Generative Adversarial Nets

based on the original work of Goodfellow et al.

Francesco Caporali

Department of Mathematics, University of Turin

March 4, 2024

The authors proposed a deep generative framework developed on a game theoretic idea.

Objective: overcome the intractability issues of deep learning models that arises from conventional statistical approaches (e.g. maximum likelihood estimation).

Strategy:

	Discriminator D	Generator G
`input`	z: noise x: sample data	z: noise
`goal`	let $D(G(z))$ close to 0, let $D(x)$ close to 1	generate $G(z)$, let $D(G(z))$ close to 1

Let D, G be two maps generated through neural nets:

Discriminator. $D(x) \coloneqq D(x, \theta_D)$, $D : X \to [0, 1]$.
Generator. $G(z) \coloneqq G(z, \theta_G)$, $G : Z \to X$;

Given $x \in X$ (data space), $z \in Z$ (noise space), we define:

- $p_{\texttt{data}}(x)$, density of the input data;
- $p_G(x)$, density of the Generator;
- $p_{\texttt{noise}}(z)$, density of a prior noise.

D and G play a minimax game with value function $V(D, G)$,

$$V(D, G) \coloneqq \mathbb{E}_{x \sim p_{\texttt{data}}}[\log(D(x))] + \mathbb{E}_{z \sim p_{\texttt{noise}}}[\log(1 - D(G(z)))].$$

We aim to find D^*, G^* s.t. $V(D^*, G^*) = \min_G \max_D V(D, G)$.

Discriminator goal

$$\max_D \underbrace{\mathbb{E}_{x\sim p_{\texttt{data}}}[\log(D(x))]}_{D(x)\approx 1} + \underbrace{\mathbb{E}_{z\sim p_{\texttt{noise}}}[\log(1-D(G(z)))]}_{D(G(z))\approx 0}.$$

Generator goal

$$\min_G \max_D \cancel{\mathbb{E}_{x\sim p_{\texttt{data}}}[\log(D(x))]} + \underbrace{\mathbb{E}_{z\sim p_{\texttt{noise}}}[\log(1-D(G(z)))]}_{D(G(z))\approx 1}.$$

Remark

Under the (unrealistic) assumption of being able to find the optimal discriminator $D_G^* := \max_D V(D,G)$, it is possible to show that iteratively alternating computations of D_G^* and updates of G (to minimize $V(D_G^*, G)$), one would obtain $p_G = p_{\texttt{data}}$.

Proposition 1 [Goodfellow et al.]

For a fixed generator G, the associated optimal discriminator D_G^* is

$$D_G^*(x) = \frac{p_{\texttt{data}}(x)}{p_{\texttt{data}}(x) + p_G(x)}, \forall\, x \in X.$$

Proof.

1. Thanks to the Radon-Nikodym Theorem we have

$$\mathbb{E}_{z \sim p_{\texttt{noise}}}[\log(1 - D(G(z)))] = \mathbb{E}_{x \sim p_G}[\log(1 - D(x))].$$

2. Hence we can rewrite $V(D, G)$ as

$$V(D, G) = \int_X \log(D(x))\, p_{\texttt{data}}(x) + \log(1 - D(x))\, p_G(x) dx,$$

with $\operatorname{supp}(p_G) = G(\operatorname{supp}(p_{\texttt{noise}})) \subset X$, $\operatorname{supp}(p_{\texttt{data}}) \subset X$.

3. Consider $f : [0,1] \to \mathbb{R}$, $f(y) := \log(y)\, a + \log(1-y)\, b$, with $a, b > 0$, then f is a concave map and the only maximum is realized by $y = \dfrac{a}{a+b}$.

4. Defining $\tilde{D} := \dfrac{p_{\texttt{data}}}{p_{\texttt{data}} + p_g}$, we have, by monotonicity of the integral, $\forall\, D$,

$$V(D,G) = \int_X \log(D(x))\, p_{\texttt{data}}(x) + \log(1-D(x))\, p_G(x) dx \leq$$

$$\leq \int_X \log\left(\tilde{D}(x)\right) p_{\texttt{data}}(x) + \log\left(1-\tilde{D}(x)\right) p_G(x) dx = V(\tilde{D}, G),$$

$$\implies \tilde{D} = \arg\max_D V(D,G) = D^*_G.$$

□

Theorem 1 [Goodfellow et al.]

The global minimum of the virtual training criterion given by $C(G) \coloneqq \max_D V(D, G)$, is achieved if and only if $p_G = p_{\texttt{data}}$.
For such minimum G, $C(G) = -\log(4)$.

Proof.

1. Let us first rewrite the criterion in a more suitable way:

$$
\begin{aligned}
C(G) &= V(\arg\max_D V(D, G), G) = V(D_G^*, G) = \\
&= \mathbb{E}_{x \sim p_{\texttt{data}}}[\log(D_G^*(x))] + \mathbb{E}_{z \sim p_G}[\log(1 - D_G^*(x))] = \\
&= \mathbb{E}_{x \sim p_{\texttt{data}}}\left[\log\left(\frac{p_{\texttt{data}}(x)}{p_{\texttt{data}}(x) + p_G(x)}\right)\right] + \\
&\quad + \mathbb{E}_{z \sim p_G}\left[\log\left(\frac{p_G(x)}{p_{\texttt{data}}(x) + p_G(x)}\right)\right].
\end{aligned}
$$

2. Let $\tilde{G}$ be a generator such that $p_{\tilde{G}} = p_{\texttt{data}}$, then $C(\tilde{G}) = 2\log(1/2) = -\log(4)$.
3. Morever, adding and subtracting $-\log(4)$, we have that

$$C(G) = -\log(4) + 2\,\mathrm{JSD}\left(p_{\texttt{data}} \,\|\, p_G\right), \forall\, G.$$

4. We recall the properties of the Jensen-Shannon divergence: $\forall\, p, q$ densities $\mathrm{JSD}(p \,\|\, q) \geq 0$ and $\mathrm{JSD}(p \,\|\, q) = 0$ if and only if $p = q$.
5. Those properties are sufficient to conclude that

$$\min_G C(G) = -\log(4) \text{ and}$$
$$C(G) = -\log(4) \iff p_{\texttt{data}} = p_G.$$

□

Proposition 2 [Goodfellow et al.]

If G and D have enough capacity, and at each iteration of the algorithm, D reaches its optimum given G, and p_G is updated so as to improve the criterion

$$\mathbb{E}_{x \sim p_{\texttt{data}}}[\log(D_G^*(x))] + \mathbb{E}_{z \sim p_G}[\log(1 - D_G^*(x))]$$

then p_G converges to $p_{\texttt{data}}$.

Sketch.

1. $f(G) := \max_D V(D, G)$, $f_{D^*_{G_{\text{old}}}}(G) := V(D^*_{G_{\text{old}}}, G)$, with G_{old} which is the resulting generator after its last update.
2. We would like to update G descending the stochastic gradient of f because this would mean that p_G is closer to $p_{\texttt{data}}$ than $p_{G_{\text{old}}}$ was.

3. However this is equivalent to update G descending the stochastic gradient of $f_{D^*_{G_{\text{old}}}}$ (which is what we do).
   This assertion follows from the fact that the subgradients of $f_{D^*_{G_{\text{old}}}}$ are a subset of the subgradients of f: $\partial f_{D^*_{G_{\text{old}}}} \subset \partial f$ (from the convexity of $V(D, G)$).

Remark

Note that the algorithm used to train the GANs does not optimize the underlying distribution of G, p_G, but the parameters of a neural net, θ_G.

This raises theoretical problems, anyway the performances of the model are impressive, therefore GANs are reasonable to be used in practice.

Algorithm. Minibatch SGD for training D and G

function *train*$(D, G,$ *data*$, k,$ *n_epochs*$)$
  for *epoch* **in** $1, \dots,$ *n_epochs* **do**
    for _ **in** $1, \dots, k$ **do**
      sample $\mathbf{z} = (z_i)_{i=1}^m$, $z_i \sim p_{\texttt{noise}}$
      sample $\mathbf{x} = (x_i)_{i=1}^m$, $x_i \sim p_{\texttt{data}}$
      $\theta_G = \theta_G + \lambda \nabla_{\theta_G} \widehat{V}_m(D, G)$
    sample $\mathbf{z} = (z_i)_{i=1}^m$, $z_i \sim p_{\texttt{noise}}$
    $\theta_D = \theta_D - \lambda \nabla_{\theta_D} \widehat{V}_m(D, G)$
  return D, G

Where the sample mean of the loss $V(D, G)$ is computed on the m-dimensional minibatches $\mathbf{z}, \mathbf{x}$:

$$\widehat{V}_m(D, G) := \frac{1}{m} \sum_{i=1}^{m} \log\left(D(x_i)\right) + \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D(G(z_i))\right).$$

Remark

Actual implementation:

- 6^{th} *line.* It is performed one step of the ascent of the stochastic gradient via backpropagation, with loss

$$\widehat{V}_m(D, G) = \frac{1}{m} \sum_{i=1}^{m} \log\left(D(x_i)\right) + \log\left(1 - D(G(z_i))\right);$$

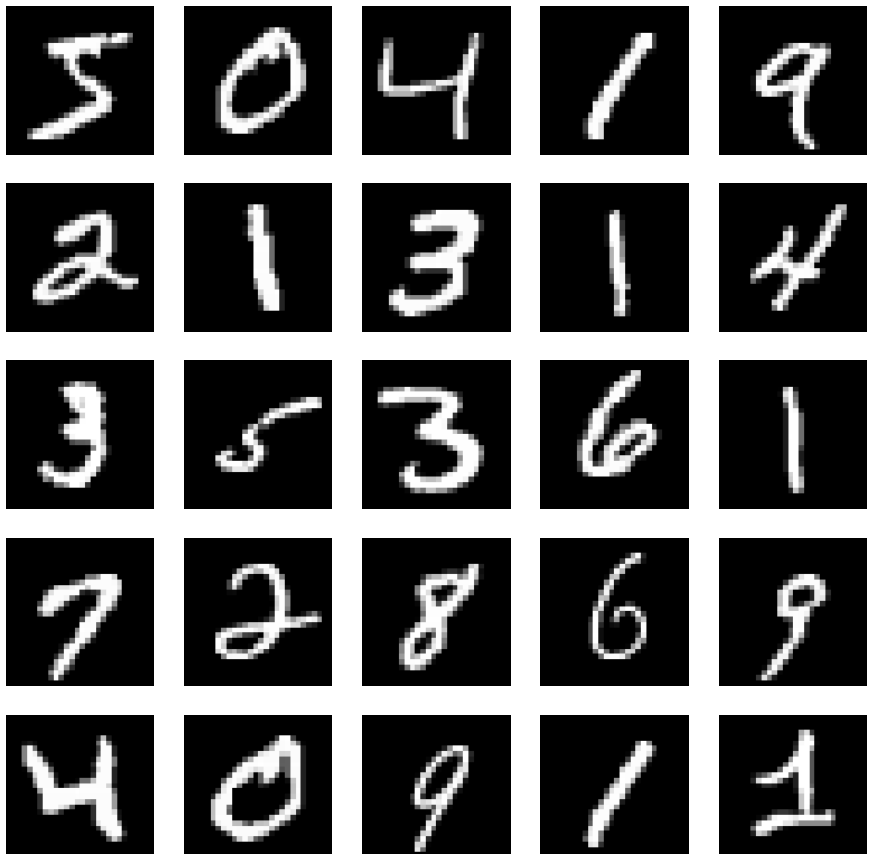
- 8^{th} *line.* Instead of one step of the ascent of $\widehat{V}_m(D, G)$ we, equivalently, do a step of descent with loss

$$\frac{1}{m} \sum_{i=1}^{m} \log\left(D(G(z_i))\right);$$

Note. The classical SGD is often substituted by SGD with momentum or adaptative algorithm (like ADAM).

We performed practical simuations, training and generating data using GANs (based on the work of Vandegar).

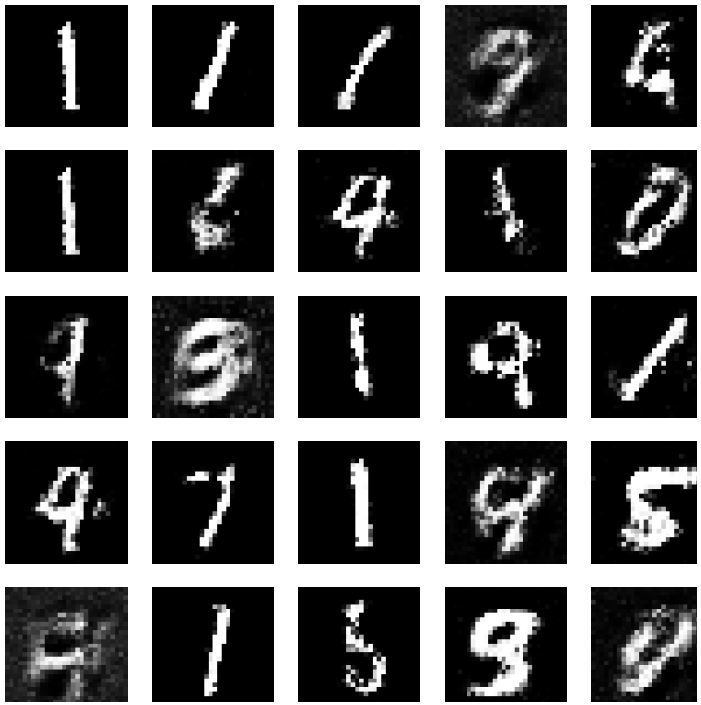
The dataset used as train set is `MNIST`, a database of 60000 images of handwritten digits.



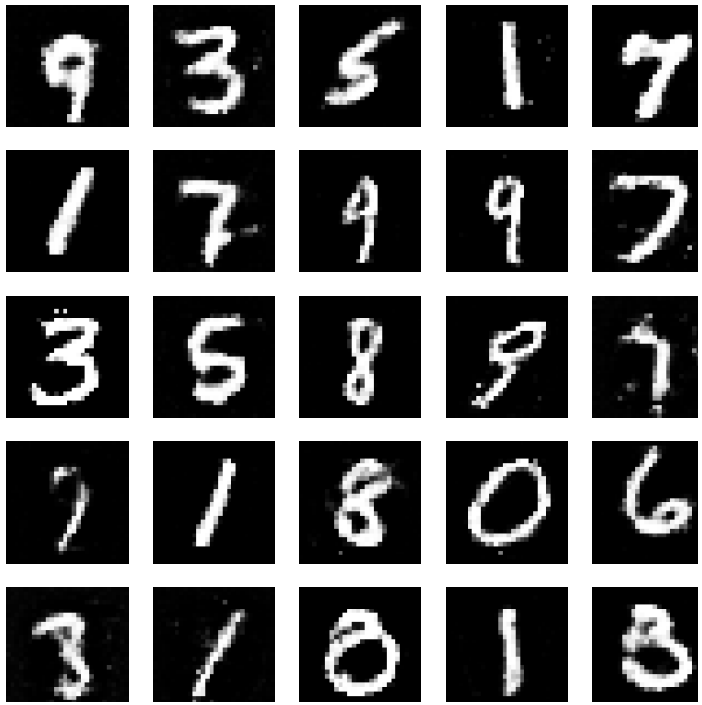
Generated samples obtained after 10^4 and 10^5 epochs of training using nets D, G with architectures α_D, α_G:

$$\alpha_D = \big((28 \times 28, 240, 240, 1), (\texttt{LeakyReLU}, \texttt{LeakyReLU}, \texttt{Sigmoid})\big),$$

$$\alpha_G = \big((100, 1200, 1200, 28 \times 28), (\texttt{ReLU}, \texttt{ReLU}, \texttt{Tanh})\big).$$



(a) 10^4 epochs



(b) 10^5 epochs

References

[1] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2014. doi:10.48550/arXiv.1406.2661.

[2] Maxime Vandegar. Generative adversarial networks. `https://github.com/MaximeVandegar/Papers-in-100-Lines-of-Code/tree/main/Generative_Adversarial_Networks`, 2020. Accessed: 2024/04/03.