Generative Adversarial Networks

Advanced Topics in Machine Learning

Andrea Cini, Daniele Zambon, Daniele Grattarola, Prof. Cesare Alippi

USI

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** $G(\mathbf{z})$ maps random noise $\mathbf{z}$ to the data space;
- **Discriminator** $D(\mathbf{x})$ decides whether sample $\mathbf{x}$ 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.

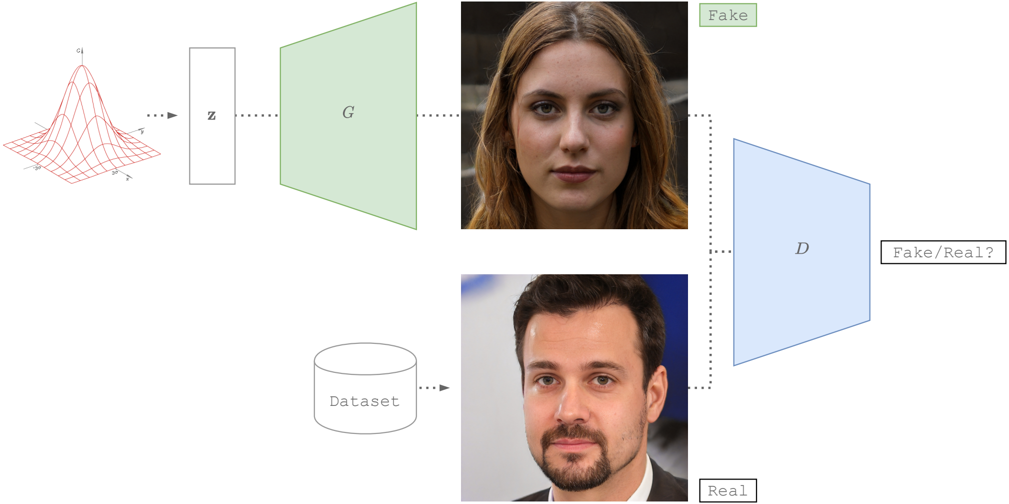
Two main **neural networks**:

- **Generator** $G(\mathbf{z})$ maps random noise $\mathbf{z}$ to the data space;
- **Discriminator** $D(\mathbf{x})$ decides whether sample $\mathbf{x}$ 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;

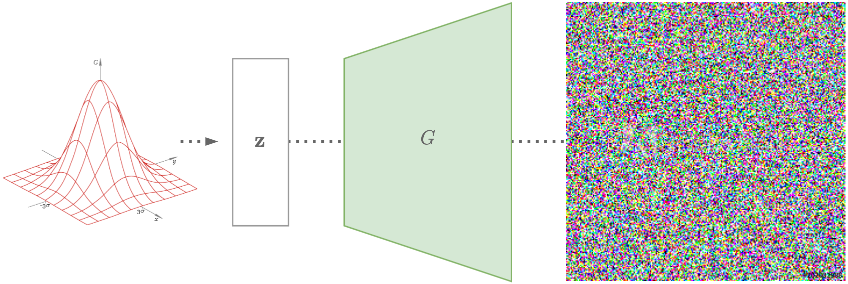
Generator

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

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

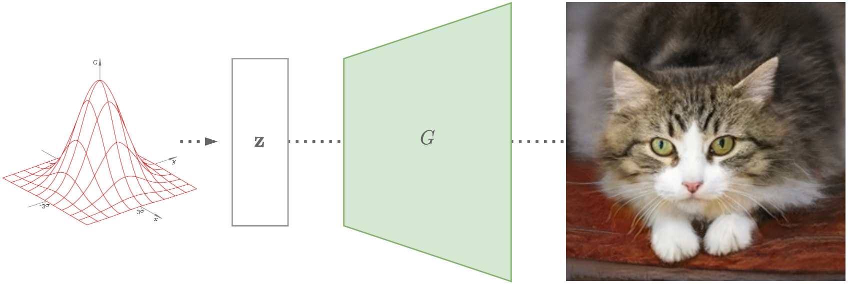
Generator

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



Generator

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



(thiscatdoesnotexist.com)

Discriminator

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

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

Discriminator

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

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

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

Training the generator

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

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

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

Putting everything together

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

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

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

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

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

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

Convergence

The min-max game goes on until G perfectly fools D.

Convergence

The min-max game goes on until 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:

$$\mathop{\mathbb{E}}_{\mathbf{x}\sim p^*(\mathbf{x})}\Big[\underbrace{\log D(\mathbf{x})}_{\log(0.5)}\Big] + \mathop{\mathbb{E}}_{\mathbf{z}\sim p_z(\mathbf{z})}\Big[\underbrace{\log\left(1 - D(G(\mathbf{z}))\right)}_{\log(0.5)}\Big] = \log(0.25)$$

.

Convergence

The min-max game goes on until 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:

$$\mathop{\mathbb{E}}_{\mathbf{x}\sim p^*(\mathbf{x})}\left[\underbrace{\log D(\mathbf{x})}_{\log(0.5)}\right] + \mathop{\mathbb{E}}_{\mathbf{z}\sim p_z(\mathbf{z})}\left[\underbrace{\log\left(1 - D(G(\mathbf{z}))\right)}_{\log(0.5)}\right] = \log(0.25)$$

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

Convergence

The min-max game goes on until 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:

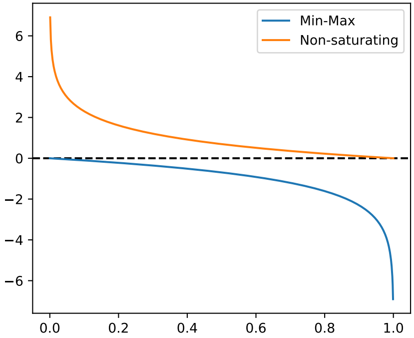
$$\mathop{\mathbb{E}}_{\mathbf{x}\sim p^*(\mathbf{x})}\Big[\underbrace{\log D(\mathbf{x})}_{\log(0.5)}\Big] + \mathop{\mathbb{E}}_{\mathbf{z}\sim p_z(\mathbf{z})}\Big[\underbrace{\log\left(1 - D(G(\mathbf{z}))\right)}_{\log(0.5)}\Big] = \log(0.25)$$

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

Note: convergence is guaranteed given $\underbrace{\text{sufficient capacity}}_{\text{Difficult to know...}}$ of the neural networks.

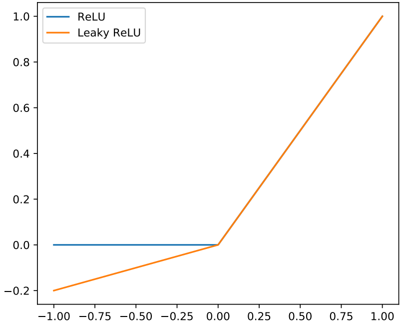
Tips and Tricks

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



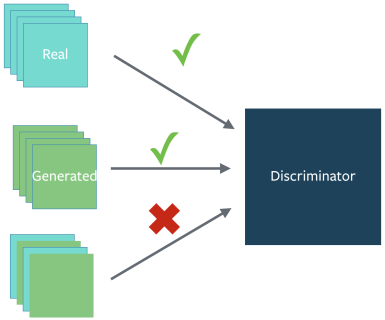
Tips and Tricks

- 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: $\nabla_{\theta_G} \frac{1}{m} \sum_{i=1}^{m} -\log D(G(\mathbf{z}^{(i)}))$;
- Avoid **sparse gradients**: use LeakyReLU, average pooling.
- **Do not mix** real and fake samples to train D;



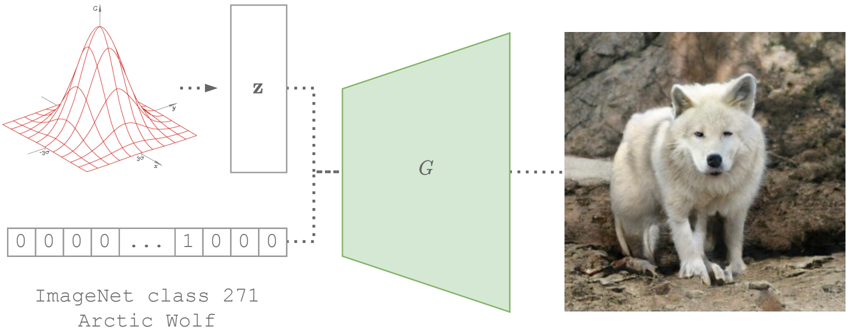
Tips and Tricks

- 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 to train D;
- Check out **github.com/soumith/ganhacks** for more ~~dark magic~~ empirical tips.

Conditional GANs

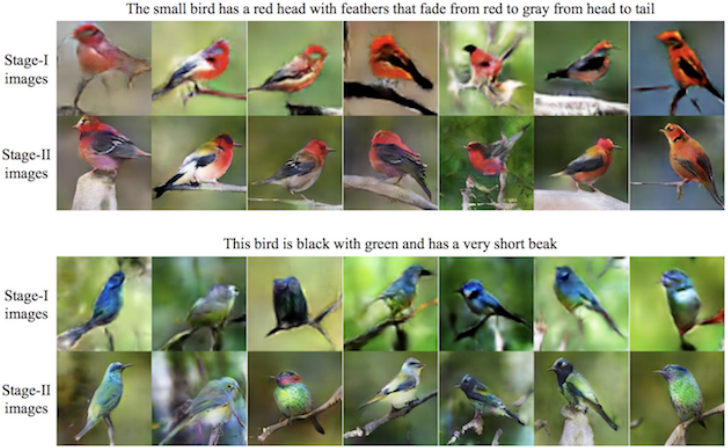
In many cases, the samples from p^* are divided in classes (*e.g.*, ImageNet).

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



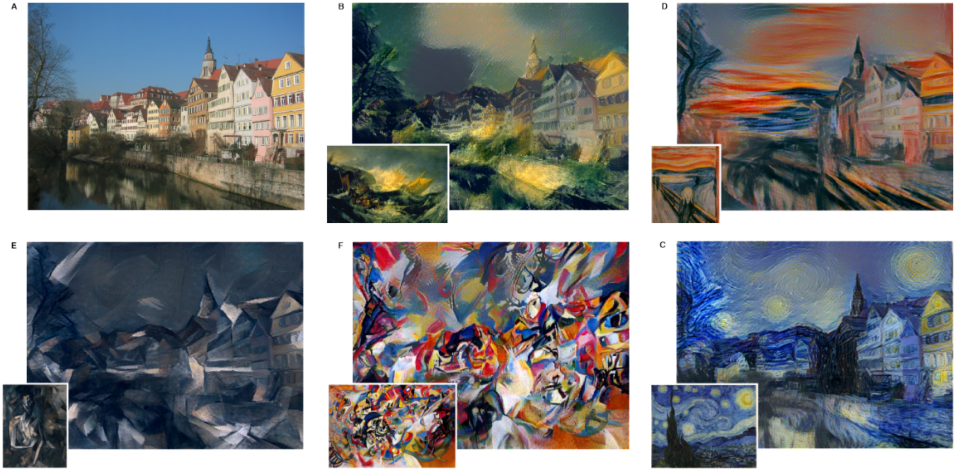
Applications

Text to image translation [2]



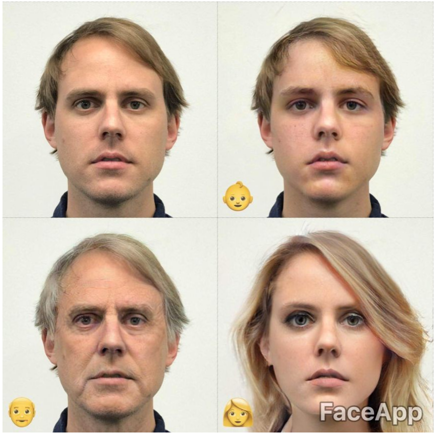
[2] S. Reed *et al.*, "Generative adversarial text to image synthesis," 2016.

Style transfer [3]



[3] T. Karras *et al.*, "A Style-Based Generator Architecture for Generative Adversarial Networks," 2018.

Style transfer [4]



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

Super-resolution [5]

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

Demo

Demo

cutt.ly/sfO5CU9

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] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 4396–4405.

[4] 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.

[5] 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.