Title: Generative adversarial network

URL: https://en.wikipedia.org/wiki/Generative_adversarial_network

PageID: 50073184

Categories: Category:2014 in artificial intelligence, Category:Cognitive science, Category:Generative artificial intelligence, Category:Neural network architectures, Category:Unsupervised learning

-----

Supervised learning

Unsupervised learning

Semi-supervised learning

Self-supervised learning

Reinforcement learning

Meta-learning

Online learning

Batch learning

Curriculum learning

Rule-based learning

Neuro-symbolic AI

Neuromorphic engineering

Quantum machine learning

Classification

Generative modeling

Regression

Clustering

Dimensionality reduction

Density estimation

Anomaly detection

Data cleaning

AutoML

Association rules

Semantic analysis

Structured prediction

Feature engineering

Feature learning

Learning to rank

Grammar induction

Ontology learning

Multimodal learning

Apprenticeship learning

Decision trees

Ensembles Bagging Boosting Random forest

Bagging

Boosting

Random forest

k -NN

Linear regression

Naive Bayes

Artificial neural networks

Logistic regression

Perceptron

Relevance vector machine (RVM)

Support vector machine (SVM)

BIRCH

CURE

Hierarchical

k -means

Fuzzy

Expectation–maximization (EM)

DBSCAN

OPTICS

Mean shift

Factor analysis

CCA

ICA

LDA

NMF

PCA

PGD

t-SNE

SDL

Graphical models Bayes net Conditional random field Hidden Markov

Bayes net

Conditional random field

Hidden Markov

RANSAC

k -NN

Local outlier factor

Isolation forest

Autoencoder

Deep learning

Feedforward neural network

Recurrent neural network LSTM GRU ESN reservoir computing

LSTM

GRU

ESN

reservoir computing

Boltzmann machine Restricted

Restricted

GAN

Diffusion model

SOM

Convolutional neural network U-Net LeNet AlexNet DeepDream

U-Net

LeNet

AlexNet

DeepDream

Neural field Neural radiance field Physics-informed neural networks

Neural radiance field

Physics-informed neural networks

Transformer Vision

Vision

Mamba

Spiking neural network

Memtransistor

Electrochemical RAM (ECRAM)

Q-learning

Policy gradient

SARSA

Temporal difference (TD)

Multi-agent Self-play

Self-play

Active learning

Crowdsourcing

Human-in-the-loop

Mechanistic interpretability

RLHF

Coefficient of determination

Confusion matrix

Learning curve

ROC curve

Kernel machines

Bias–variance tradeoff

Computational learning theory

Empirical risk minimization

Occam learning

PAC learning

Statistical learning

VC theory

Topological deep learning

AAAI

ECML PKDD

NeurIPS

ICML

ICLR

IJCAI

ML

JMLR

Glossary of artificial intelligence

List of datasets for machine-learning research List of datasets in computer vision and image processing

List of datasets in computer vision and image processing

Outline of machine learning

v

t

e

A generative adversarial network (GAN) is a class of machine learning frameworks and a prominent framework for approaching generative artificial intelligence . The concept was initially developed by Ian Goodfellow and his colleagues in June 2014. [1] In a GAN, two neural networks compete with each other in the form of a zero-sum game , where one agent's gain is another agent's loss.

Given a training set, this technique learns to generate new data with the same statistics as the training set. For example, a GAN trained on photographs can generate new photographs that look at least superficially authentic to human observers, having many realistic characteristics. Though originally proposed as a form of generative model for unsupervised learning , GANs have also proved useful for semi-supervised learning , [2] fully supervised learning , [3] and reinforcement learning . [4]

The core idea of a GAN is based on the "indirect" training through the discriminator, another neural network that can tell how "realistic" the input seems, which itself is also being updated dynamically. [5] This means that the generator is not trained to minimize the distance to a specific image, but rather to fool the discriminator. This enables the model to learn in an unsupervised manner.

GANs are similar to mimicry in evolutionary biology , with an evolutionary arms race between both networks.

Definition

Mathematical

The original GAN is defined as the following game : [1]

Each probability space (Ω , μ ref) {\displaystyle (\Omega ,\mu _{\text{ref}})} defines a GAN game.

There are 2 players: generator and discriminator.

The generator's strategy set is P (Ω) {\displaystyle {\mathcal {P}}(\Omega)} , the set of all probability measures μ G {\displaystyle \mu _{G}} on Ω {\displaystyle \Omega } .

The discriminator's strategy set is the set of Markov kernels μ D : Ω → P [0 , 1] {\displaystyle \mu _{D}:\Omega \to {\mathcal {P}}[0,1]} , where P [0 , 1] {\displaystyle {\mathcal {P}}[0,1]} is the set of probability measures on [0 , 1] {\displaystyle [0,1]} .

The GAN game is a zero-sum game , with objective function L (μ G , μ D) := E x ∼ μ ref , y ∼ μ D (x) ■ [ln ■ y] + E x ∼ μ G , y ∼ μ D (x) ■ [ln ■ (1 − y)] . {\displaystyle L(\mu _{G},\mu _{D}):=\operatorname {E} _{x\sim \mu _{\text{ref}},y\sim \mu _{D}(x)}[\ln y]+\operatorname {E} _{x\sim \mu _{G},y\sim \mu _{D}(x)}[\ln(1-y)].} The generator aims to minimize the objective, and the discriminator aims to maximize the objective.

The generator's task is to approach μ G ≈ μ ref {\displaystyle \mu _{G}\approx \mu _{\text{ref}}} , that is, to match its own output distribution as closely as possible to the reference distribution. The discriminator's task is to output a value close to 1 when the input appears to be from the reference distribution, and to output a value close to 0 when the input looks like it came from the generator distribution.

In practice

The generative network generates candidates while the discriminative network evaluates them. [1] This creates a contest based on data distributions, where the generator learns to map from a latent space to the true data distribution, aiming to produce candidates that the discriminator cannot distinguish from real data. The discriminator's goal is to correctly identify these candidates, but as the generator improves, its task becomes more challenging, increasing the discriminator's error rate. [1] [6]

A known dataset serves as the initial training data for the discriminator. Training involves presenting it with samples from the training dataset until it achieves acceptable accuracy. The generator is trained based on whether it succeeds in fooling the discriminator. Typically, the generator is seeded with randomized input that is sampled from a predefined latent space (e.g. a multivariate normal distribution). Thereafter, candidates synthesized by the generator are evaluated by the discriminator. Independent backpropagation procedures are applied to both networks so that the generator produces better samples, while the discriminator becomes more skilled at flagging synthetic samples. [7] When used for image generation, the generator is typically a deconvolutional neural network , and the discriminator is a convolutional neural network .

Relation to other statistical machine learning methods

GANs are implicit generative models , [8] which means that they do not explicitly model the likelihood function nor provide a means for finding the latent variable corresponding to a given sample, unlike alternatives such as flow-based generative model .

Compared to fully visible belief networks such as WaveNet and PixelRNN and autoregressive models in general, GANs can generate one complete sample in one pass, rather than multiple passes through the network.

Compared to Boltzmann machines and linear ICA , there is no restriction on the type of function used by the network.

Since neural networks are universal approximators , GANs are asymptotically consistent . Variational autoencoders might be universal approximators, but it is not proven as of 2017. [9]

Mathematical properties

Measure-theoretic considerations

This section provides some of the mathematical theory behind these methods.

In modern probability theory based on measure theory , a probability space also needs to be equipped with a σ-algebra . As a result, a more rigorous definition of the GAN game would make the following changes:

Each probability space (Ω , B , μ ref) {\displaystyle (\Omega ,{\mathcal {B}},\mu _{\text{ref}})} defines a GAN game.

The generator's strategy set is P (Ω , B) {\displaystyle {\mathcal {P}}(\Omega ,{\mathcal {B}})} , the set of all probability measures μ G {\displaystyle \mu _{G}} on the measure-space (Ω , B) {\displaystyle (\Omega ,{\mathcal {B}})} .

The discriminator's strategy set is the set of Markov kernels μ D : (Ω , B) → P ([0 , 1] , B ([0 , 1])) {\displaystyle \mu _{D}:(\Omega ,{\mathcal {B}})\to {\mathcal {P}}([0,1],{\mathcal {B}}([0,1]))} , where B ([0 , 1]) {\displaystyle {\mathcal {B}}([0,1])} is the Borel σ-algebra on [0 , 1] {\displaystyle [0,1]} .

Since issues of measurability never arise in practice, these will not concern us further.

Choice of the strategy set

In the most generic version of the GAN game described above, the strategy set for the discriminator contains all Markov kernels μ D : Ω → P [0 , 1] {\displaystyle \mu _{D}:\Omega \to {\mathcal {P}}[0,1]} , and the strategy set for the generator contains arbitrary probability distributions μ G {\displaystyle \mu _{G}} on Ω {\displaystyle \Omega } .

However, as shown below, the optimal discriminator strategy against any μ G {\displaystyle \mu _{G}} is deterministic, so there is no loss of generality in restricting the discriminator's strategies to deterministic functions D : Ω → [0 , 1] {\displaystyle D:\Omega \to [0,1]} . In most applications, D {\displaystyle D} is a deep neural network function.

As for the generator, while μ G {\displaystyle \mu _{G}} could theoretically be any computable probability distribution, in practice, it is usually implemented as a pushforward : μ G = μ Z ■ G − 1 {\displaystyle \mu _{G}=\mu _{Z}\circ G^{-1}} . That is, start with a random variable z ~ μ Z {\displaystyle z\sim \mu _{Z}} , where μ Z {\displaystyle \mu _{Z}} is a probability distribution that is easy to compute (such as the uniform distribution , or the Gaussian distribution), then define a function G : Ω Z → Ω {\displaystyle G:\Omega _{Z}\to \Omega } . Then the distribution μ G {\displaystyle \mu _{G}} is the distribution of G (z) {\displaystyle G(z)} .

Consequently, the generator's strategy is usually defined as just G {\displaystyle G} , leaving z ~ μ Z {\displaystyle z\sim \mu _{Z}} implicit. In this formalism, the GAN game objective is L (G , D) := E x ~ μ ref ■ [ln ■ D (x)] + E z ~ μ Z ■ [ln ■ (1 − D (G (z)))] . {\displaystyle L(G,D):=\operatorname {E} _{x\sim \mu _{\text{ref}}}[\ln D(x)]+\operatorname {E} _{z\sim \mu _{Z}}[\ln(1-D(G(z)))].}

Generative reparametrization

The GAN architecture has two main components. One is casting optimization into a game, of form min G max D L (G , D) {\displaystyle \min _{G}\max _{D}L(G,D)} , which is different from the usual kind of optimization, of form min θ L (θ) {\displaystyle \min _{\theta }L(\theta)} . The other is the decomposition of μ G {\displaystyle \mu _{G}} into μ Z ■ G − 1 {\displaystyle \mu _{Z}\circ G^{-1}} , which can be understood as a reparametrization trick.

To see its significance, one must compare GAN with previous methods for learning generative models, which were plagued with "intractable probabilistic computations that arise in maximum likelihood estimation and related strategies". [1]

At the same time, Kingma and Welling [10] and Rezende et al. [11] developed the same idea of reparametrization into a general stochastic backpropagation method. Among its first applications was the variational autoencoder .

Move order and strategic equilibria

In the original paper, as well as most subsequent papers, it is usually assumed that the generator moves first , and the discriminator moves second , thus giving the following minimax game: min μ G max μ D L (μ G , μ D) := E x ∼ μ ref , y ∼ μ D (x) ■ [ln ■ y] + E x ∼ μ G , y ∼ μ D (x) ■ [ln ■ (1 − y)] . {\displaystyle \min _{\mu _{G}}\max _{\mu _{D}}L(\mu _{G},\mu _{D}):=\operatorname {E} _{x\sim \mu _{\text{ref}},y\sim \mu _{D}(x)}[\ln y]+\operatorname {E} _{x\sim \mu _{G},y\sim \mu _{D}(x)}[\ln(1-y)].}

If both the generator's and the discriminator's strategy sets are spanned by a finite number of strategies, then by the minimax theorem , min μ G max μ D L (μ G , μ D) = max μ D min μ G L (μ G , μ D) {\displaystyle \min _{\mu _{G}}\max _{\mu _{D}}L(\mu _{G},\mu _{D})=\max _{\mu _{D}}\min _{\mu _{G}}L(\mu _{G},\mu _{D})} that is, the move order does not matter.

However, since the strategy sets are both not finitely spanned, the minimax theorem does not apply, and the idea of an "equilibrium" becomes delicate. To wit, there are the following different concepts of equilibrium:

Equilibrium when generator moves first, and discriminator moves second: μ ^ G ∈ arg ■ min μ G max μ D L (μ G , μ D) , μ ^ D ∈ arg ■ max μ D L (μ ^ G , μ D) , {\displaystyle {\hat {\mu }}_{G}\in \arg \min _{\mu _{G}}\max _{\mu _{D}}L(\mu _{G},\mu _{D}),\quad {\hat {\mu }}_{D}\in \arg \max _{\mu _{D}}L({\hat {\mu }}_{G},\mu _{D}),\quad }

Equilibrium when discriminator moves first, and generator moves second: μ ^ D ∈ arg ■ max μ D min μ G L (μ G , μ D) , μ ^ G ∈ arg ■ min μ G L (μ G , μ ^ D) , {\displaystyle {\hat {\mu }}_{D}\in \arg \max _{\mu _{D}}\min _{\mu _{G}}L(\mu _{G},\mu _{D}),\quad {\hat {\mu }}_{G}\in \arg \min _{\mu _{G}}L(\mu _{G},{\hat {\mu }}_{D}),}

Nash equilibrium (μ ^ D , μ ^ G) {\displaystyle ({\hat {\mu }}_{D},{\hat {\mu }}_{G})} , which is stable under simultaneous move order: μ ^ D ∈ arg ■ max μ D L (μ ^ G , μ D) , μ ^ G ∈ arg ■ min μ G L (μ G , μ ^ D) {\displaystyle {\hat {\mu }}_{D}\in \arg \max _{\mu _{D}}L({\hat {\mu }}_{G},\mu _{D}),\quad {\hat {\mu }}_{G}\in \arg \min _{\mu _{G}}L(\mu _{G},{\hat {\mu }}_{D})}

For general games, these equilibria do not have to agree, or even to exist. For the original GAN game, these equilibria all exist, and are all equal. However, for more general GAN games, these do not necessarily exist, or agree. [12]

Main theorems for GAN game

The original GAN paper proved the following two theorems: [1]

Theorem (the optimal discriminator computes the Jensen–Shannon divergence) — For any fixed generator strategy μ G {\displaystyle \mu _{G}} , let the optimal reply be D ∗ = arg ■ max D L (μ G , D) {\displaystyle D^{*}=\arg \max _{D}L(\mu _{G},D)} , then

D ∗ (x) = d μ ref d (μ ref + μ G) L (μ G , D ∗) = 2 D J S (μ ref ; μ G) − 2 ln ■ 2 {\displaystyle {\begin{aligned}D^{*}(x)&={\frac {d\mu _{\text{ref}}}{d(\mu _{\text{ref}}+\mu _{G})}}\\[6pt]L(\mu _{G},D^{*})&=2D_{JS}(\mu _{\text{ref}};\mu _{G})-2\ln 2\end{aligned}}}

where the derivative is the Radon–Nikodym derivative , and D J S {\displaystyle D_{JS}} is the Jensen–Shannon divergence .

By Jensen's inequality,

E x ~ μ ref , y ~ μ D (x) ■ [ln ■ y] ≤ E x ~ μ ref ■ [ln ■ E y ~ μ D (x) ■ [y]] {\displaystyle \operatorname {E} _{x\sim \mu _{\text{ref}},y\sim \mu _{D}(x)}[\ln y]\leq \operatorname {E} _{x\sim \mu _{\text{ref}}}[\ln \operatorname {E} _{y\sim \mu _{D}(x)}[y]]} and similarly for the other term. Therefore, the optimal reply can be deterministic, i.e. μ D (x) = δ D (x) {\displaystyle \mu _{D}(x)=\delta _{D(x)}} for some function D : Ω → [0 , 1] {\displaystyle D:\Omega \to [0,1]} , in which case

L (μ G , μ D) := E x ~ μ ref ■ [ln ■ D (x)] + E x ~ μ G ■ [ln ■ (1 − D (x))] . {\displaystyle L(\mu _{G},\mu _{D}):=\operatorname {E} _{x\sim \mu _{\text{ref}}}[\ln D(x)]+\operatorname {E} _{x\sim \mu _{G}}[\ln(1-D(x))].}

To define suitable density functions, we define a base measure μ := μ ref + μ G {\displaystyle \mu :=\mu _{\text{ref}}+\mu _{G}} , which allows us to take the Radon–Nikodym derivatives

ρ ref = d μ ref d μ ρ G = d μ G d μ {\displaystyle \rho _{\text{ref}}={\frac {d\mu _{\text{ref}}}{d\mu }}\quad \rho _{G}={\frac {d\mu _{G}}{d\mu }}} with ρ ref + ρ G = 1 {\displaystyle \rho _{\text{ref}}+\rho _{G}=1} .

We then have

L (μ G , μ D) := ∫ μ (d x) [ρ ref (x) ln ■ (D (x)) + ρ G (x) ln ■ (1 − D (x))] . {\displaystyle L(\mu _{G},\mu _{D}):=\int \mu (dx)\left[\rho _{\text{ref}}(x)\ln(D(x))+\rho _{G}(x)\ln(1-D(x))\right].}

The integrand is just the negative cross-entropy between two Bernoulli random variables with parameters ρ ref (x) {\displaystyle \rho _{\text{ref}}(x)} and D (x) {\displaystyle D(x)} . We can write this as − H (ρ ref (x)) − D K L (ρ ref (x) ■ D (x)) {\displaystyle -H(\rho _{\text{ref}}(x))-D_{KL}(\rho _{\text{ref}}(x)\parallel D(x))} , where H {\displaystyle H} is the binary entropy function , so

L (μ G , μ D) = − ∫ μ (d x) (H (ρ ref (x)) + D K L (ρ ref (x) ■ D (x))) . {\displaystyle L(\mu _{G},\mu _{D})=-\int \mu (dx)(H(\rho _{\text{ref}}(x))+D_{KL}(\rho _{\text{ref}}(x)\parallel D(x))).}

This means that the optimal strategy for the discriminator is D (x) = ρ ref (x) {\displaystyle D(x)=\rho _{\text{ref}}(x)} , with L (μ G , μ D ∗) = − ∫ μ (d x) H (ρ ref (x)) = D J S (μ ref ■ μ G) − 2 ln ■ 2 {\displaystyle L(\mu _{G},\mu _{D}^{*})=-\int \mu (dx)H(\rho _{\text{ref}}(x))=D_{JS}(\mu _{\text{ref}}\parallel \mu _{G})-2\ln 2}

after routine calculation.

Interpretation : For any fixed generator strategy μ G {\displaystyle \mu _{G}} , the optimal discriminator keeps track of the likelihood ratio between the reference distribution and the generator distribution: D (x) 1 − D (x) = d μ ref d μ G (x) = μ ref (d x) μ G (d x) ; D (x) = σ (ln ■ μ ref (d x) − ln ■ μ G (d x)) {\displaystyle {\frac {D(x)}{1-D(x)}}={\frac {d\mu _{\text{ref}}}{d\mu _{G}}}(x)={\frac {\mu _{\text{ref}}(dx)}{\mu _{G}(dx)}};\quad D(x)=\sigma (\ln \mu _{\text{ref}}(dx)-\ln \mu _{G}(dx))} where σ {\displaystyle \sigma } is the logistic function .

In particular, if the prior probability for an image x {\displaystyle x} to come from the reference distribution is equal to 1 2 {\displaystyle {\frac {1}{2}}} , then D (x) {\displaystyle D(x)} is just the posterior probability that x {\displaystyle x} came from the reference distribution: D (x) = Pr (x came from reference distribution ■ x) . {\displaystyle D(x)=\Pr(x{\text{ came from reference distribution}}\mid x).}

Theorem (the unique equilibrium point) — For any GAN game, there exists a pair (μ ^ D , μ ^ G) {\displaystyle ({\hat {\mu }}_{D},{\hat {\mu }}_{G})} that is both a sequential equilibrium and a Nash equilibrium:

L (μ ^ G , μ ^ D) = min μ G max μ D L (μ G , μ D) = max μ D min μ G L (μ G , μ D) = − 2 ln ■ 2 μ ^ D ∈ arg ■ max μ D min μ G L (μ G , μ D) , μ ^ G ∈ arg ■ min μ G max μ D L (μ G , μ D) μ ^ D ∈ arg ■ max μ D L (μ ^ G , μ D) , μ ^ G ∈ arg ■ min μ G L (μ G , μ ^ D) ∀ x ∈ Ω , μ ^ D (x) = δ 1 2 , μ ^ G = μ ref {\displaystyle {\begin{aligned}&L;({\hat {\mu }}_{G},{\hat {\mu }}_{D})=\min _{\mu _{G}}\max _{\mu _{D}}L(\mu _{G},\mu _{D})=&\max _{\mu _{D}}\min _{\mu _{G}}L(\mu _{G},\mu _{D})=-2\ln 2\\[6pt]&{\hat {\mu }}_{D}\in \arg \max _{\mu _{D}}\min _{\mu _{G}}L(\mu _{G},\mu _{D}),&\quad {\hat {\mu }}_{G}\in \arg \min _{\mu _{G}}\max _{\mu _{D}}L(\mu _{G},\mu

_{D})\\[6pt]&{\hat {\mu }}_{D}\in \arg \max _{\mu _{D}}L({\hat {\mu }}_{G},\mu _{D}),&\quad {\hat {\mu }}_{G}\in \arg \min _{\mu _{G}}L(\mu _{G},{\hat {\mu }}_{D})\\[6pt]&\forall x\in \Omega ,{\hat {\mu }}_{D}(x)=\delta _{\frac {1}{2}},&\quad {\hat {\mu }}_{G}=\mu _{\text{ref}}\end{aligned}}}

That is, the generator perfectly mimics the reference, and the discriminator outputs 1 2 {\displaystyle {\frac {1}{2}}} deterministically on all inputs.

From the previous proposition,

arg ■ min μ G max μ D L (μ G , μ D) = μ ref ; min μ G max μ D L (μ G , μ D) = − 2 ln ■ 2. {\displaystyle \arg \min _{\mu _{G}}\max _{\mu _{D}}L(\mu _{G},\mu _{D})=\mu _{\text{ref}};\quad \min _{\mu _{G}}\max _{\mu _{D}}L(\mu _{G},\mu _{D})=-2\ln 2.}

For any fixed discriminator strategy μ D {\displaystyle \mu _{D}} , any μ G {\displaystyle \mu _{G}} concentrated on the set

{ x ■ E y ∼ μ D (x) ■ [ln ■ (1 − y)] = inf x E y ∼ μ D (x) ■ [ln ■ (1 − y)] } {\displaystyle \{x\mid \operatorname {E} _{y\sim \mu _{D}(x)}[\ln(1-y)]=\inf _{x}\operatorname {E} _{y\sim \mu _{D}(x)}[\ln(1-y)]\}} is an optimal strategy for the generator. Thus,

arg ■ max μ D min μ G L (μ G , μ D) = arg ■ max μ D E x ∼ μ ref , y ∼ μ D (x) ■ [ln ■ y] + inf x E y ∼ μ D (x) ■ [ln ■ (1 − y)] . {\displaystyle \arg \max _{\mu _{D}}\min _{\mu _{G}}L(\mu _{G},\mu _{D})=\arg \max _{\mu _{D}}\operatorname {E} _{x\sim \mu _{\text{ref}},y\sim \mu _{D}(x)}[\ln y]+\inf _{x}\operatorname {E} _{y\sim \mu _{D}(x)}[\ln(1-y)].}

By Jensen's inequality, the discriminator can only improve by adopting the deterministic strategy of always playing D (x) = E y ∼ μ D (x) ■ [y] {\displaystyle D(x)=\operatorname {E} _{y\sim \mu _{D}(x)}[y]} . Therefore,

arg ■ max μ D min μ G L (μ G , μ D) = arg ■ max D E x ∼ μ ref ■ [ln ■ D (x)] + inf x ln ■ (1 − D (x)) {\displaystyle \arg \max _{\mu _{D}}\min _{\mu _{G}}L(\mu _{G},\mu _{D})=\arg \max _{D}\operatorname {E} _{x\sim \mu _{\text{ref}}}[\ln D(x)]+\inf _{x}\ln(1-D(x))}

By Jensen's inequality,

ln ■ E x ∼ μ ref ■ [D (x)] + inf x ln ■ (1 − D (x)) = ln ■ E x ∼ μ ref ■ [D (x)] + ln ■ (1 − sup x D (x)) = ln ■ [E x ∼ μ ref ■ [D (x)] (1 − sup x D (x))] ≤ ln ■ [sup x D (x)) (1 − sup x D (x))] ≤ ln ■ 1 4 , {\displaystyle {\begin{aligned}&\ln \operatorname {E} _{x\sim \mu _{\text{ref}}}[D(x)]+\inf _{x}\ln(1-D(x))\\[6pt]={}&\ln \operatorname {E} _{x\sim \mu _{\text{ref}}}[D(x)]+\ln(1-\sup _{x}D(x))\\[6pt]={}&\ln[\operatorname {E} _{x\sim \mu _{\text{ref}}}[D(x)](1-\sup _{x}D(x))]\leq \ln[\sup _{x}D(x))(1-\sup _{x}D(x))]\leq \ln {\frac {1}{4}},\end{aligned}}}

with equality if D (x) = 1 2 {\displaystyle D(x)={\frac {1}{2}}} , so

∀ x ∈ Ω , μ ^ D (x) = δ 1 2 ; max μ D min μ G L (μ G , μ D) = − 2 ln ■ 2. {\displaystyle \forall x\in \Omega ,{\hat {\mu }}_{D}(x)=\delta _{\frac {1}{2}};\quad \max _{\mu _{D}}\min _{\mu _{G}}L(\mu _{G},\mu _{D})=-2\ln 2.}

Finally, to check that this is a Nash equilibrium, note that when μ G = μ ref {\displaystyle \mu _{G}=\mu _{\text{ref}}} , we have

L (μ G , μ D) := E x ∼ μ ref , y ∼ μ D (x) ■ [ln ■ (y (1 − y))] {\displaystyle L(\mu _{G},\mu _{D}):=\operatorname {E} _{x\sim \mu _{\text{ref}},y\sim \mu _{D}(x)}[\ln(y(1-y))]} which is always maximized by y = 1 2 {\displaystyle y={\frac {1}{2}}} .

When ∀ x ∈ Ω , μ D (x) = δ 1 2 {\displaystyle \forall x\in \Omega ,\mu _{D}(x)=\delta _{\frac {1}{2}}} , any strategy is optimal for the generator.

Training and evaluating GAN

Training

Unstable convergence

While the GAN game has a unique global equilibrium point when both the generator and discriminator have access to their entire strategy sets, the equilibrium is no longer guaranteed when they have a restricted strategy set. [12]

In practice, the generator has access only to measures of form μ Z ■ G θ – 1 {\displaystyle \mu _{Z}\circ G_{\theta }^{-1}} , where G θ {\displaystyle G_{\theta }} is a function computed by a neural network with parameters θ {\displaystyle \theta } , and μ Z {\displaystyle \mu _{Z}} is an easily sampled distribution, such as the uniform or normal distribution. Similarly, the discriminator has access only to functions of form D ζ {\displaystyle D_{\zeta }} , a function computed by a neural network with parameters ζ {\displaystyle \zeta } . These restricted strategy sets take up a vanishingly small proportion of their entire strategy sets. [13]

Further, even if an equilibrium still exists, it can only be found by searching in the high-dimensional space of all possible neural network functions. The standard strategy of using gradient descent to find the equilibrium often does not work for GAN, and often the game "collapses" into one of several failure modes. To improve the convergence stability, some training strategies start with an easier task, such as generating low-resolution images [14] or simple images (one object with uniform background), [15] and gradually increase the difficulty of the task during training. This essentially translates to applying a curriculum learning scheme. [16]

Mode collapse

GANs often suffer from mode collapse where they fail to generalize properly, missing entire modes from the input data. For example, a GAN trained on the MNIST dataset containing many samples of each digit might only generate pictures of digit 0. This was termed "the Helvetica scenario". [1]

One way this can happen is if the generator learns too fast compared to the discriminator. If the discriminator D {\displaystyle D} is held constant, then the optimal generator would only output elements of arg ■ max x D (x) {\displaystyle \arg \max _{x}D(x)} . [citation needed] So for example, if during GAN training for generating MNIST dataset, for a few epochs, the discriminator somehow prefers the digit 0 slightly more than other digits, the generator may seize the opportunity to generate only digit 0, then be unable to escape the local minimum after the discriminator improves.

Some researchers perceive the root problem to be a weak discriminative network that fails to notice the pattern of omission, while others assign blame to a bad choice of objective function . Many solutions have been proposed, but it is still an open problem. [17] [18]

Even the state-of-the-art architecture, BigGAN (2019), could not avoid mode collapse. The authors resorted to "allowing collapse to occur at the later stages of training, by which time a model is sufficiently trained to achieve good results". [19]

Two time-scale update rule

The two time-scale update rule (TTUR) is proposed to make GAN convergence more stable by making the learning rate of the generator lower than that of the discriminator. The authors argued that the generator should move slower than the discriminator, so that it does not "drive the discriminator steadily into new regions without capturing its gathered information".

They proved that a general class of games that included the GAN game, when trained under TTUR, "converges under mild assumptions to a stationary local Nash equilibrium". [20]

They also proposed using the Adam stochastic optimization [21] to avoid mode collapse, as well as the Fréchet inception distance for evaluating GAN performances.

Vanishing gradient

Conversely, if the discriminator learns too fast compared to the generator, then the discriminator could almost perfectly distinguish μ G θ , μ ref {\displaystyle \mu _{G_{\theta }},\mu _{\text{ref}}} . In such case, the generator G θ {\displaystyle G_{\theta }} could be stuck with a very high loss no matter which direction it changes its θ {\displaystyle \theta } , meaning that the gradient ∇ θ L (G θ , D ζ) {\displaystyle \nabla _{\theta }L(G_{\theta },D_{\zeta })} would be close to zero. In such case, the generator cannot learn, a case of the vanishing gradient problem . [13]

Intuitively speaking, the discriminator is too good, and since the generator cannot take any small step (only small steps are considered in gradient descent) to improve its payoff, it does not even try.

One important method for solving this problem is the Wasserstein GAN .

Evaluation

GANs are usually evaluated by Inception score (IS), which measures how varied the generator's outputs are (as classified by an image classifier, usually Inception-v3), or Fréchet inception distance (FID), which measures how similar the generator's outputs are to a reference set (as classified by a learned image featurizer, such as Inception-v3 without its final layer). Many papers that propose new GAN architectures for image generation report how their architectures break the state of the art on FID or IS.

Another evaluation method is the Learned Perceptual Image Patch Similarity (LPIPS), which starts with a learned image featurizer f θ : Image → R n {\displaystyle f_{\theta }:{\text{Image}}\to \mathbb {R} ^{n}} , and finetunes it by supervised learning on a set of (x , x ′ , p e r c e p t u a l d i f f e r e n c e ■ (x , x ′)) {\displaystyle (x,x',\operatorname {perceptual~difference} (x,x'))} , where x {\displaystyle x} is an image, x ′ {\displaystyle x'} is a perturbed version of it, and p e r c e p t u a l d i f f e r e n c e ■ (x , x ′) {\displaystyle \operatorname {perceptual~difference} (x,x')} is how much they differ, as reported by human subjects. The model is finetuned so that it can approximate ■ f θ (x) − f θ (x ′) ■ ≈ p e r c e p t u a l d i f f e r e n c e ■ (x , x ′) {\displaystyle \|f_{\theta }(x)-f_{\theta }(x')\|\approx \operatorname {perceptual~difference} (x,x')} . This finetuned model is then used to define LPIPS ■ (x , x ′) := ■ f θ (x) − f θ (x ′) ■ {\displaystyle \operatorname {LPIPS} (x,x'):=\|f_{\theta }(x)-f_{\theta }(x')\|} . [22]

Other evaluation methods are reviewed in. [23]

Variants

There is a veritable zoo of GAN variants. [24] Some of the most prominent are as follows:

Conditional GAN

Conditional GANs are similar to standard GANs except they allow the model to conditionally generate samples based on additional information. For example, if we want to generate a cat face given a dog picture, we could use a conditional GAN.

The generator in a GAN game generates μ G {\displaystyle \mu _{G}} , a probability distribution on the probability space Ω {\displaystyle \Omega } . This leads to the idea of a conditional GAN, where instead of generating one probability distribution on Ω {\displaystyle \Omega } , the generator generates a different probability distribution μ G (c) {\displaystyle \mu _{G}(c)} on Ω {\displaystyle \Omega } , for each given class label c {\displaystyle c} .

For example, for generating images that look like ImageNet , the generator should be able to generate a picture of cat when given the class label "cat".

In the original paper, [1] the authors noted that GAN can be trivially extended to conditional GAN by providing the labels to both the generator and the discriminator.

Concretely, the conditional GAN game is just the GAN game with class labels provided: L (μ G , D) := E c ∼ μ C , x ∼ μ ref (c) ■ [ln ■ D (x , c)] + E c ∼ μ C , x ∼ μ G (c) ■ [ln ■ (1 − D (x , c))] {\displaystyle L(\mu _{G},D):=\operatorname {E} _{c\sim \mu _{C},x\sim \mu _{\text{ref}}(c)}[\ln D(x,c)]+\operatorname {E} _{c\sim \mu _{C},x\sim \mu _{G}(c)}[\ln(1-D(x,c))]} where μ C {\displaystyle \mu _{C}} is a probability distribution over classes, μ ref (c) {\displaystyle \mu _{\text{ref}}(c)} is the probability distribution of real images of class c {\displaystyle c} , and μ G (c) {\displaystyle \mu _{G}(c)} the probability distribution of images generated by the generator when given class label c {\displaystyle c} .

In 2017, a conditional GAN learned to generate 1000 image classes of ImageNet . [25]

GANs with alternative architectures

The GAN game is a general framework and can be run with any reasonable parametrization of the generator G {\displaystyle G} and discriminator D {\displaystyle D} . In the original paper, the authors demonstrated it using multilayer perceptron networks and convolutional neural networks . Many alternative architectures have been tried.

Deep convolutional GAN (DCGAN): [26] For both generator and discriminator, uses only deep networks consisting entirely of convolution-deconvolution layers, that is, fully convolutional networks. [27]

Self-attention GAN (SAGAN): [28] Starts with the DCGAN, then adds residually-connected standard self-attention modules to the generator and discriminator.

Variational autoencoder GAN (VAEGAN): [29] Uses a variational autoencoder (VAE) for the generator.

Transformer GAN (TransGAN): [30] Uses the pure transformer architecture for both the generator and discriminator, entirely devoid of convolution-deconvolution layers.

Flow-GAN: [31] Uses flow-based generative model for the generator, allowing efficient computation of the likelihood function.

GANs with alternative objectives

Many GAN variants are merely obtained by changing the loss functions for the generator and discriminator.

Original GAN:

We recast the original GAN objective into a form more convenient for comparison: { min D L D (D , μ G) = – E x ~ μ G ■ [ln ■ D (x)] – E x ~ μ ref ■ [ln ■ (1 – D (x))] min G L G (D , μ G) = – E x ~ μ G ■ [ln ■ (1 – D (x))] {\displaystyle {\begin{cases}\min _{D}L_{D}(D,\mu _{G})=-\operatorname {E} _{x\sim \mu _{G}}[\ln D(x)]-\operatorname {E} _{x\sim \mu _{\text{ref}}}[\ln(1-D(x))]\\\min _{G}L_{G}(D,\mu _{G})=-\operatorname {E} _{x\sim \mu _{G}}[\ln(1-D(x))]\end{cases}}}

Original GAN, non-saturating loss:

This objective for generator was recommended in the original paper for faster convergence. [1] L G = E x ~ μ G ■ [ln ■ D (x)] {\displaystyle L_{G}=\operatorname {E} _{x\sim \mu _{G}}[\ln D(x)]} The effect of using this objective is analyzed in Section 2.2.2 of Arjovsky et al. [32]

Original GAN, maximum likelihood:

L G = E x ~ μ G ■ [(exp ■ σ – 1 ■ D) (x)] {\displaystyle L_{G}=\operatorname {E} _{x\sim \mu _{G}}[({\exp }\circ \sigma ^{-1}\circ D)(x)]} where σ {\displaystyle \sigma } is the logistic function. When the discriminator is optimal, the generator gradient is the same as in maximum likelihood estimation , even though GAN cannot perform maximum likelihood estimation itself . [33] [34]

Hinge loss GAN : [35] L D = – E x ~ p ref ■ [min (0 , – 1 + D (x))] – E x ~ μ G ■ [min (0 , – 1 – D (x))] {\displaystyle L_{D}=-\operatorname {E} _{x\sim p_{\text{ref}}}\left[\min \left(0,-1+D(x)\right)\right]-\operatorname {E} _{x\sim \mu _{G}}\left[\min \left(0,-1-D\left(x\right)\right)\right]} L G = – E x ~ μ G ■ [D (x)] {\displaystyle L_{G}=-\operatorname {E} _{x\sim \mu _{G}}[D(x)]} Least squares GAN: [36] L D = E x ~ μ ref ■ [(D (x) – b) 2] + E x ~ μ G ■ [(D (x) – a) 2] {\displaystyle L_{D}=\operatorname {E} _{x\sim \mu _{\text{ref}}}[(D(x)-b)^{2}]+\operatorname {E} _{x\sim \mu _{G}}[(D(x)-a)^{2}]} L G = E x ~ μ G ■ [(D (x) – c) 2] {\displaystyle L_{G}=\operatorname {E} _{x\sim \mu _{G}}[(D(x)-c)^{2}]} where a , b , c {\displaystyle a,b,c} are parameters to be chosen. The authors recommended a = – 1 , b = 1 , c = 0 {\displaystyle a=-1,b=1,c=0} .

Wasserstein GAN (WGAN)

The Wasserstein GAN modifies the GAN game at two points:

The discriminator's strategy set is the set of measurable functions of type D : Ω → R {\displaystyle D:\Omega \to \mathbb {R} } with bounded Lipschitz norm : ■ D ■ L ≤ K {\displaystyle \|D\|_{L}\leq K} , where K {\displaystyle K} is a fixed positive constant.

The objective is L W G A N (μ G , D) := E x ∼ μ G ■ [D (x)] − E x ∼ μ ref [D (x)] {\displaystyle L_{WGAN}(\mu _{G},D):=\operatorname {E} _{x\sim \mu _{G}}[D(x)]-\mathbb {E} _{x\sim \mu _{\text{ref}}}[D(x)]}

One of its purposes is to solve the problem of mode collapse (see above). [13] The authors claim "In no experiment did we see evidence of mode collapse for the WGAN algorithm".

GANs with more than two players

Adversarial autoencoder

An adversarial autoencoder (AAE) [37] is more autoencoder than GAN. The idea is to start with a plain autoencoder , but train a discriminator to discriminate the latent vectors from a reference distribution (often the normal distribution).

InfoGAN

In conditional GAN, the generator receives both a noise vector z {\displaystyle z} and a label c {\displaystyle c} , and produces an image G (z , c) {\displaystyle G(z,c)} . The discriminator receives image-label pairs (x , c) {\displaystyle (x,c)} , and computes D (x , c) {\displaystyle D(x,c)} .

When the training dataset is unlabeled, conditional GAN does not work directly.

The idea of InfoGAN is to decree that every latent vector in the latent space can be decomposed as (z , c) {\displaystyle (z,c)} : an incompressible noise part z {\displaystyle z} , and an informative label part c {\displaystyle c} , and encourage the generator to comply with the decree, by encouraging it to maximize I (c , G (z , c)) {\displaystyle I(c,G(z,c))} , the mutual information between c {\displaystyle c} and G (z , c) {\displaystyle G(z,c)} , while making no demands on the mutual information z {\displaystyle z} between G (z , c) {\displaystyle G(z,c)} .

Unfortunately, I (c , G (z , c)) {\displaystyle I(c,G(z,c))} is intractable in general, The key idea of InfoGAN is Variational Mutual Information Maximization: [38] indirectly maximize it by maximizing a lower bound I ^ (G , Q) = E z ∼ μ Z , c ∼ μ C [ln ■ Q (c ■ G (z , c))] ; I (c , G (z , c)) ≥ sup Q I ^ (G , Q) {\displaystyle {\hat {I}}(G,Q)=\mathbb {E} _{z\sim \mu _{Z},c\sim \mu _{C}}[\ln Q(c\mid G(z,c))];\quad I(c,G(z,c))\geq \sup _{Q}{\hat {I}}(G,Q)} where Q {\displaystyle Q} ranges over all Markov kernels of type Q : Ω Y → P (Ω C) {\displaystyle Q:\Omega _{Y}\to {\mathcal {P}}(\Omega _{C})} .

The InfoGAN game is defined as follows: [39]

Three probability spaces define an InfoGAN game:

(Ω X , μ ref) {\displaystyle (\Omega _{X},\mu _{\text{ref}})} , the space of reference images.

(Ω Z , μ Z) {\displaystyle (\Omega _{Z},\mu _{Z})} , the fixed random noise generator.

(Ω C , μ C) {\displaystyle (\Omega _{C},\mu _{C})} , the fixed random information generator.

There are 3 players in 2 teams: generator, Q, and discriminator. The generator and Q are on one team, and the discriminator on the other team.

The objective function is L (G , Q , D) = L G A N (G , D) − λ I ^ (G , Q) {\displaystyle L(G,Q,D)=L_{GAN}(G,D)-\lambda {\hat {I}}(G,Q)} where L G A N (G , D) = E x ∼ μ ref , ■ [ln ■ D (x)] + E z ∼ μ Z ■ [ln ■ (1 − D (G (z , c)))] {\displaystyle L_{GAN}(G,D)=\operatorname {E} _{x\sim \mu _{\text{ref}},}[\ln D(x)]+\operatorname {E} _{z\sim \mu _{Z}}[\ln(1-D(G(z,c)))]} is the original GAN game objective, and I ^ (G , Q) = E z ∼ μ Z , c ∼ μ C [ln ■ Q (c ■ G (z , c))] {\displaystyle {\hat {I}}(G,Q)=\mathbb {E} _{z\sim \mu _{Z},c\sim \mu _{C}}[\ln Q(c\mid G(z,c))]}

Generator-Q team aims to minimize the objective, and discriminator aims to maximize it: min G , Q max D L (G , Q , D) {\displaystyle \min _{G,Q}\max _{D}L(G,Q,D)}

Bidirectional GAN (BiGAN)

The standard GAN generator is a function of type G : Ω Z → Ω X {\displaystyle G:\Omega _{Z}\to \Omega _{X}} , that is, it is a mapping from a latent space Ω Z {\displaystyle \Omega _{Z}} to the image space Ω X {\displaystyle \Omega _{X}} . This can be understood as a "decoding" process, whereby every latent vector z ∈ Ω Z {\displaystyle z\in \Omega _{Z}} is a code for an image x ∈ Ω X {\displaystyle x\in \Omega _{X}} , and the generator performs the decoding. This naturally leads to the idea of training another network that performs "encoding", creating an autoencoder out of the encoder-generator pair.

Already in the original paper, [1] the authors noted that "Learned approximate inference can be performed by training an auxiliary network to predict z {\displaystyle z} given x {\displaystyle x} ". The bidirectional GAN architecture performs exactly this. [40]

The BiGAN is defined as follows:

Two probability spaces define a BiGAN game:

(Ω X , μ X) {\displaystyle (\Omega _{X},\mu _{X})} , the space of reference images.

(Ω Z , μ Z) {\displaystyle (\Omega _{Z},\mu _{Z})} , the latent space.

There are 3 players in 2 teams: generator, encoder, and discriminator. The generator and encoder are on one team, and the discriminator on the other team.

The generator's strategies are functions G : Ω Z → Ω X {\displaystyle G:\Omega _{Z}\to \Omega _{X}} , and the encoder's strategies are functions E : Ω X → Ω Z {\displaystyle E:\Omega _{X}\to \Omega _{Z}} . The discriminator's strategies are functions D : Ω X → [0 , 1] {\displaystyle D:\Omega _{X}\to [0,1]} .

The objective function is L (G , E , D) = E x ∼ μ X [ln ■ D (x , E (x))] + E z ∼ μ Z [ln ■ (1 − D (G (z) , z))] {\displaystyle L(G,E,D)=\mathbb {E} _{x\sim \mu _{X}}[\ln D(x,E(x))]+\mathbb {E} _{z\sim \mu _{Z}}[\ln(1-D(G(z),z))]}

Generator-encoder team aims to minimize the objective, and discriminator aims to maximize it: min G , E max D L (G , E , D) {\displaystyle \min _{G,E}\max _{D}L(G,E,D)}

In the paper, they gave a more abstract definition of the objective as: L (G , E , D) = E (x , z) ∼ μ E , X [ln ■ D (x , z)] + E (x , z) ∼ μ G , Z [ln ■ (1 − D (x , z))] {\displaystyle L(G,E,D)=\mathbb {E} _{(x,z)\sim \mu _{E,X}}[\ln D(x,z)]+\mathbb {E} _{(x,z)\sim \mu _{G,Z}}[\ln(1-D(x,z))]} where μ E , X (d x , d z) = μ X (d x) ⋅ δ E (x) (d z) {\displaystyle \mu _{E,X}(dx,dz)=\mu _{X}(dx)\cdot \delta _{E(x)}(dz)} is the probability distribution on Ω X × Ω Z {\displaystyle \Omega _{X}\times \Omega _{Z}} obtained by pushing μ X {\displaystyle \mu _{X}} forward via x ■ (x , E (x)) {\displaystyle x\mapsto (x,E(x))} , and μ G , Z (d x , d z) = δ G (z) (d x) ⋅ μ Z (d z) {\displaystyle \mu _{G,Z}(dx,dz)=\delta _{G(z)}(dx)\cdot \mu _{Z}(dz)} is the probability distribution on Ω X × Ω Z {\displaystyle \Omega _{X}\times \Omega _{Z}} obtained by pushing μ Z {\displaystyle \mu _{Z}} forward via z ■ (G (x) , z) {\displaystyle z\mapsto (G(x),z)} .

Applications of bidirectional models include semi-supervised learning , [41] interpretable machine learning , [42] and neural machine translation . [43]

CycleGAN

CycleGAN is an architecture for performing translations between two domains, such as between photos of horses and photos of zebras, or photos of night cities and photos of day cities.

The CycleGAN game is defined as follows: [44]

There are two probability spaces (Ω X , μ X) , (Ω Y , μ Y) {\displaystyle (\Omega _{X},\mu _{X}),(\Omega _{Y},\mu _{Y})} , corresponding to the two domains needed for translations fore-and-back.

There are 4 players in 2 teams: generators G X : Ω X → Ω Y , G Y : Ω Y → Ω X {\displaystyle G_{X}:\Omega _{X}\to \Omega _{Y},G_{Y}:\Omega _{Y}\to \Omega _{X}} , and discriminators D X : Ω X → [0 , 1] , D Y : Ω Y → [0 , 1] {\displaystyle D_{X}:\Omega _{X}\to [0,1],D_{Y}:\Omega _{Y}\to [0,1]} .

The objective function is L (G X , G Y , D X , D Y) = L G A N (G X , D X) + L G A N (G Y , D Y) + λ L c y c l e (G X , G Y) {\displaystyle

L(G_{X},G_{Y},D_{X},D_{Y})=L_{GAN}(G_{X},D_{X})+L_{GAN}(G_{Y},D_{Y})+\lambda L_{cycle}(G_{X},G_{Y})}

where λ {\displaystyle \lambda } is a positive adjustable parameter, L G A N {\displaystyle L_{GAN}} is the GAN game objective, and L c y c l e {\displaystyle L_{cycle}} is the cycle consistency loss : L c y c l e (G X , G Y) = E x ∼ μ X ■ G X (G Y (x)) − x ■ + E y ∼ μ Y ■ G Y (G X (y)) − y ■ {\displaystyle L_{cycle}(G_{X},G_{Y})=E_{x\sim \mu _{X}}\|G_{X}(G_{Y}(x))-x\|+E_{y\sim \mu _{Y}}\|G_{Y}(G_{X}(y))-y\|} The generators aim to minimize the objective, and the discriminators aim to maximize it: min G X , G Y max D X , D Y L (G X , G Y , D X , D Y) {\displaystyle \min _{G_{X},G_{Y}}\max _{D_{X},D_{Y}}L(G_{X},G_{Y},D_{X},D_{Y})}

Unlike previous work like pix2pix, [45] which requires paired training data, cycleGAN requires no paired data. For example, to train a pix2pix model to turn a summer scenery photo to winter scenery photo and back, the dataset must contain pairs of the same place in summer and winter, shot at the same angle; cycleGAN would only need a set of summer scenery photos, and an unrelated set of winter scenery photos.

GANs with particularly large or small scales

BigGAN

The BigGAN is essentially a self-attention GAN trained on a large scale (up to 80 million parameters) to generate large images of ImageNet (up to 512 x 512 resolution), with numerous engineering tricks to make it converge. [19] [46]

Invertible data augmentation

When there is insufficient training data, the reference distribution μ ref {\displaystyle \mu _{\text{ref}}} cannot be well-approximated by the empirical distribution given by the training dataset. In such cases, data augmentation can be applied, to allow training GAN on smaller datasets. Naïve data augmentation, however, brings its problems.

Consider the original GAN game, slightly reformulated as follows: { min D L D (D , μ G) = − E x ∼ μ ref ■ [ln ■ D (x)] − E x ∼ μ G ■ [ln ■ (1 − D (x))] min G L G (D , μ G) = − E x ∼ μ G ■ [ln ■ (1 − D (x))] {\displaystyle {\begin{cases}\min _{D}L_{D}(D,\mu _{G})=-\operatorname {E} _{x\sim \mu _{\text{ref}}}[\ln D(x)]-\operatorname {E} _{x\sim \mu _{G}}[\ln(1-D(x))]\\\min _{G}L_{G}(D,\mu _{G})=-\operatorname {E} _{x\sim \mu _{G}}[\ln(1-D(x))]\end{cases}}} Now we use data augmentation by randomly sampling semantic-preserving transforms T : Ω → Ω {\displaystyle T:\Omega \to \Omega } and applying them to the dataset, to obtain the reformulated GAN game: { min D L D (D , μ G) = − E x ∼ μ ref , T ∼ μ trans ■ [ln ■ D (T (x))] − E x ∼ μ G ■ [ln ■ (1 − D (x))] min G L G (D , μ G) = − E x ∼ μ G ■ [ln ■ (1 − D (x))] {\displaystyle {\begin{cases}\min _{D}L_{D}(D,\mu _{G})=-\operatorname {E} _{x\sim \mu _{\text{ref}},T\sim \mu _{\text{trans}}}[\ln D(T(x))]-\operatorname {E} _{x\sim \mu _{G}}[\ln(1-D(x))]\\\min _{G}L_{G}(D,\mu _{G})=-\operatorname {E} _{x\sim \mu _{G}}[\ln(1-D(x))]\end{cases}}} This is equivalent to a GAN game with a different distribution μ ref ′ {\displaystyle \mu _{\text{ref}}'} , sampled by T (x) {\displaystyle T(x)} , with x ∼ μ ref , T ∼ μ trans {\displaystyle x\sim \mu _{\text{ref}},T\sim \mu _{\text{trans}}} . For example, if μ ref {\displaystyle \mu _{\text{ref}}} is the distribution of images in ImageNet, and μ trans {\displaystyle \mu _{\text{trans}}} samples identity-transform with probability 0.5, and horizontal-reflection with probability 0.5, then μ ref ′ {\displaystyle \mu _{\text{ref}}'} is the distribution of images in ImageNet and horizontally-reflected ImageNet, combined.

The result of such training would be a generator that mimics μ ref ′ {\displaystyle \mu _{\text{ref}}'} . For example, it would generate images that look like they are randomly cropped, if the data augmentation uses random cropping.

The solution is to apply data augmentation to both generated and real images: { min D L D (D , μ G) = − E x ∼ μ ref , T ∼ μ trans ■ [ln ■ D (T (x))] − E x ∼ μ G , T ∼ μ trans ■ [ln ■ (1 − D (T (x)))] min G L G (D , μ G) = − E x ∼ μ G , T ∼ μ trans ■ [ln ■ (1 − D (T (x)))] {\displaystyle {\begin{cases}\min _{D}L_{D}(D,\mu _{G})=-\operatorname {E} _{x\sim \mu _{\text{ref}},T\sim \mu _{\text{trans}}}[\ln D(T(x))]-\operatorname {E} _{x\sim \mu _{G},T\sim \mu _{\text{trans}}}[\ln(1-D(T(x)))]\\\min _{G}L_{G}(D,\mu _{G})=-\operatorname {E} _{x\sim \mu _{G},T\sim \mu _{\text{trans}}}[\ln(1-D(T(x)))]\end{cases}}} The authors demonstrated high-quality generation using just 100-picture-large datasets. [47]

The StyleGAN-2-ADA paper points out a further point on data augmentation: it must be invertible . [48] Continue with the example of generating ImageNet pictures. If the data augmentation is "randomly rotate the picture by 0, 90, 180, 270 degrees with equal probability", then there is no way for the generator to know which is the true orientation: Consider two generators G , G ′ {\displaystyle G,G'} , such that for any latent z {\displaystyle z} , the generated image G (z) {\displaystyle G(z)} is a 90-degree rotation of G ′ (z) {\displaystyle G'(z)} . They would have exactly the same expected loss, and so neither is preferred over the other.

The solution is to only use invertible data augmentation: instead of "randomly rotate the picture by 0, 90, 180, 270 degrees with equal probability", use "randomly rotate the picture by 90, 180, 270 degrees with 0.1 probability, and keep the picture as it is with 0.7 probability". This way, the generator is still rewarded to keep images oriented the same way as un-augmented ImageNet pictures.

Abstractly, the effect of randomly sampling transformations T : Ω → Ω {\displaystyle T:\Omega \to \Omega } from the distribution μ trans {\displaystyle \mu _{\text{trans}}} is to define a Markov kernel K trans : Ω → P (Ω) {\displaystyle K_{\text{trans}}:\Omega \to {\mathcal {P}}(\Omega)} . Then, the data-augmented GAN game pushes the generator to find some μ ^ G ∈ P (Ω) {\displaystyle {\hat {\mu }}_{G}\in {\mathcal {P}}(\Omega)} , such that K trans ∗ μ ref = K trans ∗ μ ^ G {\displaystyle K_{\text{trans}}*\mu _{\text{ref}}=K_{\text{trans}}*{\hat {\mu }}_{G}} where ∗ {\displaystyle *} is the Markov kernel convolution .

A data-augmentation method is defined to be invertible if its Markov kernel K trans {\displaystyle K_{\text{trans}}} satisfies K trans ∗ μ = K trans ∗ μ ′ ■ μ = μ ′ ∀ μ , μ ′ ∈ P (Ω) {\displaystyle K_{\text{trans}}*\mu =K_{\text{trans}}*\mu '\implies \mu =\mu '\quad \forall \mu ,\mu '\in {\mathcal {P}}(\Omega)} Immediately by definition, we see that composing multiple invertible data-augmentation methods results in yet another invertible method. Also by definition, if the data-augmentation method is invertible, then using it in a GAN game does not change the optimal strategy μ ^ G {\displaystyle {\hat {\mu }}_{G}} for the generator, which is still μ ref {\displaystyle \mu _{\text{ref}}} .

There are two prototypical examples of invertible Markov kernels:

Discrete case : Invertible stochastic matrices , when Ω {\displaystyle \Omega } is finite.

For example, if Ω = { ↑ , ↓ , ← , → } {\displaystyle \Omega =\{\uparrow ,\downarrow ,\leftarrow ,\rightarrow \}} is the set of four images of an arrow, pointing in 4 directions, and the data augmentation is "randomly rotate the picture by 90, 180, 270 degrees with probability p {\displaystyle p} , and keep the picture as it is with probability (1 − 3 p) {\displaystyle (1-3p)} ", then the Markov kernel K trans {\displaystyle K_{\text{trans}}} can be represented as a stochastic matrix: [K trans] = [(1 − 3 p) p p p p (1 − 3 p) p p p p (1 − 3 p) p p p p (1 − 3 p)] {\displaystyle [K_{\text{trans}}]={\begin{bmatrix}(1-3p)&p;&p;&p;\\p&(1-3p)&p;&p;\\p&p;&(1-3p)&p;\\p&p;&p;&(1-3p)\end{bmatrix}}} and K trans {\displaystyle K_{\text{trans}}} is an invertible kernel iff [K trans] {\displaystyle [K_{\text{trans}}]} is an invertible matrix, that is, p ≠ 1 / 4 {\displaystyle p\neq 1/4} .

Continuous case : The gaussian kernel, when Ω = R n {\displaystyle \Omega =\mathbb {R} ^{n}} for some n ≥ 1 {\displaystyle n\geq 1} .

For example, if Ω = R 256 2 {\displaystyle \Omega =\mathbb {R} ^{256^{2}}} is the space of 256x256 images, and the data-augmentation method is "generate a gaussian noise z ~ N (0 , I 256 2) {\displaystyle z\sim {\mathcal {N}}(0,I_{256^{2}})} , then add ■ z {\displaystyle \epsilon z} to the image", then K trans {\displaystyle K_{\text{trans}}} is just convolution by the density function of N (0 , ■ 2 I 256 2) {\displaystyle {\mathcal {N}}(0,\epsilon ^{2}I_{256^{2}})} . This is invertible, because convolution by a gaussian is just convolution by the heat kernel , so given any μ ∈ P (R n) {\displaystyle \mu \in {\mathcal {P}}(\mathbb {R} ^{n})} , the convolved distribution K trans ∗ μ {\displaystyle K_{\text{trans}}*\mu } can be obtained by heating up R n {\displaystyle \mathbb {R} ^{n}} precisely according to μ {\displaystyle \mu } , then wait for time ■ 2 / 4 {\displaystyle \epsilon ^{2}/4} . With that, we can recover μ {\displaystyle \mu } by running the heat equation backwards in time for ■ 2 / 4 {\displaystyle \epsilon ^{2}/4} .

More examples of invertible data augmentations are found in the paper. [48]

SinGAN

SinGAN pushes data augmentation to the limit, by using only a single image as training data and performing data augmentation on it. The GAN architecture is adapted to this training method by using a multi-scale pipeline.

The generator G {\displaystyle G} is decomposed into a pyramid of generators G = G 1 ■ G 2 ■ ■ ■ G N {\displaystyle G=G_{1}\circ G_{2}\circ \cdots \circ G_{N}} , with the lowest one generating the image G N (z N) {\displaystyle G_{N}(z_{N})} at the lowest resolution, then the generated image is scaled up to r (G N (z N)) {\displaystyle r(G_{N}(z_{N}))} , and fed to the next level to generate an image G N – 1 (z N – 1 + r (G N (z N))) {\displaystyle G_{N-1}(z_{N-1}+r(G_{N}(z_{N})))} at a higher resolution, and so on. The discriminator is decomposed into a pyramid as well. [49]

StyleGAN series

The StyleGAN family is a series of architectures published by Nvidia 's research division.

Progressive GAN

Progressive GAN [14] is a method for training GAN for large-scale image generation stably, by growing a GAN generator from small to large scale in a pyramidal fashion. Like SinGAN, it decomposes the generator as G = G 1 ■ G 2 ■ ■ ■ G N {\displaystyle G=G_{1}\circ G_{2}\circ \cdots \circ G_{N}} , and the discriminator as D = D 1 ■ D 2 ■ ■ ■ D N {\displaystyle D=D_{1}\circ D_{2}\circ \cdots \circ D_{N}} .

During training, at first only G N , D N {\displaystyle G_{N},D_{N}} are used in a GAN game to generate 4x4 images. Then G N – 1 , D N – 1 {\displaystyle G_{N-1},D_{N-1}} are added to reach the second stage of GAN game, to generate 8x8 images, and so on, until we reach a GAN game to generate 1024x1024 images.

To avoid shock between stages of the GAN game, each new layer is "blended in" (Figure 2 of the paper [14]). For example, this is how the second stage GAN game starts:

Just before, the GAN game consists of the pair G N , D N {\displaystyle G_{N},D_{N}} generating and discriminating 4x4 images.

Just after, the GAN game consists of the pair ((1 – α) + α · G N – 1) ■ u ■ G N , D N ■ d ■ ((1 – α) + α · D N – 1) {\displaystyle ((1-\alpha)+\alpha \cdot G_{N-1})\circ u\circ G_{N},D_{N}\circ d\circ ((1-\alpha)+\alpha \cdot D_{N-1})} generating and discriminating 8x8 images. Here, the functions u , d {\displaystyle u,d} are image up- and down-sampling functions, and α {\displaystyle \alpha } is a blend-in factor (much like an alpha in image composing) that smoothly glides from 0 to 1.

StyleGAN-1

StyleGAN-1 is designed as a combination of Progressive GAN with neural style transfer . [50]

The key architectural choice of StyleGAN-1 is a progressive growth mechanism, similar to Progressive GAN. Each generated image starts as a constant 4 × 4 × 512 {\displaystyle 4\times 4\times 512} array, and repeatedly passed through style blocks. Each style block applies a "style latent vector" via affine transform ("adaptive instance normalization"), similar to how neural style transfer uses Gramian matrix . It then adds noise, and normalize (subtract the mean, then divide by the variance).

At training time, usually only one style latent vector is used per image generated, but sometimes two ("mixing regularization") in order to encourage each style block to independently perform its stylization without expecting help from other style blocks (since they might receive an entirely different style latent vector).

After training, multiple style latent vectors can be fed into each style block. Those fed to the lower layers control the large-scale styles, and those fed to the higher layers control the fine-detail styles.

Style-mixing between two images x , x ′ {\displaystyle x,x'} can be performed as well. First, run a gradient descent to find z , z ′ {\displaystyle z,z'} such that G (z) ≈ x , G (z ′) ≈ x ′ {\displaystyle

G(z)\approx x,G(z')\approx x'} . This is called "projecting an image back to style latent space". Then, z {\displaystyle z} can be fed to the lower style blocks, and z ′ {\displaystyle z'} to the higher style blocks, to generate a composite image that has the large-scale style of x {\displaystyle x} , and the fine-detail style of x ′ {\displaystyle x'} . Multiple images can also be composed this way.

StyleGAN-2

StyleGAN-2 improves upon StyleGAN-1, by using the style latent vector to transform the convolution layer's weights instead, thus solving the "blob" problem. [51]

This was updated by the StyleGAN-2-ADA ("ADA" stands for "adaptive"), [48] which uses invertible data augmentation as described above. It also tunes the amount of data augmentation applied by starting at zero, and gradually increasing it until an "overfitting heuristic" reaches a target level, thus the name "adaptive".

StyleGAN-3

StyleGAN-3 [52] improves upon StyleGAN-2 by solving the "texture sticking" problem, which can be seen in the official videos. [53] They analyzed the problem by the Nyquist–Shannon sampling theorem , and argued that the layers in the generator learned to exploit the high-frequency signal in the pixels they operate upon.

To solve this, they proposed imposing strict lowpass filters between each generator's layers, so that the generator is forced to operate on the pixels in a way faithful to the continuous signals they represent, rather than operate on them as merely discrete signals. They further imposed rotational and translational invariance by using more signal filters . The resulting StyleGAN-3 is able to solve the texture sticking problem, as well as generating images that rotate and translate smoothly.

Other uses

Other than for generative and discriminative modelling of data, GANs have been used for other things.

GANs have been used for transfer learning to enforce the alignment of the latent feature space, such as in deep reinforcement learning . [54] This works by feeding the embeddings of the source and target task to the discriminator which tries to guess the context. The resulting loss is then (inversely) backpropagated through the encoder.

Applications

Science

Iteratively reconstruct astronomical images [55]

Simulate gravitational lensing for dark matter research. [56] [57] [58]

Model the distribution of dark matter in a particular direction in space and to predict the gravitational lensing that will occur. [59] [60]

Model high energy jet formation [61] and showers through calorimeters of high-energy physics experiments. [62] [63] [64] [65]

Approximate bottlenecks in computationally expensive simulations of particle physics experiments. Applications in the context of present and proposed CERN experiments have demonstrated the potential of these methods for accelerating simulation and/or improving simulation fidelity. [66] [67]

Reconstruct velocity and scalar fields in turbulent flows. [68] [69] [70]

GAN-generated molecules were validated experimentally in mice. [71] [72]

Medical

One of the major concerns in medical imaging is preserving patient privacy. Due to these reasons, researchers often face difficulties in obtaining medical images for their research purposes. GAN has been used for generating synthetic medical images , such as MRI and PET images to address this challenge. [73]

GAN can be used to detect glaucomatous images helping the early diagnosis which is essential to avoid partial or total loss of vision. [74]

GANs have been used to create forensic facial reconstructions of deceased historical figures. [75]

Malicious

Concerns have been raised about the potential use of GAN-based human image synthesis for sinister purposes, e.g., to produce fake, possibly incriminating, photographs and videos. [76] GANs can be used to generate unique, realistic profile photos of people who do not exist, in order to automate creation of fake social media profiles. [77]

In 2019 the state of California considered [78] and passed on October 3, 2019, the bill AB-602 , which bans the use of human image synthesis technologies to make fake pornography without the consent of the people depicted, and bill AB-730 , which prohibits distribution of manipulated videos of a political candidate within 60 days of an election. Both bills were authored by Assembly member Marc Berman and signed by Governor Gavin Newsom . The laws went into effect in 2020. [79]

DARPA's Media Forensics program studies ways to counteract fake media, including fake media produced using GANs. [80]

Fashion, art and advertising

GANs can be used to generate art; The Verge wrote in March 2019 that "The images created by GANs have become the defining look of contemporary AI art." [81] GANs can also be used to

inpaint photographs [82]

generate fashion models, [83] shadows, [84] photorealistic renders of interior design , industrial design , shoes, etc. [85] Such networks were reported to be used by Facebook . [86]

Some have worked with using GAN for artistic creativity, as "creative adversarial network". [87] [88] A GAN, trained on a set of 15,000 portraits from WikiArt from the 14th to the 19th century, created the 2018 painting Edmond de Belamy , which sold for US$432,500. [89]

GANs were used by the video game modding community to up-scale low-resolution 2D textures in old video games by recreating them in 4k or higher resolutions via image training, and then down-sampling them to fit the game's native resolution (resembling supersampling anti-aliasing). [90]

In 2020, Artbreeder was used to create the main antagonist in the sequel to the psychological web horror series Ben Drowned . The author would later go on to praise GAN applications for their ability to help generate assets for independent artists who are short on budget and manpower. [91] [92]

In May 2020, Nvidia researchers taught an AI system (termed "GameGAN") to recreate the game of Pac-Man simply by watching it being played. [93] [94]

In August 2019, a large dataset consisting of 12,197 MIDI songs each with paired lyrics and melody alignment was created for neural melody generation from lyrics using conditional GAN-LSTM (refer to sources at GitHub AI Melody Generation from Lyrics). [95]

Miscellaneous

GANs have been used to

show how an individual's appearance might change with age. [96]

reconstruct 3D models of objects from images , [97]

generate novel objects as 3D point clouds, [98]

model patterns of motion in video. [99]

inpaint missing features in maps, transfer map styles in cartography [100] or augment street view imagery. [101]

use feedback to generate images and replace image search systems. [102]

visualize the effect that climate change will have on specific houses. [103]

reconstruct an image of a person's face after listening to their voice. [104]

produces videos of a person speaking, given only a single photo of that person. [105]

recurrent sequence generation. [106]

History

In 1991, Juergen Schmidhuber published "artificial curiosity", neural networks in a zero-sum game . [107] The first network is a generative model that models a probability distribution over output patterns. The second network learns by gradient descent to predict the reactions of the environment to these patterns. GANs can be regarded as a case where the environmental reaction is 1 or 0 depending on whether the first network's output is in a given set. [108]

Other people had similar ideas but did not develop them similarly. An idea involving adversarial networks was published in a 2010 blog post by Olli Niemitalo. [109] This idea was never implemented and did not involve stochasticity in the generator and thus was not a generative model. It is now known as a conditional GAN or cGAN. [110] An idea similar to GANs was used to model animal behavior by Wei Li, Melvin Gauci and Roderich Gross in 2013. [111]

Another inspiration for GANs was noise-contrastive estimation, [112] which uses the same loss function as GANs and which Goodfellow studied during his PhD in 2010–2014.

Adversarial machine learning has other uses besides generative modeling and can be applied to models other than neural networks. In control theory, adversarial learning based on neural networks was used in 2006 to train robust controllers in a game theoretic sense, by alternating the iterations between a minimizer policy, the controller, and a maximizer policy, the disturbance. [113] [114]

In 2017, a GAN was used for image enhancement focusing on realistic textures rather than pixel-accuracy, producing a higher image quality at high magnification. [115] In 2017, the first faces were generated. [116] These were exhibited in February 2018 at the Grand Palais. [117] [118] Faces generated by StyleGAN [119] in 2019 drew comparisons with Deepfakes . [120] [121] [122]

See also

Artificial intelligence art – Visual media created with AI Pages displaying short descriptions of redirect targets

Deepfake – Realistic artificially generated media

Deep learning – Branch of machine learning

Diffusion model – Deep learning algorithm

Generative artificial intelligence – Subset of AI using generative models

Synthetic media – Artificial production of media by automated means

References

External links

Art portal

Knight, Will. "5 Big Predictions for Artificial Intelligence in 2017" . MIT Technology Review . Retrieved January 5, 2017 .

Karras, Tero; Laine, Samuli; Aila, Timo (2018). "A Style-Based Generator Architecture for Generative Adversarial Networks". arXiv : 1812.04948 [cs.NE].

This Person Does Not Exist – photorealistic images of people who do not exist, generated by StyleGAN

This Cat Does Not Exist Archived March 5, 2019, at the Wayback Machine – photorealistic images of cats who do not exist, generated by StyleGAN

Wang, Zhengwei; She, Qi; Ward, Tomas E. (2019). "Generative Adversarial Networks in Computer Vision: A Survey and Taxonomy". arXiv : 1906.01529 [cs.LG].

v

t

e

Autoencoder

Deep learning

Fine-tuning

Foundation model

Generative adversarial network

Generative pre-trained transformer

Large language model

Model Context Protocol

Neural network

Prompt engineering

Reinforcement learning from human feedback

Retrieval-augmented generation

Self-supervised learning

Stochastic parrot

Synthetic data

Top-p sampling

Transformer

Variational autoencoder

Vibe coding

Vision transformer

Waluigi effect

Word embedding

Character.ai

ChatGPT

DeepSeek

Ernie

Gemini

Grok

Copilot

Claude

Gemini

Gemma

GPT 1 2 3 J 4 4o 4.5 4.1 OSS 5

1

2

3

J

4

4o

4.5

4.1

OSS

5

Llama

o1

o3

o4-mini

Qwen

Base44

Claude Code

Cursor

Devstral

GitHub Copilot

Kimi-Dev

Qwen3-Coder

Replit

Xcode

Aurora

Firefly

Flux

GPT Image 1

Ideogram

Imagen

Midjourney

Qwen-Image

Recraft

Seedream

Stable Diffusion

Dream Machine

Hailuo AI

Kling

Midjourney Video

Runway Gen

Seedance

Sora

Veo

Wan

15.ai

Eleven

MiniMax Speech 2.5

WaveNet

Eleven Music

Endel

Lyria

Riffusion

Suno AI

Udio

Agentforce

AutoGLM

AutoGPT

ChatGPT Agent

Devin AI

Manus

OpenAI Codex

Operator

Replit Agent

01.AI

Aleph Alpha

Anthropic

Baichuan

Canva

Cognition AI

Cohere

Contextual AI

DeepSeek

ElevenLabs

Google DeepMind

HeyGen

Hugging Face

Inflection AI

Krikey AI

Kuaishou

Luma Labs

Meta AI

MiniMax

Mistral AI

Moonshot AI

OpenAI

Perplexity AI

Runway

Safe Superintelligence

Salesforce

Scale AI

SoundHound

Stability AI

Synthesia

Thinking Machines Lab

Upstage

xAI

Z.ai

Category

v

t

e

History timeline

timeline

Companies

Projects

Parameter Hyperparameter

Hyperparameter

Loss functions

Regression Bias–variance tradeoff Double descent Overfitting

Bias–variance tradeoff

Double descent

Overfitting

Clustering

Gradient descent SGD Quasi-Newton method Conjugate gradient method

SGD

Quasi-Newton method

Conjugate gradient method

Backpropagation

Attention

Convolution

Normalization Batchnorm

Batchnorm

Activation Softmax Sigmoid Rectifier

Softmax

Sigmoid

Rectifier

Gating

Weight initialization

Regularization

Datasets Augmentation

Augmentation

Prompt engineering

Reinforcement learning Q-learning SARSA Imitation Policy gradient

Q-learning

SARSA

Imitation

Policy gradient

Diffusion

Latent diffusion model

Autoregression

Adversary

RAG

Uncanny valley

RLHF

Self-supervised learning

Reflection

Recursive self-improvement

Hallucination

Word embedding

Vibe coding

Machine learning In-context learning

In-context learning

Artificial neural network Deep learning

Deep learning

Language model Large language model NMT

Large language model

NMT

Reasoning language model

Model Context Protocol

Intelligent agent

Artificial human companion

Humanity's Last Exam

Artificial general intelligence (AGI)

AlexNet

WaveNet

Human image synthesis

HWR

OCR

Computer vision

Speech synthesis 15.ai ElevenLabs

15.ai

ElevenLabs

Speech recognition Whisper

Whisper

Facial recognition

AlphaFold

Text-to-image models Aurora DALL-E Firefly Flux Ideogram Imagen Midjourney Recraft Stable Diffusion

Aurora

DALL-E

Firefly

Flux

Ideogram

Imagen

Midjourney

Recraft

Stable Diffusion

Text-to-video models Dream Machine Runway Gen Hailuo AI Kling Sora Veo

Dream Machine

Runway Gen

Hailuo AI

Kling

Sora

Veo

Music generation Riffusion Suno AI Udio

Riffusion

Suno AI

Udio

Word2vec

Seq2seq

GloVe

BERT

T5

Llama

Chinchilla AI

PaLM

GPT 1 2 3 J ChatGPT 4 4o o1 o3 4.5 4.1 o4-mini 5

1

2

3

J

ChatGPT

4

4o

o1

o3

4.5

4.1

o4-mini

5

Claude

Gemini Gemini (language model) Gemma

Gemini (language model)

Gemma

Grok

LaMDA

BLOOM

DBRX

Project Debater

IBM Watson

IBM Watsonx

Granite

PanGu-Σ

DeepSeek

Qwen

AlphaGo

AlphaZero

OpenAI Five

Self-driving car

MuZero

Action selection AutoGPT

AutoGPT

Robot control

Alan Turing

Warren Sturgis McCulloch

Walter Pitts

John von Neumann

Claude Shannon

Shun'ichi Amari

Kunihiko Fukushima

Takeo Kanade

Marvin Minsky

John McCarthy

Nathaniel Rochester

Allen Newell

Cliff Shaw

Herbert A. Simon

Oliver Selfridge

Frank Rosenblatt

Bernard Widrow

Joseph Weizenbaum

Seymour Papert

Seppo Linnainmaa

Paul Werbos

Geoffrey Hinton

John Hopfield

Jürgen Schmidhuber

Yann LeCun

Yoshua Bengio

Lotfi A. Zadeh

Stephen Grossberg

Alex Graves

James Goodnight

Andrew Ng

Fei-Fei Li

Alex Krizhevsky

Ilya Sutskever

Oriol Vinyals

Quoc V. Le

Ian Goodfellow

Demis Hassabis

David Silver

Andrej Karpathy

Ashish Vaswani

Noam Shazeer

Aidan Gomez

John Schulman

Mustafa Suleyman

Jan Leike

Daniel Kokotajlo

François Chollet

Neural Turing machine

Differentiable neural computer

Transformer Vision transformer (ViT)

Vision transformer (ViT)

Recurrent neural network (RNN)

Long short-term memory (LSTM)

Gated recurrent unit (GRU)

Echo state network

Multilayer perceptron (MLP)

Convolutional neural network (CNN)

Residual neural network (RNN)

Highway network

Mamba

Autoencoder

Variational autoencoder (VAE)

Generative adversarial network (GAN)

Graph neural network (GNN)

Category