Paper Review:

Generative Adversarial Nets (2014)

As we stand on the threshold of a new era, where machines collaborate with human imagination, Generative AI stands tall as a transformative force, breathing life into a world previously untouched by artificial creativity.

Obviously, I didn't write that. I made an AI write it for me. And the point I'm trying to make here is, Generative AI has already gained footing as an integral part of our lives. And while we explore the infinite possibilities it opens up, it might be rewarding to look back at where it all began.

I want this paper review to be as basic as possible. By basic, I mean reduced to the barest of the details. By barest of the details, I mean the simplest of terms. And I want it to be so because the concept of Generative AI is a concept we are being exposed to so extensively today. And I believe that if with some effort, an idea as groundbreaking as Ian Goodfellow's can be understood by absolutely anyone interested enough, then that effort is worth it. Certainly.

For context, we're talking 2014. It would be two years from then until GAN's take off actually. That is to say, this paper was THE groundbreaking introduction to the method of adversarial training. It laid the foundation for all subsequent advancements in the field of generative AI.

Let's begin!

Proposed framework:

> Estimating generative models via an adversarial process, in which they simultaneously train two models: a generative model G that captures the data distribution, and a discriminative model D that estimates the probability that a sample came from the training data rather than G. The training procedure for G is to maximize the probability of D making a mistake.

The paper proposed an idea that was until then, unexplored: train two ML models simultaneously, by making them compete. Hence, 'adversarial' of course.

The proposal introduced a clever approach to training a generative model using two sub-models: the generator and the discriminator. The generator is trained to create new examples, while the discriminator aims to distinguish between real and fake examples. Both models are trained together in an adversarial manner, playing a zero-sum game. The objective is for the generator to produce convincing examples that can deceive the discriminator approximately half the time.

Until 2014, the most striking successes in deep learning had involved discriminative models. Discriminators are algorithms in machine learning that are used to categorize data into distinct groups or classes. Although Convolutional neural networks (a class of neural networks that specializes in processing data that has a grid-like topology) were invented in the 1980s, their breakthrough in the 2000s required fast implementations on GPU's. By 2012, we had ALexNet designed by Alex Krizhevsky.

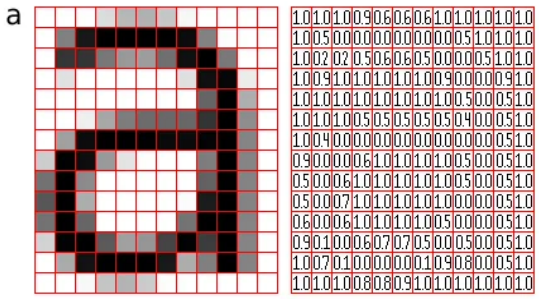


Figure 1: Representation of image as a grid of pixels

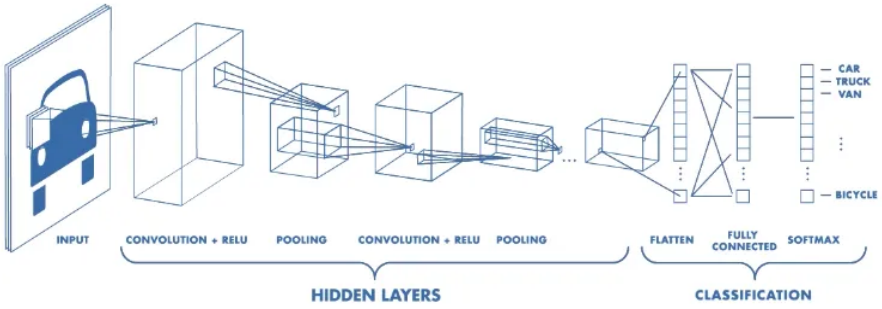


Figure 2: Architecture of a CNN

The core idea of a Generative Adversarial Network is the indirect training approach involving a discriminator neural network. The discriminator assesses the realism of the input, and both the generator and discriminator are dynamically updated during training.

Analogy:

In the context of the GAN framework, we can envision the generative model as a team of counterfeiters aiming to produce counterfeit currency and utilize it without being detected. On the other hand, the discriminative model can be likened to the police, working diligently to identify the counterfeit currency. This setup creates a competitive game where both teams strive to enhance their techniques, ultimately reaching a point where the counterfeit

currency becomes indistinguishable from the genuine articles. Through this iterative process of competition and improvement, the generative and discriminative models push each other to achieve higher levels of realism and accuracy, mirroring the dynamics between counterfeiters and law enforcement.

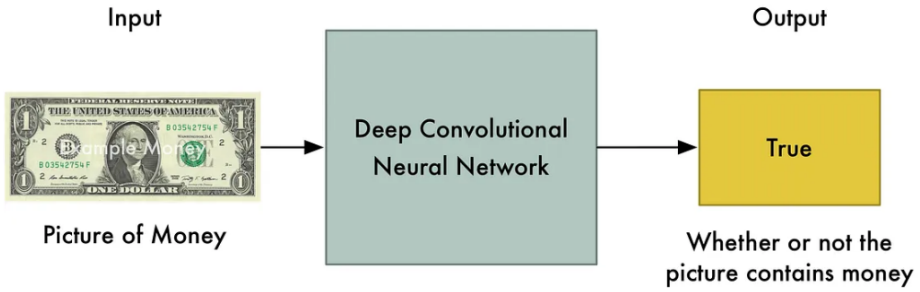


Figure 2: The Discriminator Network

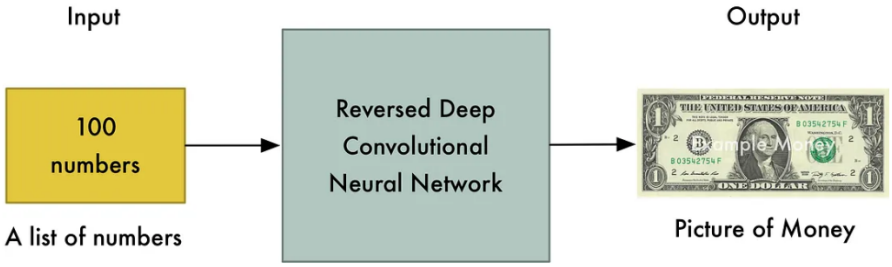


Figure 2: The Generator network

In the first round, the Generator will produce counterfeit creations that bear little resemblance to real money. This is because the Generator lacks any knowledge about the characteristics and appearance of genuine currency. At this stage, its forgeries may be considered as poor imitations due to the lack of understanding about what money is supposed to look like.

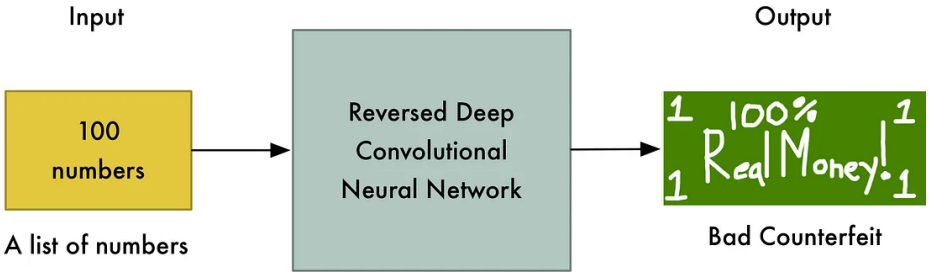


Figure 2: The initial rounds: Generator

But at this point, both the Generator and the Discriminator are not good at their respective tasks. The Generator creates poor imitations of money, while the Discriminator struggles to distinguish between real currency and the Generator's creations.

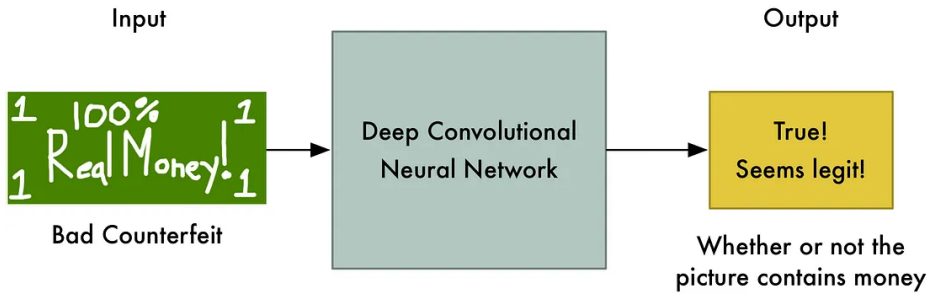


Figure 2: The initial rounds: Discriminator

At this stage, we intervene by informing the Discriminator that a particular dollar bill is counterfeit. We then present the Discriminator with a genuine dollar bill and ask it to identify the differences between the real and fake bills. The Discriminator actively searches for new details or features that can assist in distinguishing between the two. For instance, it might observe that real money typically features a portrait of a person, while the fake money lacks this characteristic. By incorporating this newfound knowledge, the Discriminator gradually improves its ability to differentiate between real and counterfeit bills. It becomes slightly more proficient at its task, enhancing its accuracy and discernment.

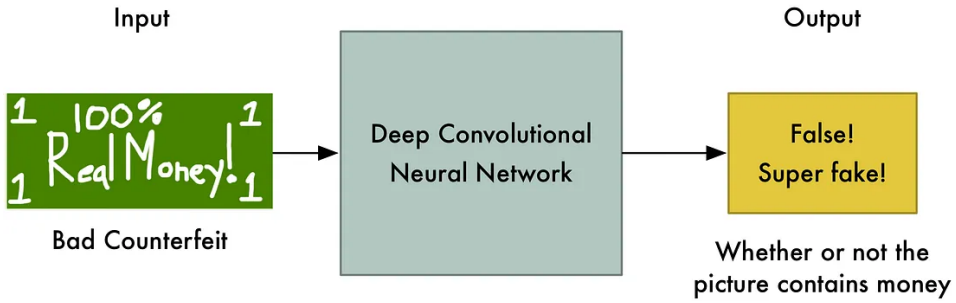


Figure 2: The Discriminator levels up

At this point, we inform the Generator that its generated money images are consistently being identified as fake, urging it to enhance its performance. Additionally, we disclose to the Generator that the Discriminator has now become adept at identifying faces. To outsmart the Discriminator and create more convincing counterfeits, we suggest to the Generator that the most effective strategy is to incorporate a face onto the counterfeit bill.

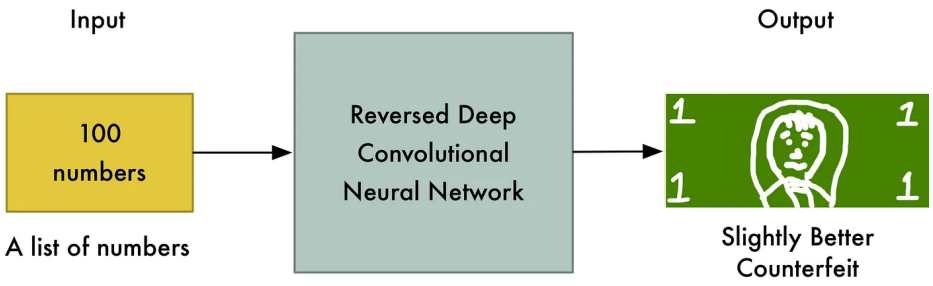


Figure 2: The Generator levels up

This iterative game between the Generator and the Discriminator repeats thousands of times until both networks reach an expert level. Over time, the Generator becomes highly proficient in generating near-perfect counterfeits, while the Discriminator transforms into a master detective, meticulously scrutinizing even the slightest flaws or mistakes. The constant back-and-forth competition and refinement between the two networks drive them towards exceptional levels of performance and accuracy.

Theory:

When applying the adversarial modeling framework, the most straightforward scenario is when both models are multilayer perceptrons. To learn the generator's distribution, denoted as pg over data x, we establish a prior on input noise variables, pz(z). The generator, represented by a multilayer perceptron with parameters θg, maps the noise variables to the data space, producing output G(z; θg) using a differentiable function.

In parallel, we introduce a second multilayer perceptron, D(x; θd), which outputs a single scalar. This discriminator model assesses the probability that a given input x originates from the actual data rather than being generated by the generator, pg. During training, we aim to maximize the probability of the discriminator assigning the correct labels to both real training examples and samples generated by G. The training process involves simultaneous optimization of both models. The discriminator is trained to maximize its ability to correctly classify real and generated examples, while the generator is trained to minimize the logarithm of the discriminator's output when applied to G(z). This objective, log(1 − D(G(z))), guides the generator to produce samples that can deceive the discriminator.

By iteratively training the generator and discriminator in this adversarial manner, the models gradually improve their performance, with the generator learning to generate more realistic samples over time.

In other words, D and G play the following two-player minimax game with value function V (G, D):

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

The Math:

Let's take a look at the loss function:

$$\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})]$$

The first term represents the discriminator's predictions on the real data. It is the expectation of the log of the discriminator output when the input is from the real data distribution. Simply put, if the discriminator is given a bunch of real data inputs, what is the discriminator's average prediction? The discriminator would want to maximize the term D(x), which is the confidence rate in response to real data. The log transformation is only used for scaling.

$$\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

The second term represents the discriminator's predictions on the fake data. D(G(z))) is the discriminator's output when fed fake data produced by the generator. The discriminator would want to minimize the term D(G(z))), which is the confidence rate in response to fake data. In contrast, the generator would want to maximize this value. Obviously, to make the expression something the discriminator would want to maximize, we take 1-D(G(z))), which is the confidence rate in response to fake data subtracted from 1, that is- the confidence rate in predicting fake data as fake.

So, we end up with two terms that the discriminator wants to maximize. Hence, the discriminator would want to maximize the sum of the two terms, while the generator would want to minimize it.

$$\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

The algorithm:

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 $p_g(\boldsymbol{z})$.
    • Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
    • 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 $p_g(\boldsymbol{z})$.
  • 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
The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

Additional theory and proof:

Nothing is taken for granted here. Every claim is very well backed. First, they go about convincing you that for a static generator, the maximum of the cost function with respect to the discriminator is a constant = -ln4. Let's look at the cost function and expand it into its integral form:

$$V(G,D) = \int_{\boldsymbol{x}} p_{\text{data}}(\boldsymbol{x}) \log(D(\boldsymbol{x}))dx + \int_{z} p_{\boldsymbol{z}}(\boldsymbol{z}) \log(1 - D(g(\boldsymbol{z})))dz$$
$$= \int_{\boldsymbol{x}} p_{\text{data}}(\boldsymbol{x}) \log(D(\boldsymbol{x})) + p_g(\boldsymbol{x}) \log(1 - D(\boldsymbol{x}))dx$$

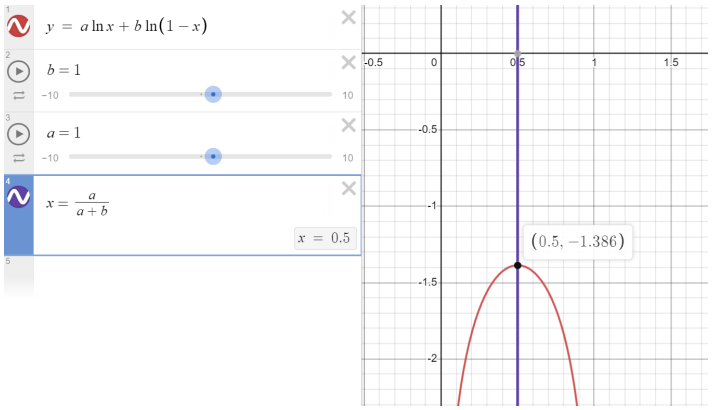
This is observably of the form: y = *a*ln(x)+*b*ln(1-x), the maximum of which would occur at x=a/(a+b).

$$\frac{\partial}{\partial y} \; a\log(y) + b\log(1-y) = 0$$

$$\frac{a}{y} + \frac{b}{1-y}(-1) = 0 \quad \Rightarrow \quad \frac{a}{y} = \frac{b}{1-y}$$

$$a - ay = by \quad \Rightarrow \quad a = ay + by = (a+b)y$$

$$\Rightarrow \quad y = \frac{a}{a+b}$$



https://www.desmos.com/calculator/kd45ijf4rh

Hence,

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

That is, the optimal discriminator is represented by a simple relation of the probability distribution of the real and fake samples. Now, if the probability distribution of the real data and the generator data/fake data are identical the optimized discriminator would return a value:

$$\mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}}[-\log 2] + \mathbb{E}_{\boldsymbol{x} \sim p_g}[-\log 2] = -\log 4$$

Moving on to the generator, considering an optimal discriminator:

$$
=\mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}}\left[\log \frac{p_{\text{data}}(\boldsymbol{x})}{P_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}\right] + \mathbb{E}_{\boldsymbol{x} \sim p_g}\left[\log \frac{p_g(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}\right]
$$

Take a look at the definition for Kullback–Leibler divergence:

$$D_{KL}(P||Q) = \mathbb{E}_{x \sim P}\left[\log \frac{P(x)}{Q(x)}\right]$$

Doing minor manipulations,

$$
\begin{aligned}
D_{KL}(P||Q) &= \mathbb{E}_{x \sim P}\left[\log \frac{P(x)}{Q(x)}\right] \\
&= \mathbb{E}_{x \sim P}\left[\log \frac{2P(x)}{2Q(x)}\right] \\
&= \mathbb{E}_{x \sim P}\left[\log \frac{P(x)}{Q(x)/_2}\right] - \log(2)
\end{aligned}
$$

We can thus, write down our function as:

$$V(D, G) = -\log(4) + KL\left(p_{\text{data}} \middle\| \frac{p_{\text{data}} + p_g}{2}\right) + KL\left(p_g \middle\| \frac{p_{\text{data}} + p_g}{2}\right)$$

Take a look at the definition for Jensen-Shannon Divergence:

$$JSD(P||Q) = \frac{1}{2} D_{KL}\left(P||\frac{P+Q}{2}\right) + \frac{1}{2} D_{KL}\left(Q||\frac{P+Q}{2}\right)$$

Our function turns out to be:

$$V(D,G) = -\log(4) + 2 \cdot JSD\left(p_{\text{data}} \| p_g\right)$$

To summarize everything until now: for an optimal discriminator, the generator is trying to minimize this quantity. We know that the minimum for any JS Divergence is 0, and this occurs when Pdata = Pg. So, this proves that the cost function has one minimum, and that occurs when Pdata=Pg. And thus,

$$\min_G \; V(D,G) = -\log(4)$$

We have established that the only solution for this global minimum is when the generative model (pg) perfectly replicates the data generating process, pdata. In simpler terms, when pg matches pdata, the generative model achieves the optimal performance, reaching the global minimum of the objective function.

Future scope:

Future scope involves improving stability, enhancing quality and diversity, enabling control over generated output, exploring multimodal generation, addressing ethical considerations, and expanding real-world applications. These advancements aim to unleash the full potential of GANs in different domains.

1. Improved Training Stability: Researchers are working on making GAN training more stable by developing better training algorithms and techniques.
2. Better Generation Quality and Diversity: Efforts are underway to improve the quality and variety of generated data by exploring new architectures and methods.
3. Controllable Generation: The goal is to allow users to have more control over the generated output, such as manipulating specific attributes or conditions.
4. Multi-Modal Generation: GANs are being extended to generate data that combines multiple modalities, like images and text.
5. Ethical Considerations and Fairness: Ensuring fairness, privacy, and transparency in GANs is important, along with detecting and addressing biases in the data and outputs.
6. Real-World Applications: GANs are finding applications in various fields, including art, gaming, healthcare, and more. Ongoing research aims to expand their applications in areas like content creation, simulation, and medical image analysis.

References:

https://machinelearningmastery.com/what-are-generative-adversarial-networks-gans/

https://en.wikipedia.org/wiki/AlexNet

https://www.ibm.com/topics/convolutional-neural-networks

https://www.mathworks.com/videos/introduction-to-deep-learning-what-are-convolutional-neural-networks--1489512765771.html

https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939

https://medium.com/lis-computer-vision-blogs/generative-adversarial-network-d1fbc4ce4499

https://towardsdatascience.com/understanding-kl-divergence-f3ddc8dff254#:~:text=KL%20divergence%20is%20a%20non,distributions%20are%20from%20each%20other.

https://towardsdatascience.com/how-to-understand-and-use-jensen-shannon-divergence-b10e11b03fd6

https://www.youtube.com/watch?v=J1aG12dLo4I