# Introducing Generative Adversarial Network (GAN)

**Presentation on the module Advance Machine Learning**


3048781 Sanchit Bhavsar

3044093 Sthitadhee Panthadas


**Course: Advance Machine Learning**

**Prof. Dr. Ulf Brefeld**

LEUPHANA
UNIVERSITÄT LÜNEBURG

# Agenda

- **Understanding Generative Modeling**
- **Introducing GANs**
  - Model Architecture
  - Theoretical Foundation & Algorithm
  - Limitations of Vanila GAN
  - Wasserstein GAN
- **Timeline of GAN based Architecture**
  - Conditional GAN (CGAN)
  - Deep-Convolutional GAN (DCGAN)
  - GAN with different loss function (Least Square GAN)
- **Training GAN on MNIST Data**
- **Conclusion and Future Outlook**

# Quick Test: Which face is real?

A

B

C

# Supervised vs Unsupervised Learning

| Supervised Learning | Unsupervised Learning |
|---|---|
| **Data**: (x,y)<br>***x*** is data, ***y*** is label | **Data**: (x)<br>***x*** is data, ***no labels!*** |
| **Goal**: Learn function to map<br>$\boldsymbol{x} \longrightarrow \boldsymbol{y}$ | **Goal**: Learn the hidden or underlying structure of data |
| **Examples**: Classification, regression, Object detection etc. | **Examples**: Clustering, dimensionality reduction etc. |

# Generative Models

**Question**: can we build a model to approximate a data distribution?

- We are given and a finite sample from this distribution ,

$$X = \{x | x \sim p_{\text{data}}(x)\}, |X| = n$$

- **Goal**: Given training data, learn a model that represents new samples from same distribution

$$p_{\text{model}}(x; \theta) \approx p_{\text{data}}(x)$$

Training data ~ $p_{data}(x)$

Generated samples ~ $p_{model}(x)$

# Why Care About Generative Models?

Often overused quote:

“What I cannot create, I do not understand” -R. Feynman

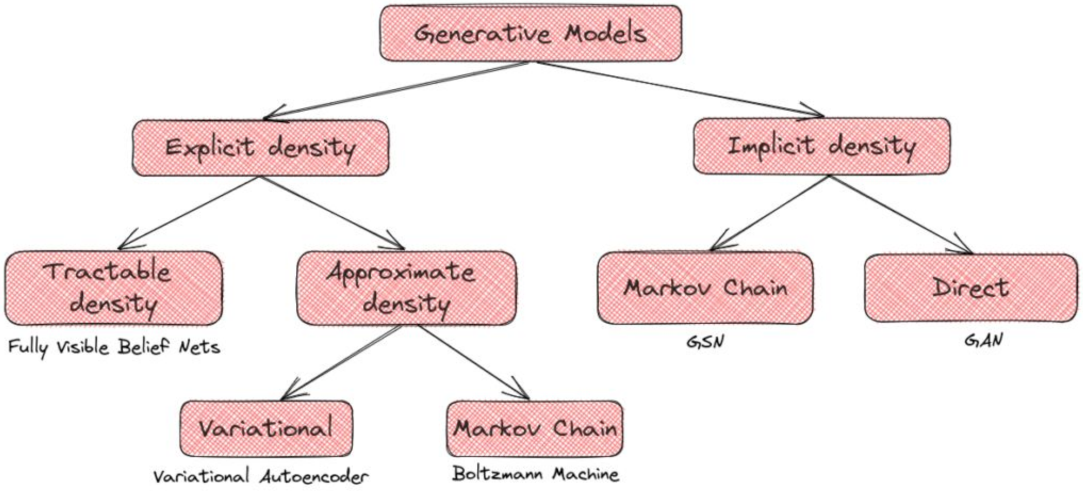
Capable of uncovering **underlying features** in a dataset,

- ❖ Noisy Input
- ❖ Simulated Data
- ❖ Features Representation of Data
- ❖ Prediction of Future State
- ❖ Missing Data


source: StyleGAN

# Taxonomy of Generative Models

- Addresses <u>density estimation</u>, a core problem in unsupervised learning
  - **Explicit density estimation**: explicitly define and solve for $p_{model}(x)$
  - **Implicit density estimation**: learn model that can sample from $p_{model}(x)$ w/o explicitly defining it





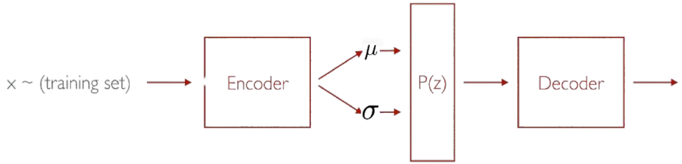

source: Taxonomy of Generative models. Adapted from Goodfellow

# WHAT ARE GANS?

Generative Adversarial Networks

## Generative Models

We try to learn the underlying the distribution from which our dataset comes from.
Eg: Variational AutoEncoders (VAE)

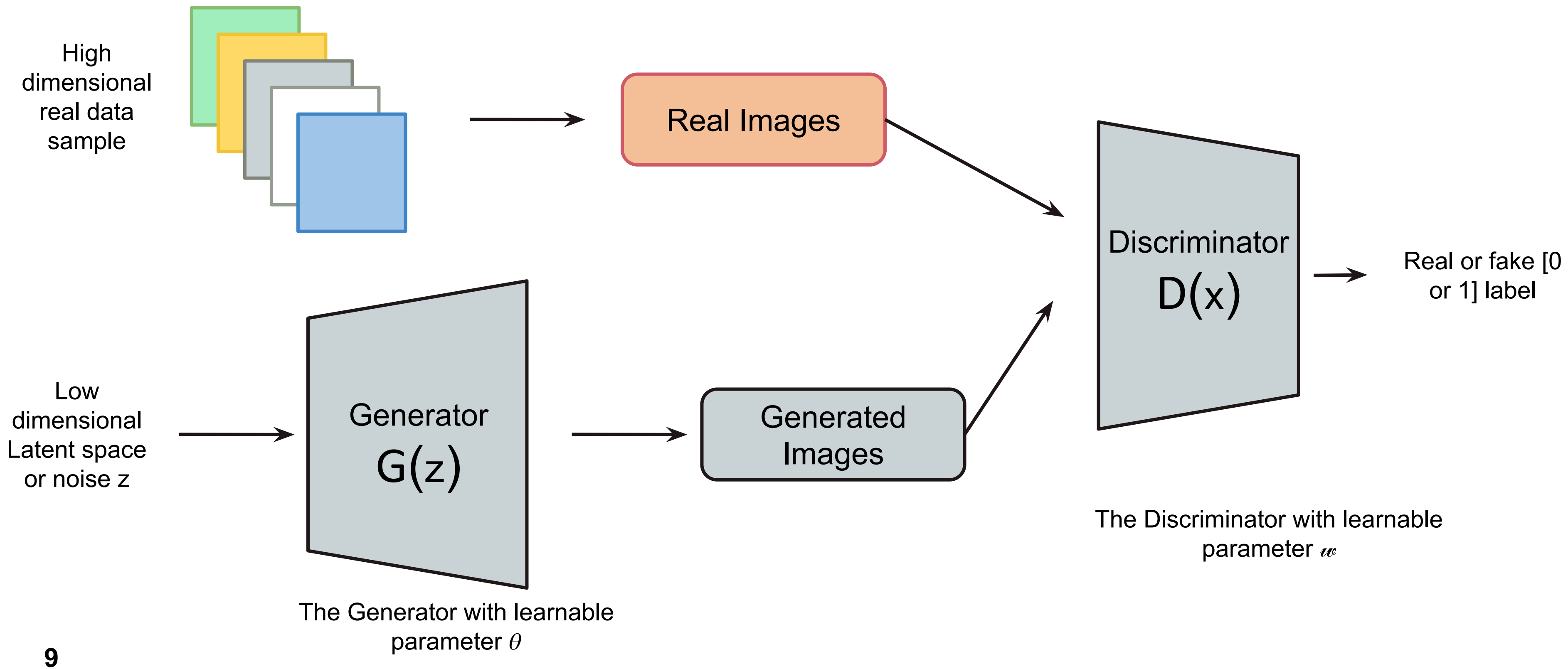



## Adversarial Training

GANS are made up of two competing networks (adversaries) that are trying beat each other.

## Neural Networks

# GAN Architecture: Two Player Game



# Objective Function

| Notation | Description |
|---|---|
| D(x) | The discriminator network |
| G(z) | The generator network |
| $z$ | STD Gaussian noise |
| $p$ | Probability distribution or labels |
| $q$ | Probability distribution or labels |
| H(p, q) | Binary Cross entropy |

$$\boxed{B.C.E = H(p,q) = -p \log q + (1-p)\log(1-q)}$$

i

$$Err(1, D(x)) + Err(0, D(G(z)))$$
$$= -[1 \cdot \log D(x) + 0 \cdot \log(1 - D(x)] - [0 \cdot \log D(G(z)) + 1 \cdot \log(1 - D(G(z))]$$
$$= -\log D(x) - \log(1 - D(G(z))$$

p = 1
q = D(.)

ii

$$\log D(x) + \log(1 - D(G(z))$$

Discriminator loss → must maximise

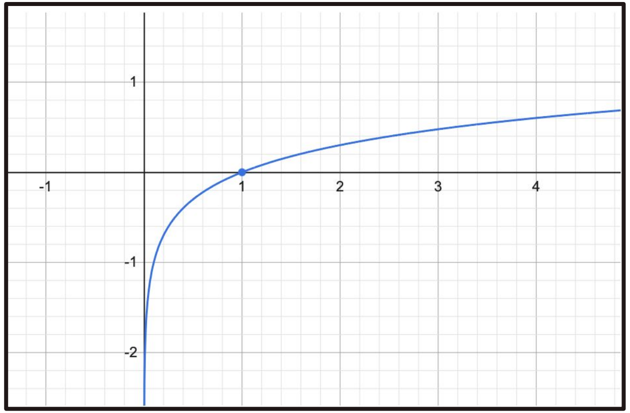
$$\log(1 - D(G(z))$$

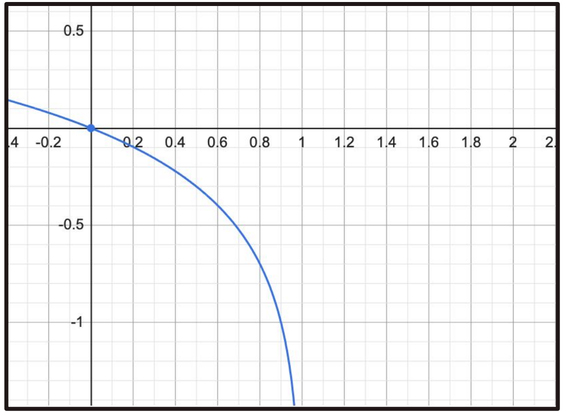
Generator loss → must minimise

# Objective Function

| Notation | Description |
|---|---|
| w | The discriminator parameters |
| $\theta$ | The generator parameters |
| $E$ | Expectation with respect to distribution |
| $x$ | Real data sample |

$$\min_{\theta} \max_{w} \mathbb{E}_{x \sim p_{data}}[\log D_w(x)] + \mathbb{E}_{z \sim p_{noise}}[\log(1 - D_w(G_\theta(z)))]$$



# GAN Algorithm

**Algorithm 1** Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, $k$, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

**for** number of training iterations **do**

  **for** $k$ steps **do**

  - Sample minibatch of $m$ noise samples $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Sample minibatch of $m$ examples $\{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(m)}\}$ 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 $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ 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.

# Global Optimality



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

Training criterion for D (discriminator) at a fixed G

$$= \int_{\boldsymbol{x}} \underbrace{p_{\text{data}}(\boldsymbol{x})}_{a} \log(\underbrace{D(\boldsymbol{x})}_{y}) + \underbrace{p_g(\boldsymbol{x})}_{b} \log(1 - \underbrace{D(\boldsymbol{x})}_{y})dx$$

Can be proven in 2 ways

$$y \to a\log(y) + b\log(1-y) \xrightarrow{\text{differentiate}} \frac{a}{a+b}$$

Note $(a, b) \in R^2 \setminus \{0, 0\}$ and $y$ achieves its maximum in the range [0,1]

$$D^*_G(\boldsymbol{x}) = \frac{p_{data}(\boldsymbol{x})}{p_{data}(\boldsymbol{x}) + p_g(\boldsymbol{x})} = 0.5$$

- Global Optimality is reached at $p_g = p_{data}$
- At that point $D^*_G(x) = 0.5$

| Notation | Description |
|---|---|
| $p_{data}$ | Prob Distribution of real data space |
| $p_g$ | Prob Distribution of generated data |
| $D^*(x)$ | Optimal discriminator at fixed G |
| $p_z$ | Prob distribution of std normal |
| V(G, D) | General Loss function of GAN |

# Connection to Jensen-Shannon Divergence

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

Can be proved via →

- Change of Variables
- Radon-Nikodym theorem (in Measure Theory)

| Notation | Description |
|---|---|
| D*(x) | Optimal discriminator at fixed G |
| C(G) | Training Criterion at fixed G |
| V(G, D) | General Loss function of GAN |

- Global Optimality is reached at $p_g = p_{data}$
- At that point $D^*_G(x) = 0.5$

# Connection to Jensen-Shannon Divergence

- Global Optimality is reached at $p_g = p_{data}$
- At that point $D^*_G(x) = 0.5$

$$\mathbb{E}_{x\sim p(x)}[f(x)] = \sum p(x)f(x)$$

$$C(G) = \mathbb{E}_{x\sim p_{data}}\left[\log \frac{p_{data}}{p_{data}+p_G}\right] + \mathbb{E}_{x\sim p_G}\left[\log \frac{p_G}{p_{data}+p_G}\right]$$

$$= \mathbb{E}_{x\sim p_{data}}\left[\log \frac{\frac{p_{data}}{2}}{\frac{p_{data}+p_G}{2}}\right] + \mathbb{E}_{x\sim p_G}\left[\log \frac{\frac{p_G}{2}}{\frac{p_{data}+p_G}{2}}\right]$$

$$= -\log 4 + \mathbb{E}_{x\sim p_{data}}\left[\log \frac{p_{data}}{\frac{p_{data}+p_G}{2}}\right] + \mathbb{E}_{x\sim p_G}\left[\log \frac{p_G}{\frac{p_{data}+p_G}{2}}\right]$$

$$= -\log 4 + KL\left(p_{data}\middle\|\frac{p_{data}+p_G}{2}\right) + KL\left(p_G\middle\|\frac{p_{data}+p_G}{2}\right)$$

$$= -\log 4 + 2JS(p_{data}\|p_G)$$

$p_g = p_{data}$ then $D^*_G(x) = 0.5$ $\longrightarrow$ $-\log 4$ or $-2\log 2$

| Notation | Description |
|---|---|
| C(G) | Virtual Training Criterion for fixed G. |
| $p_G$ | Prob Distribution of generated data |
| *KL* | Kullback-Liebler divergence |
| *JS* | Jensen Shannon Divergence |
| $D^*_G(x)$ | Optimal discriminator |
| f(x) | A general function, here log(.) |

# Convergence of Algorithm

The argument follows that →

★ Due to linearity of expectations

★ Due to each term being convex functions in $p_g$

Intuitive explanation of the terms + the proof and concept of sampling from $p_g$

Sufficient small Gradient descent updates for $p_g$ having a unique global optima D*, allows $p_g$ to converge to $p_{data}$.

**In practice,**

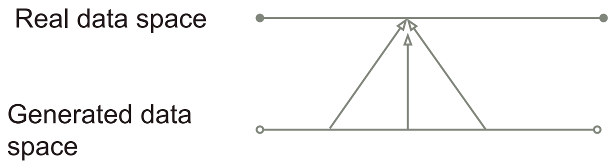
❖ $p_g$ → **neural network** of the generating resulting in generated samples,

❖ we optimize not $p_g$ but the parameters of the generator neural network $\theta_g$.

# Limitations of Vanilla GAN

**In practice,**

- ❖ Models do not converge to a local minima
- ❖ Mode collapse problem




**Because of,**

- ❖ Most real life cases functions are necessarily convex and continuous and therefore not differentiable.
- ❖ Vanishing or exploding gradients

# Motivating Wasserstein

$$\delta(\mathbb{P}_r, \mathbb{P}_g) = \sup_{A \in \Sigma} |\mathbb{P}_r(A) - \mathbb{P}_g(A)| \ .$$

$$KL(\mathbb{P}_r \| \mathbb{P}_g) = \int \log\left(\frac{P_r(x)}{P_g(x)}\right) P_r(x)$$

$$JS(\mathbb{P}_r, \mathbb{P}_g) = KL(\mathbb{P}_r \| \mathbb{P}_m) + KL(\mathbb{P}_g \| \mathbb{P}_m)$$

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma}\left[\ \|x - y\|\ \right]$$

| Notation | Description |
|---|---|
| A | Set with which you calculate the prob example $P_r(A)$ |
| Σ | Total countable set space |
| $P_r$ | Prob distribution for real data samples |
| $P_g$ | Probability distribution representing generated data |
| γ | Scalar transport mass values/energies |
| $\mathbf{\Pi}(P_r, P_g)$ | Joint distribution of the transport mass value |
| inf | Infimum |
| \|\| . \|\| | Norm |
| sup | Supremum |
| $\delta$ ( $P_r$, $P_g$ ) | Total Variation (TV) distance |
| $W$ ( $P_r$, $P_g$ ) | 1-Wasserstein or Earth Mover distance |

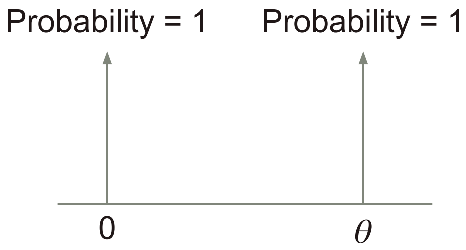
# Motivating Wasserstein

$$\delta(\mathbb{P}_r, \mathbb{P}_g) = \sup_{A \in \Sigma} |\mathbb{P}_r(A) - \mathbb{P}_g(A)| \ .$$

$$KL(\mathbb{P}_r \| \mathbb{P}_g) = \int \log \left( \frac{P_r(x)}{P_g(x)} \right) P_r(x)$$

$$JS(\mathbb{P}_r, \mathbb{P}_g) = KL(\mathbb{P}_r \| \mathbb{P}_m) + KL(\mathbb{P}_g \| \mathbb{P}_m)$$

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y) \sim \gamma} \big[ \, \|x - y\| \, \big]$$




$$W(\mathbb{P}_0, \mathbb{P}_\theta) = |\theta|,$$

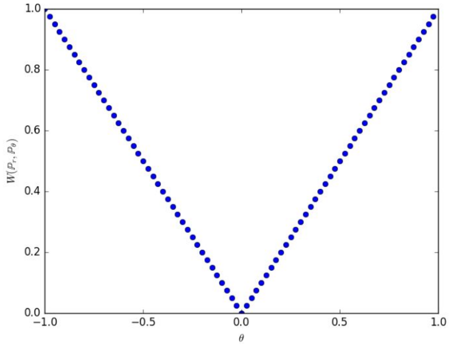
$$JS(\mathbb{P}_0, \mathbb{P}_\theta) = \begin{cases} \log 2 & \text{if } \theta \neq 0 \ , \\ 0 & \text{if } \theta = 0 \ , \end{cases}$$

$$KL(\mathbb{P}_\theta \| \mathbb{P}_0) = KL(\mathbb{P}_0 \| \mathbb{P}_\theta) = \begin{cases} +\infty & \text{if } \theta \neq 0 \ , \\ 0 & \text{if } \theta = 0 \ , \end{cases}$$

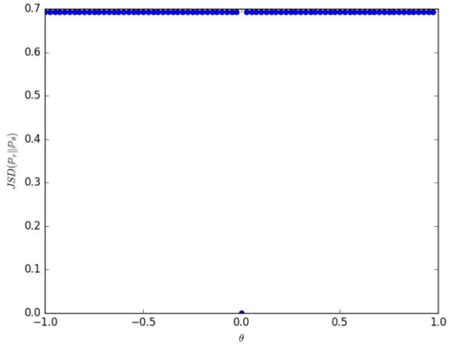
$$\text{and } \delta(\mathbb{P}_0, \mathbb{P}_\theta) = \begin{cases} 1 & \text{if } \theta \neq 0 \ , \\ 0 & \text{if } \theta = 0 \ . \end{cases}$$

# Motivating Wasserstein

- Wasserstein distance provides a **proportional value**, the further the distribution the greater the distance. In practical cases the generating sample distribution will **not overlap** the real data sample, wasserstein distance will be more practical than the other **metric measures** such as KL divergence or JSD or TV.
- Furthermore, Wasserstein distance or Earth mover distance is continuous and smoother function compared to the others.




EM distance




JSD

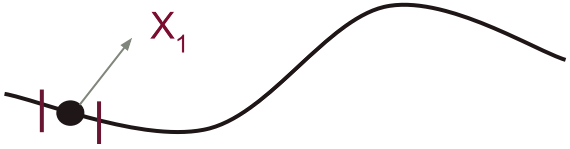
# Wasserstein and Continuity

What do you need for continuity for?

- **Smooth** gradient descent or ascent

What do we need for smooth gradient descent or ascent?

- Neural network G must be continuous in parameter $\theta \rightarrow$ which is the case in EM distance
- G must be locally Lipschitz




The slope of the curve at point $X_n$ should be less than equal to K. (K-Lipschitz)

# Wasserstein Loss

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma}\big[\, \|x - y\| \,\big]$$

- Note, for most practical probability distributions **EM distance calculation is intractable** because for that we would be **needing the joint probability $\Pi(P_r, P_g)$**.
- So, we use **Kantorovich-Rubinstein duality** to get us a new formulae for EM distance that works.

$$W(\mathbb{P}_r, \mathbb{P}_\theta) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x\sim\mathbb{P}_r}[f(x)] - \mathbb{E}_{x\sim\mathbb{P}_\theta}[f(x)]$$

- Note here the **supremum** is over **1-Lipschitz functions**
- But this can be easily changed to **k-Lipschitz functions** with $K \cdot W(\mathbb{P}_r, \mathbb{P}_g)$

| Notation | Description |
|---|---|
| sup | Supremum |
| f(x) | K-Lipschitz function |
| $\|\| f \|\|_{L\leq 1}$ | 1-Lipschitz function(s) |
| $P_\theta$ | Prob distribution representing generated data |
| $q$ | Probability distribution |
| K | Constant value for Lipschitz constraint |

# Wasserstein Loss

## Vanilla GAN Objective function

$$\min_{\theta} \max_{w} \mathbb{E}_{x\sim p_{data}}[\log D_w(x)] + \mathbb{E}_{z\sim p_{noise}}[\log(1 - D_w(G_\theta(z))]$$

## Wasserstein GAN

$$\min_{\theta} \max_{w} V(G, C) = \min_{\theta} \max_{w} \ [\ \mathbb{E}_{x\sim p_{data}}[C_w(x)] - \mathbb{E}_{z\sim p_{noise}}[(C_w(G_\theta(z))]]$$

Where $C_w$ should follow the K-Lipschitz constraint.

# Weight Clipping

How do we ensure or **enforce Lipschitz constraint**?

- **Gradient/weight clipping** or gradient penalty

❖ **Solution:**

➢ **Weight clipping** is basically to cut off the value of the weights if it goes beyond a **threshold value** and then use the threshold value instead. This has a parameter that controls the amount of clipping. [ $-c \leq$ weight $\leq c$ ]

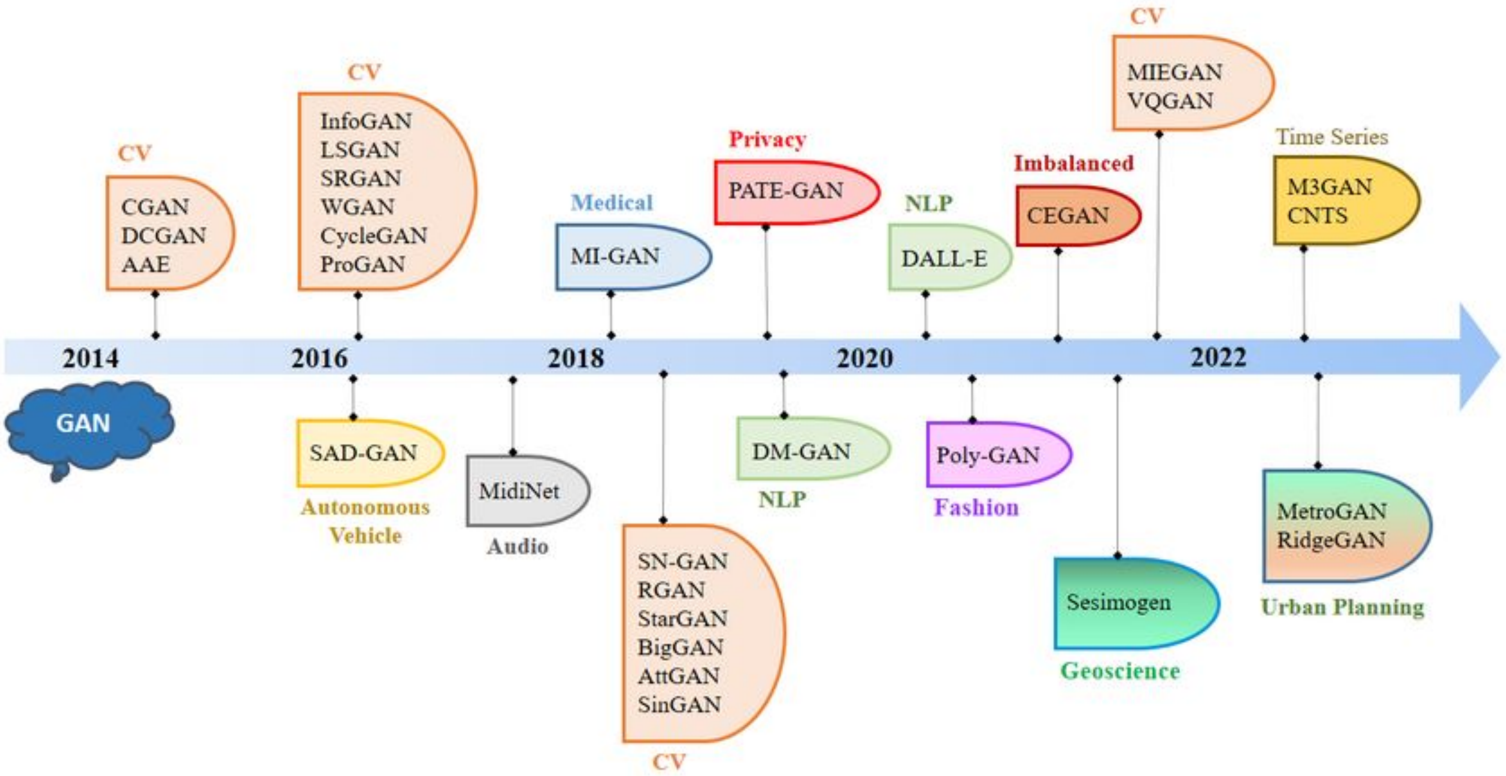
❖ **Problem**:

➢ This poses another problem. This reduces the optimality of the critic.

# WGAN vs Vanilla GAN

| WGAN | Vanilla GAN |
|---|---|
| • Discriminator acts as a critic and gives out real values (regression) | • Discriminator gives out 0 or 1 (classification) |
| • Wasserstein motivated loss | • JSD motivated loss |
| • Loss contains no log | • Loss contains log |
| • Discriminator is updated 5 times for each update in generator | • Discriminator is updated once for each update in generator |
| • Root Mean Squared Propagation is used | • Adam is used |
| • Includes clipping of parameter values | • Includes no weight clipping |

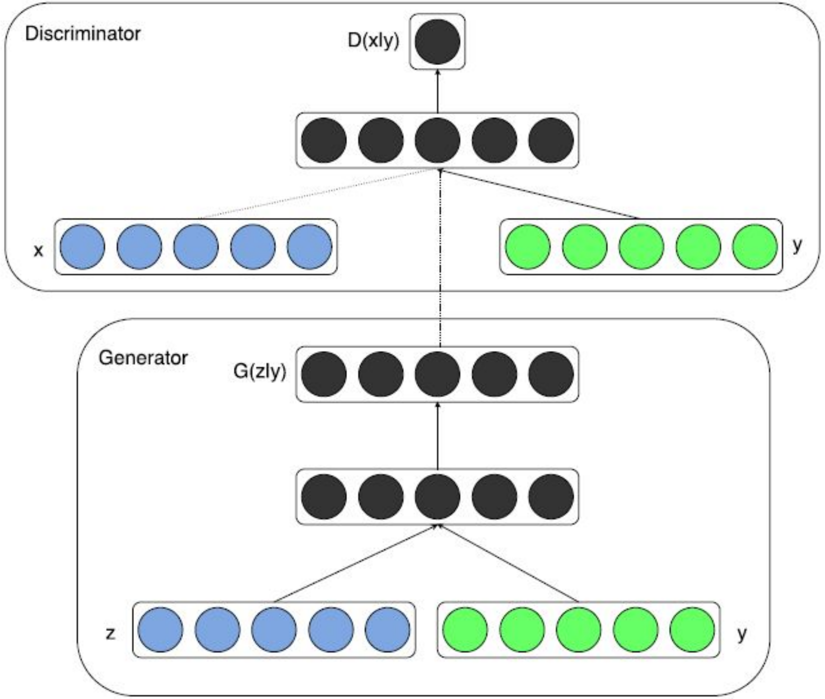
# Timeline of GAN based architecture




source: Ten Years of Generative Adversarial Nets (GANs)

# Conditional GAN (CGAN)

- ❖ In an unconditioned generative model, there is no control on modes of the data being generated
- ❖ In CGAN, the generator learns to generate a fake sample with a specific condition (such as a label associated with an image or more detailed tag) rather than a generic sample from unknown noise distribution
- ❖ Simply feeding the data **'y'** and conditioning both the generator and discriminator




Learning Loss of CGAN:

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



source: Conditional Generative Adversarial Nets

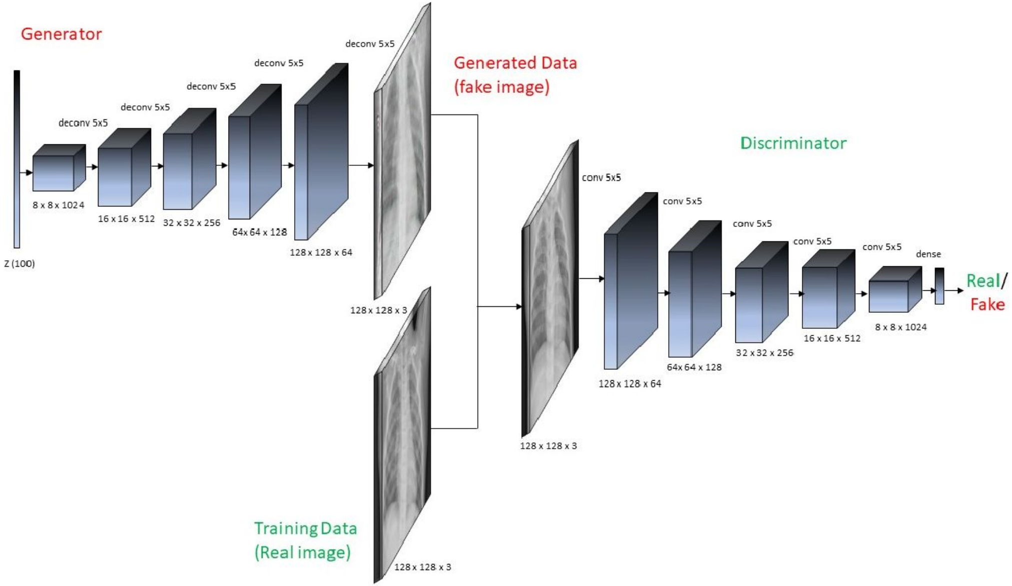
# Generated MNIST digits, each row conditioned on one label



source: Conditional Generative Adversarial Nets

# Deep Convolutional GAN (DCGAN)

- ❖ Generator is an upsampling network with fractionally-strided convolutions
- ❖ Discriminator is a convolutional network
- ❖ Eliminating fully connected layers on top of Convolutional Layers (instead of max polling)
- ❖ Batch Normalization
- ❖ Leaky ReLUs



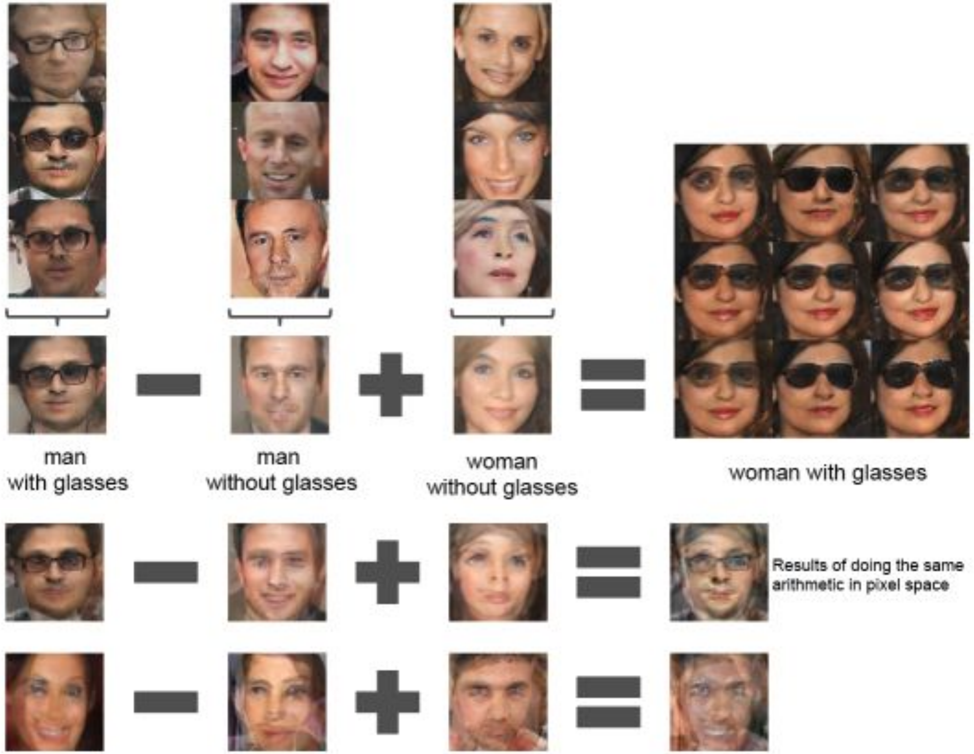



Source: Evaluation of Deep Convolutional of Chest X-ray Images

# Vector Arithmetic For Visual Concepts

Samples from the model

Average Z vectors, do arithmetic

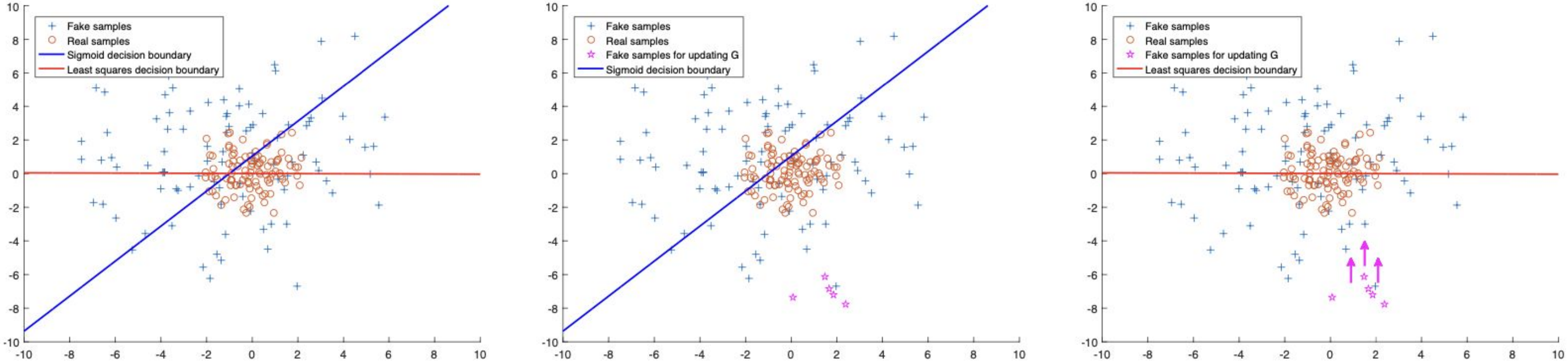





Source: Unsupervised representation learning with deep convolutional generative adversarial networks

# Least Square GAN (LSGAN)

**Idea**: Use loss function that provides smooth and non-saturation gradient in Discriminator



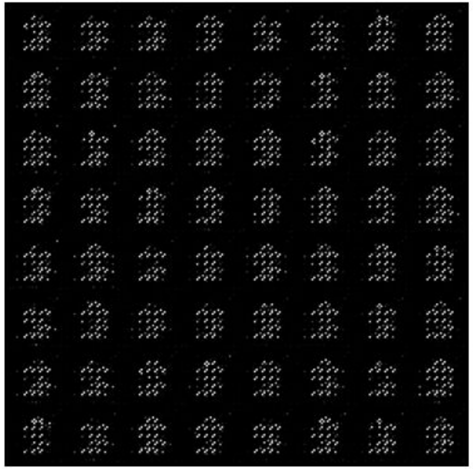

(a) (b) (c)

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



Source: Least Squares Generative Adversarial Networksy Images

# Training a GAN: MNIST Data



Epoch 1

Epoch 5

Epoch 10

Colab Link

# Conclusion

GANS,

- ❖ Don’t work with an explicit density function
- ❖ Game-theoretic approach: learn to generate from training distribution through two-player game

Pros:

- ❖ Beautiful, state-of-the-art samples!

Cons:

- ❖ Trickier / more unstable to train

Active areas of research:

- ❖ Better loss functions, more stable training (Wasserstein GAN, LSGAN, many others)
- ❖ Conditional GANs, GANs for all kinds of applications

# Explosion of GANs

LSGAN, Zhu 2017.

Wasserstein GAN, Arjovsky 2017.
Improved Wasserstein GAN, Gulrajani 2017.

Progressive GAN, Karras 2018.