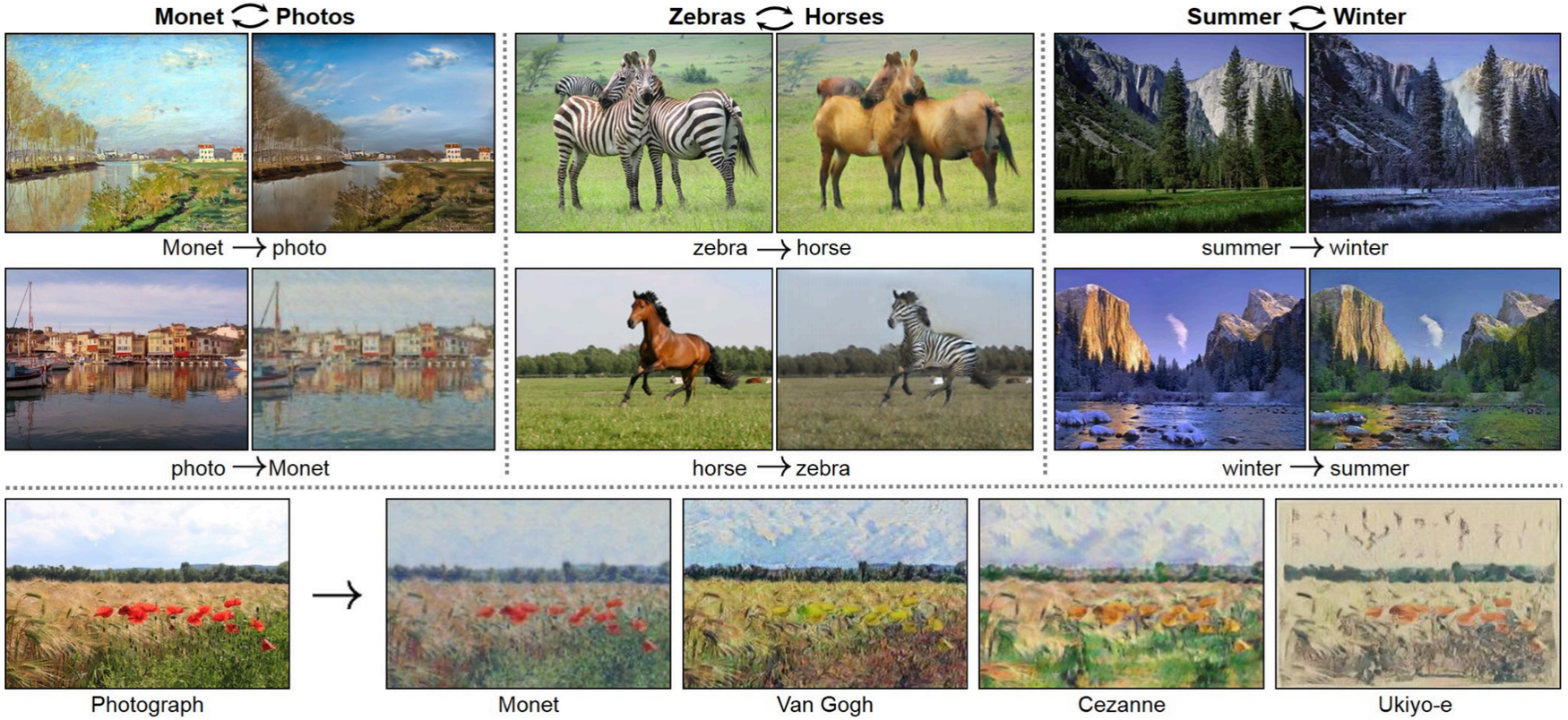
Pix2Pix: Condition on one "style" of image, generate other style.

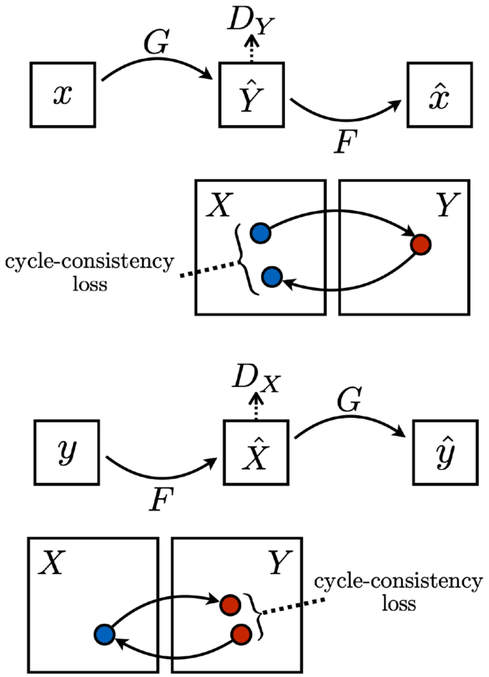


[P. Isola, J. Y. Zhu, T. Zhou & A. A. Efros, "Image-to-image translation with conditional adversarial networks." CVPR 2017.]

Example Applications: Image-to-Image Translation

CycleGAN: Learns a forward and backward mapping between two domains, with an additional cycle-consistency loss to regularize this mapping.





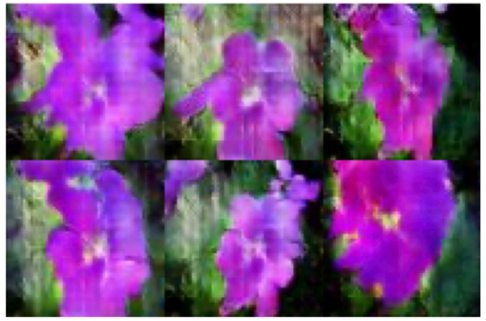
[J. Y. Zhu, T. Park, P. Isola and A. A. Efros: "Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks." ICCV, 2017.]

Example Applications: Text-to-Image Synthesis

this small bird has a pink breast and crown, and black primaries and secondaries.

this magnificent fellow is almost all black with a red crest, and white cheek patch.

the flower has petals that are bright pinkish purple with white stigma



this white and yellow flower have thin white petals and a round yellow stamen

Text descriptions (content)

Images (style)

The bird has a **yellow breast** with **grey** features and a small beak.

This is a large **white** bird with **black wings** and a **red head**.

A small bird with a **black head and wings** and features grey wings.

This bird has a **white breast**, brown and white coloring on its head and wings, and a thin pointy beak.

A small bird with **white base** and **black stripes** throughout its belly, head, and feathers.

A small sized bird that has a cream belly and a short pointed bill.

This bird is **completely red**.

[S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, & H. Lee, "Generative adversarial text to image synthesis." ICML 2016.]

Summary

- GANs are powerful generative models.
- They can be trained purely by gradient descent, no Monte Carlo sampling necessary.
  - Generator aims to fool the discriminator, and discriminator wants to maximize its own performance.
  - Relatively harder to train. Needs a lot of tuning to balance the generator and discriminator.
- Several improvements over the original model available.
  - Conditional GANs, DCGANs, Wasserstein GANs, …
  - Various applications: image-to-image translation, text-to-image synthesis, …
- Over the last decade, GANs became very popular with state-of-the-art results in image generation.
  - Today, diffusion models have become the dominating model in image generation.

[J. Ho et al. “Denoising Diffusion Probabilistic Models.” NeurIPS 2020.]
[P. Dhariwal & A. Nichol. “Diffusion Models Beat GANs on Image Synthesis.” NeurIPS 2021.]

Today

- [ ] Deep Generative Models
  - [x] Variational Autoencoders
  - [x] Generative Adversarial Networks
    - [x] Architecture & Training
    - [x] Mode Collapse
    - [x] Further Extensions & Variants

Questions?