Generative Adversarial Networks

Advanced Topics in Machine Learning

Andrea Cini, Cesare Alippi

22 September, 2021

Università della Svizzera italiana
slides credits: Daniele Grattarola

Which photo is real?

Which photo is real?

thispersondoesnotexist.com

Generative Adversarial Networks [1] in short

Goal: generate samples that **look like** the real data.

[1] I. J. Goodfellow *et al.*, *Generative Adversarial Networks*, 2014.

Generative Adversarial Networks [1] in short

Goal: generate samples that **look like** the real data.

Two main **neural networks**:

- **Generator** $G(z)$ maps random noise z to the data space;
- **Discriminator** $D(x)$ decides whether sample x was generated by G or is a real sample;

[1] I. J. Goodfellow *et al.*, *Generative Adversarial Networks*, 2014.

Generative Adversarial Networks [1] in short

Goal: generate samples that **look like** the real data.

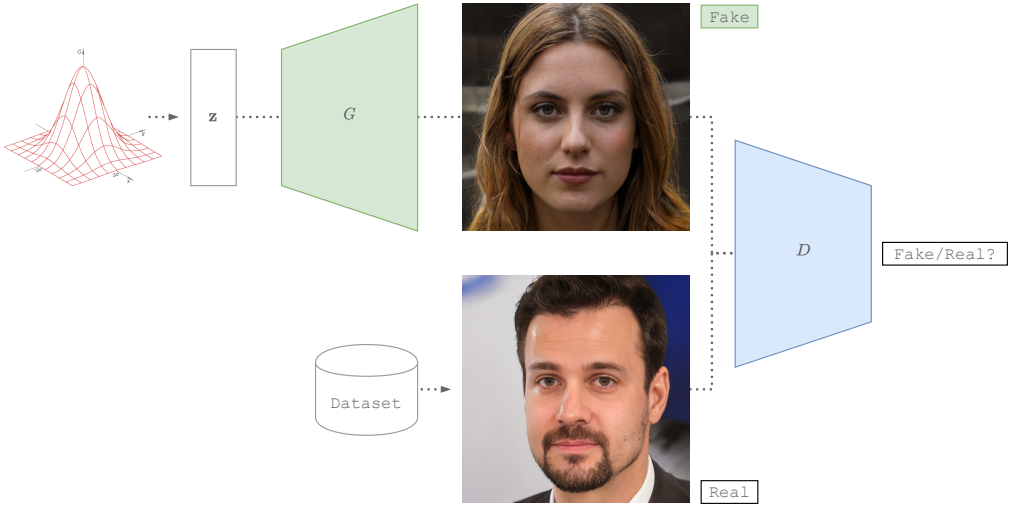
Two main **neural networks**:

- **Generator** $G(z)$ maps random noise z to the data space;
- **Discriminator** $D(x)$ decides whether sample x was generated by G or is a real sample;

The two components play $\underbrace{\textbf{against each other}}_{\text{Adversarial}}$ until the **generator** fools the **discriminator**.

[1] I. J. Goodfellow *et al.*, *Generative Adversarial Networks*, 2014.

Generative Adversarial Networks in short



Generator

Different types of neural networks can be used as **generator**:

- Convolutional for images or audio;
- Recurrent for text or sequences;
- MLPs for tabular data;

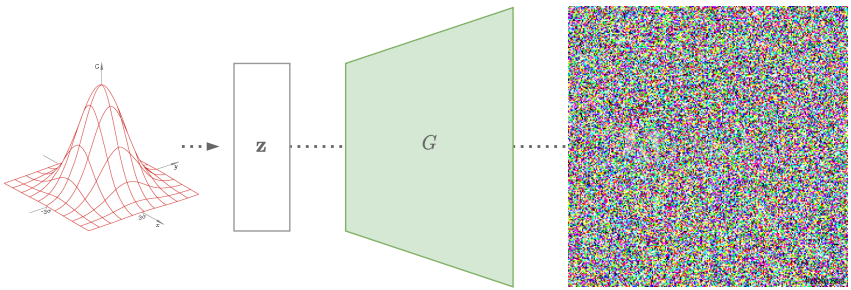
Generator

Different types of neural networks can be used as **generator**:

- **Convolutional for images or audio**; ← we focus mostly on these (easier to visualize)
- Recurrent for text or sequences;
- MLPs for tabular data;

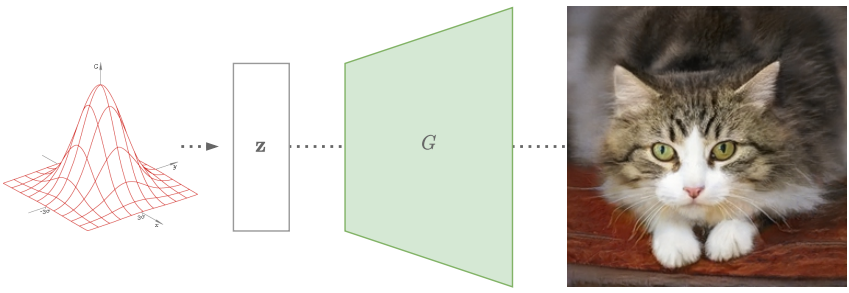
Generator

Generates an image of w by h pixels $\mathrm{x} = G(\mathrm{z}) \in [0,1]^{w \times h}$, where $\mathrm{z} \sim p_z(\mathrm{z})$ (*e.g.*, $\mathcal{N}(0, I)$).



Generator

Generates an image of w by h pixels $\mathsf{x} = G(\mathsf{z}) \in [0,1]^{w \times h}$, where $\mathsf{z} \sim p_z(\mathsf{z})$ (*e.g.*, $\mathcal{N}(0, I)$).



(thiscatdoesnotexist.com)

Discriminator

The **discriminator** looks at samples from the data space $\mathrm{x} \in [0,1]^{w \times h}$ and outputs:

- 1 if the samples come from the real data distribution, *i.e.*, $\mathrm{x} \sim p^*(\mathrm{x})$;
- 0 if the samples are **fake**, *i.e.*, $\mathrm{x} \sim p_g(\mathrm{x})$ (first $\mathrm{z} \sim p_z(\mathrm{z})$, then $\mathrm{x} = G(\mathrm{z})$);

Discriminator

The **discriminator** is trained to optimise two objectives:

- $\max_D \mathbb{E}_{\mathrm{x} \sim p^*(\mathrm{x})} [\log D(\mathrm{x})]$ (output 1 on real samples)
- $\max_D \mathbb{E}_{\mathbf{z} \sim p_z(\mathrm{z})} [\log (1 - D(G(\mathrm{z})))]$ (output 0 on **generated** samples)

(this is just a different way to write a binary classification problem)

Training the generator

Recall: the **generator** has to fool the **discriminator**.

$$\underbrace{\max_{D} \mathbb{E}_{\mathbf{z} \sim p_z(\mathbf{z})} \left[\log\left(1 - D(G(\mathbf{z}))\right)\right]}_{\textbf{Discriminator objective}}$$

Training the generator

Recall: the **generator** has to fool the **discriminator**.

$$\underbrace{\max_{D} \mathop{\mathbb{E}}_{z \sim p_z(z)} [\log (1 - D(G(z)))]}_{\textbf{Discriminator objective}} \rightarrow \underbrace{\min_{G} \mathop{\mathbb{E}}_{z \sim p_z(z)} [\log (1 - D(G(z)))]}_{\textbf{Generator objective}}$$

Fooling = making the **discriminator** output 1 on generated samples.

Putting everything together

If we combine the two objectives for $G(z)$ and $D(x)$ we get **the GAN min-max game**:

$$\min_G \max_D \mathop{\mathbb{E}}_{x \sim p^*(x)} [\log D(x)] + \mathop{\mathbb{E}}_{z \sim p_z(z)} [\log (1 - D(G(z)))]$$

Training the GAN

The GAN is trained by an iterative procedure, **repeated to convergence**:

1. Train the **discriminator** on a batch of real and fake samples;
2. Train the **generator** to fool the discriminator.

Training the GAN - Discriminator training

For k steps do:

1. Sample a minibatch of noise samples $\{z^{(1)}, \ldots, z^{(m)}\}$ from $p_z(z)$;
2. Sample a minibatch of real samples from the dataset $\{x^{(1)}, \ldots, x^{(m)}\}$;
3. Update the weights θ_D of D by gradient **ascent**:[1]

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

[1] In practice we can also do gradient descent by changing the sign.

Training the GAN - Generator training

Do once:

1. Sample a minibatch of noise samples $\{z^{(1)}, \ldots, z^{(m)}\}$ from $p_z(z)$;
2. Update the weights θ_G of G by gradient **descent**:

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

Convergence

The min-max game have a single global minimum, achieved *iff* G perfectly fools D.

Convergence

The min-max game have a single global minimum, achieved *iff* G perfectly fools D.

When this happens:

- The data distribution is **indistinguishable** from the distribution of the generated data: $p^* = p_g$.
- The global minimum of the training criterion is:

$$\mathop{\mathbb{E}}_{\mathrm{x}\sim p^*(\mathrm{x})}\Big[\underbrace{\log D(\mathrm{x})}_{\log(0.5)}\Big] + \mathop{\mathbb{E}}_{\mathrm{z}\sim p_z(\mathrm{z})}\Big[\underbrace{\log\left(1 - D(G(\mathrm{z}))\right)}_{\log(0.5)}\Big] = -\log(4)$$

.

Convergence

The min-max game have a single global minimum, achieved *iff* G perfectly fools D.

When this happens:

- The data distribution is **indistinguishable** from the distribution of the generated data: $p^* = p_g$.
- The global minimum of the training criterion is:

$$\underset{\mathrm{x}\sim p^*(\mathrm{x})}{\mathbb{E}}\Big[\underbrace{\log D(\mathrm{x})}_{\log(0.5)}\Big] + \underset{\mathrm{z}\sim p_z(\mathrm{z})}{\mathbb{E}}\Big[\underbrace{\log\left(1 - D(G(\mathrm{z}))\right)}_{\log(0.5)}\Big] = -\log(4)$$

i.e., the **discriminator** is **maximally confused** (can only output 0.5).

Convergence

The min-max game have a single global minimum, achieved *iff* G perfectly fools D.

When this happens:

- The data distribution is **indistinguishable** from the distribution of the generated data: $p^* = p_g$.
- The global minimum of the training criterion is:

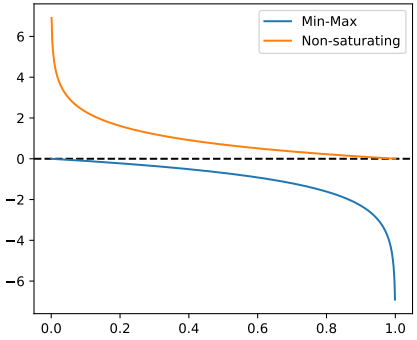
$$\mathop{\mathbb{E}}_{x \sim p^*(x)} \Big[\underbrace{\log D(x)}_{\log(0.5)} \Big] + \mathop{\mathbb{E}}_{z \sim p_z(z)} \Big[\underbrace{\log\left(1 - D(G(z))\right)}_{\log(0.5)} \Big] = -\log(4)$$

  i.e., the **discriminator** is **maximally confused** (can only output 0.5).

Note: convergence is guaranteed only in function space and in the limit of data, in practice we are limited by the amount of data available and the parametrization of G and D.

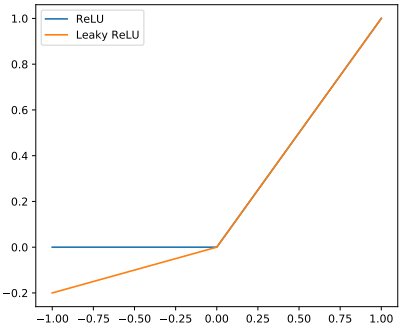
Tips and Tricks

- Use **non-saturating** loss to optimize G:
  $\nabla_{\theta_G} \frac{1}{m} \sum_{i=1}^{m} -\log D(G(\mathrm{z}^{(i)}))$;



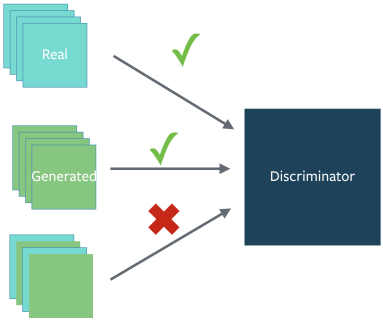
Tips and Tricks

- Use **non-saturating** loss to optimize G: $\nabla_{\theta_G} \frac{1}{m} \sum_{i=1}^{m} -\log D(G(\mathbf{z}^{(i)}))$;
- Avoid **sparse gradients**: use LeakyReLU, average pooling.



Tips and Tricks

- Use **non-saturating** loss to optimize G: $\nabla_{\theta_G} \frac{1}{m} \sum_{i=1}^{m} -\log D(G(\mathrm{z}^{(i)}))$;
- Avoid **sparse gradients**: use LeakyReLU, average pooling.
- **Do not mix** real and fake samples in the same mini-batch to train D;



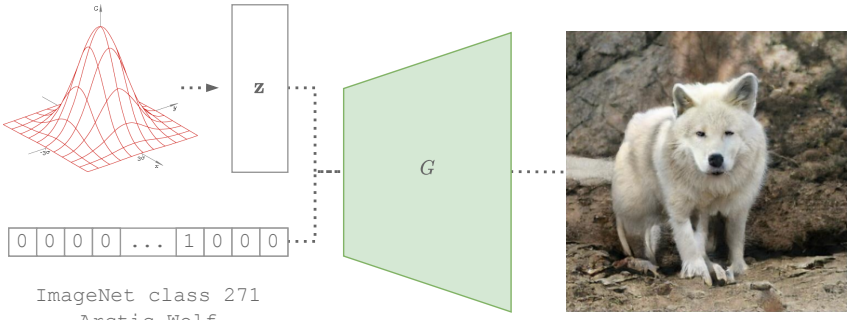
Tips and Tricks

- Use **non-saturating** loss to optimize G: $\nabla_{\theta_G} \frac{1}{m} \sum_{i=1}^{m} -\log D(G(\mathbf{z}^{(i)}))$;
- Avoid **sparse gradients**: use LeakyReLU, average pooling.
- **Do not mix** real and fake samples in the same mini-batch to train D;
- Check out **github.com/soumith/ganhacks** for more ~~dark magic~~ empirical tips.

Conditional GANs

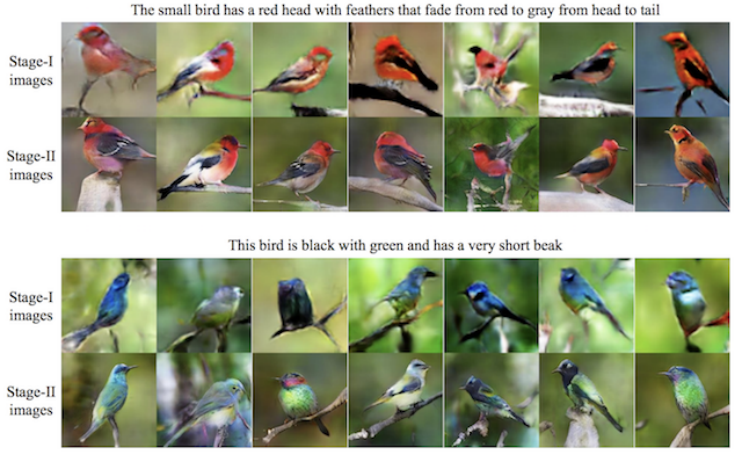
In many cases, the samples from p^* are divided in classes (*e.g.*, ImageNet).

Instead of generating **any** image from p^*, we generate $\mathrm{x} = G(\mathrm{z}, y)$, where y is a **class label**.



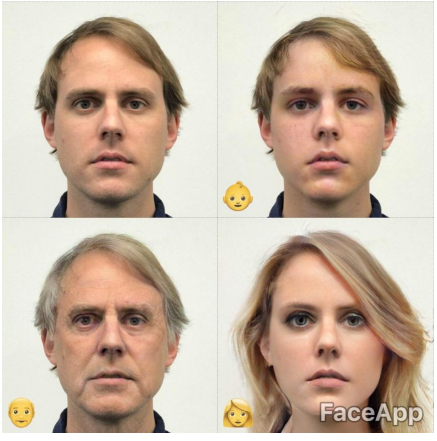
Applications

Text to image translation [2]



[2] S. Reed *et al.*, "Generative adversarial text to image synthesis," 2016.

Style transfer [3]



[3] G. Antipov *et al.*, "Face aging with conditional generative adversarial networks," 2017.

Super-resolution [4]

[4] C. Ledig *et al.*, "Photo-realistic single image super-resolution using a generative adversarial network," 2016.

Demo

Demo

cutt.ly/9Eaqevz

Questions?

References i

[1] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, *Generative adversarial networks*, 2014. arXiv: 1406.2661 [stat.ML].

[2] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee, "Generative adversarial text to image synthesis," *arXiv preprint arXiv:1605.05396*, 2016.

[3] G. Antipov, M. Baccouche, and J.-L. Dugelay, "Face aging with conditional generative adversarial networks," in *2017 IEEE international conference on image processing (ICIP)*, IEEE, 2017, pp. 2089–2093.

[4] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, and Z. Wang, "Photo-realistic single image super-resolution using a generative adversarial network," *arXiv preprint arXiv:1609.04802*, 2016.