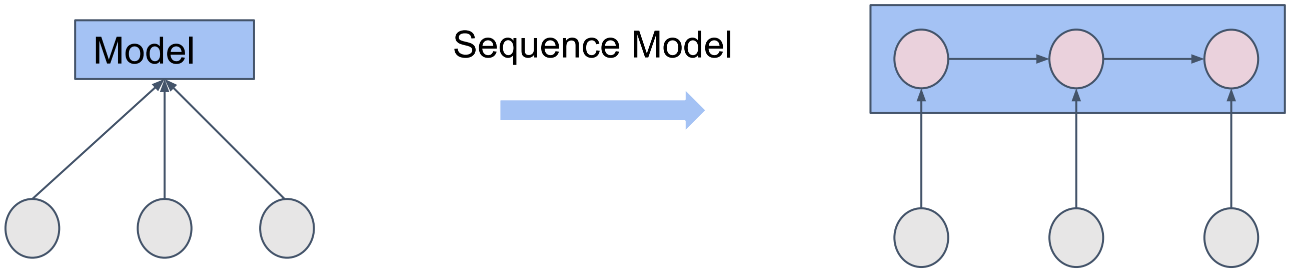
# RNN and Generative Deep Learning

# Agenda

1. Recurrent Neural Network
2. Generative Models
3. Generative Adversarial Network

# Recurrent Neural Network

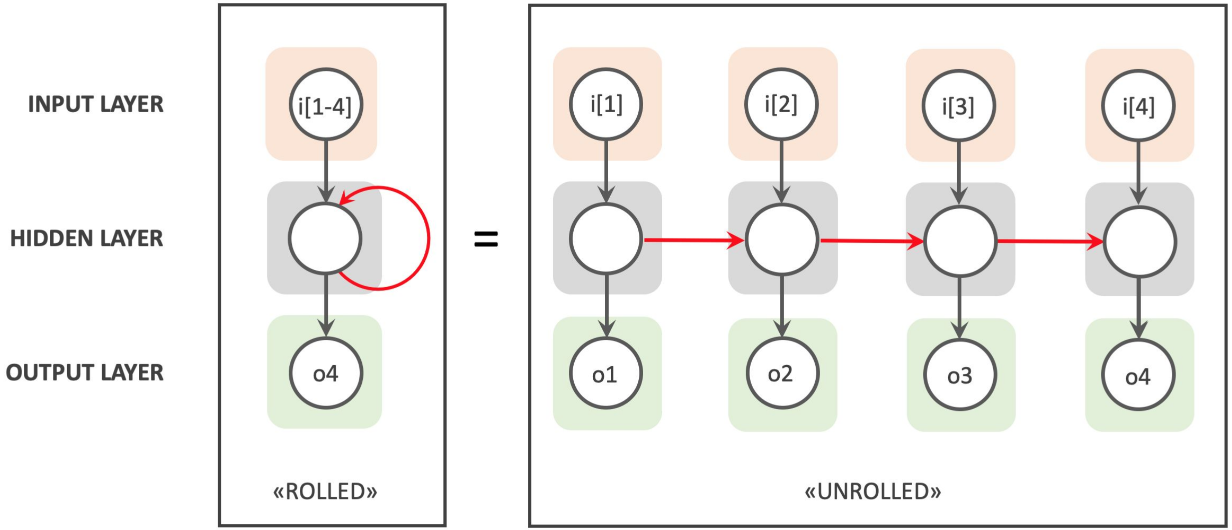
# Sequential Data




Machine learning models should capture this kind of order information in **sequential data (time-series and NLP data)**

# Recurrent Neural Network (Elman 1990)

- Recurrent neural network is proposed to utilize information from **previous** time steps and **current** information to make reasoning at the current step




https://www.bouvet.no/bouvet-deler/explaining-recurrent-neural-networks

# Neuron computation of RNN

- Model parameters are tied across all time steps (run the **same** RNN cell)

Hidden layers vector **at time step t**

Input vector **at time step t**

$$\mathbf{h}_t = \sigma(\mathbf{W}^{hh}\mathbf{h}_{t-1} + \mathbf{W}^{hx}\mathbf{x}_t)$$

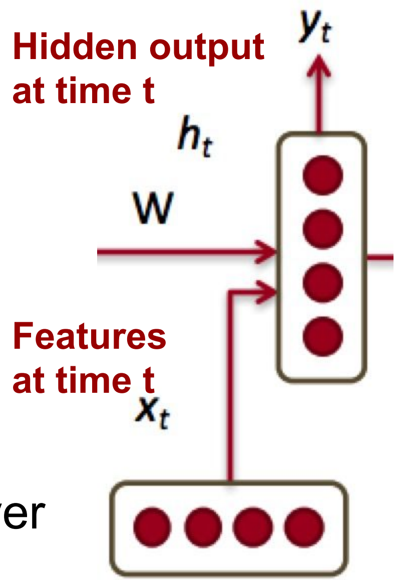
**Time-Invariant** Model Parameters

Predictions at time step t

$$\tilde{y}_t = softmax(\mathbf{W}^s\mathbf{h}_t)$$

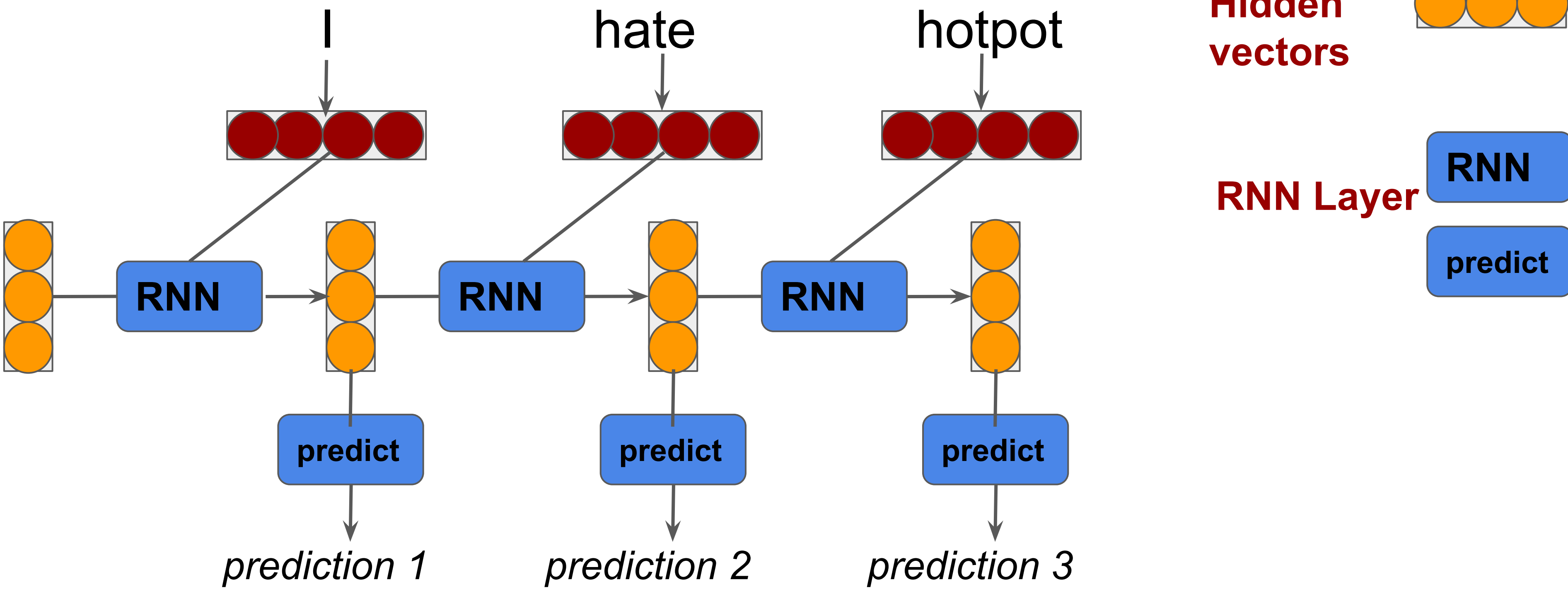
- We need $h_0 \in \mathbb{R}^{D_h}$ as the initialization vector for the hidden layer at time step **0**.

- Inputs enter and move forward at each time step



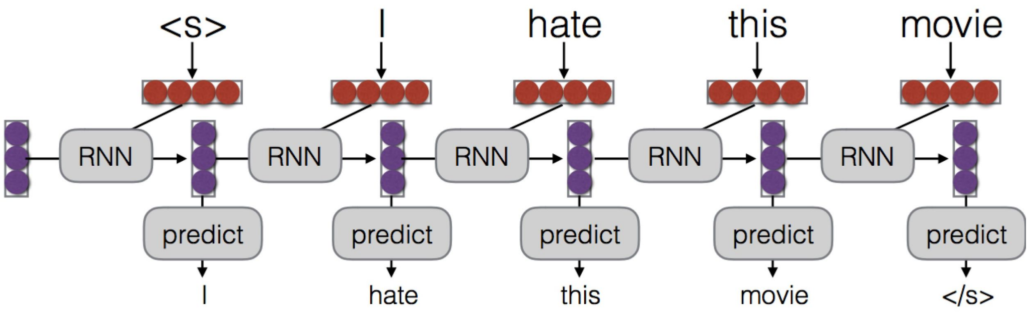

*Focus on certain time step*

# Sequence Tagging

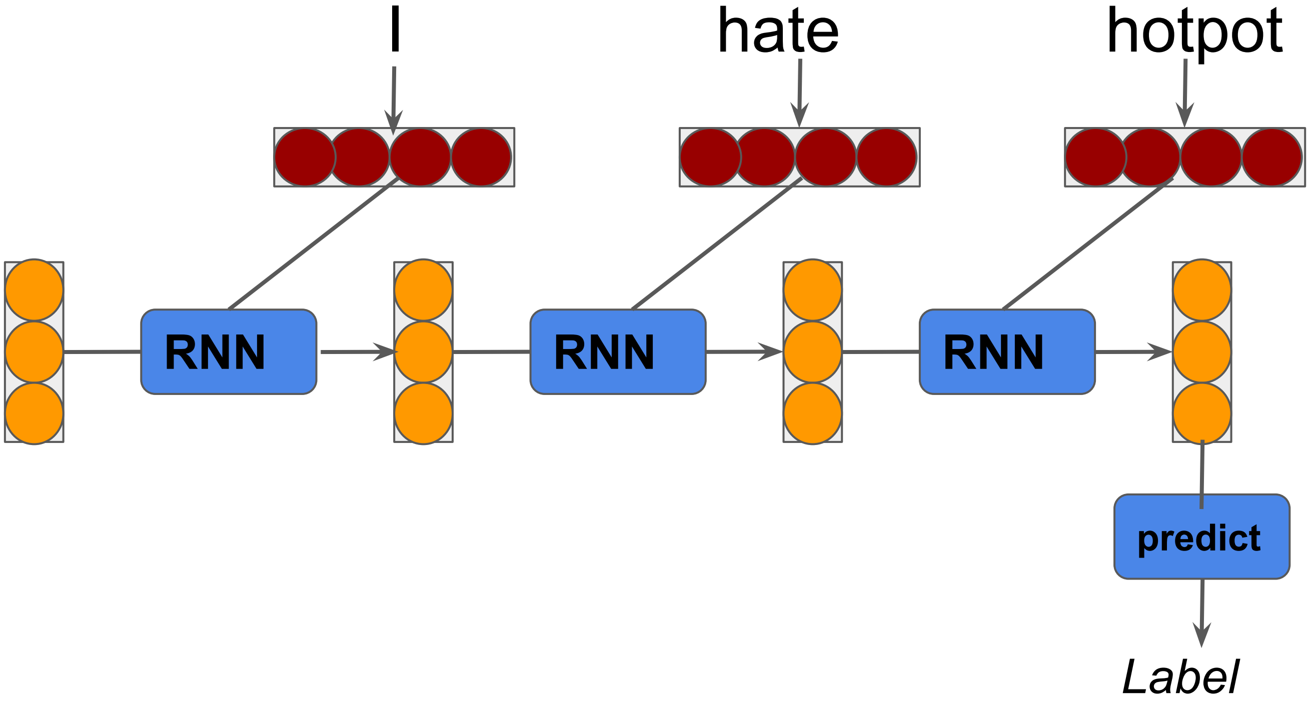


# RNN-based Language Model

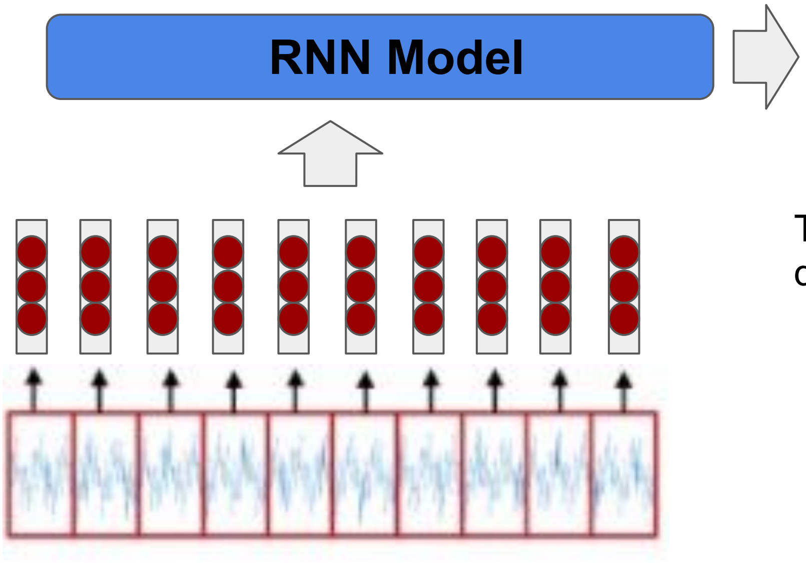
- A language model computes **a probability for a sequence of words**:
    - P(w1,..., wT)
    - Useful for machine translation, Chatbot and Question Answer Systems.
- Language Modelling can be formulated as a tagging problem
- Each label/tag is the next word!



# Sequence Classification



# RNN-based Time Series Prediction




**Discrete/Continuous Labels**

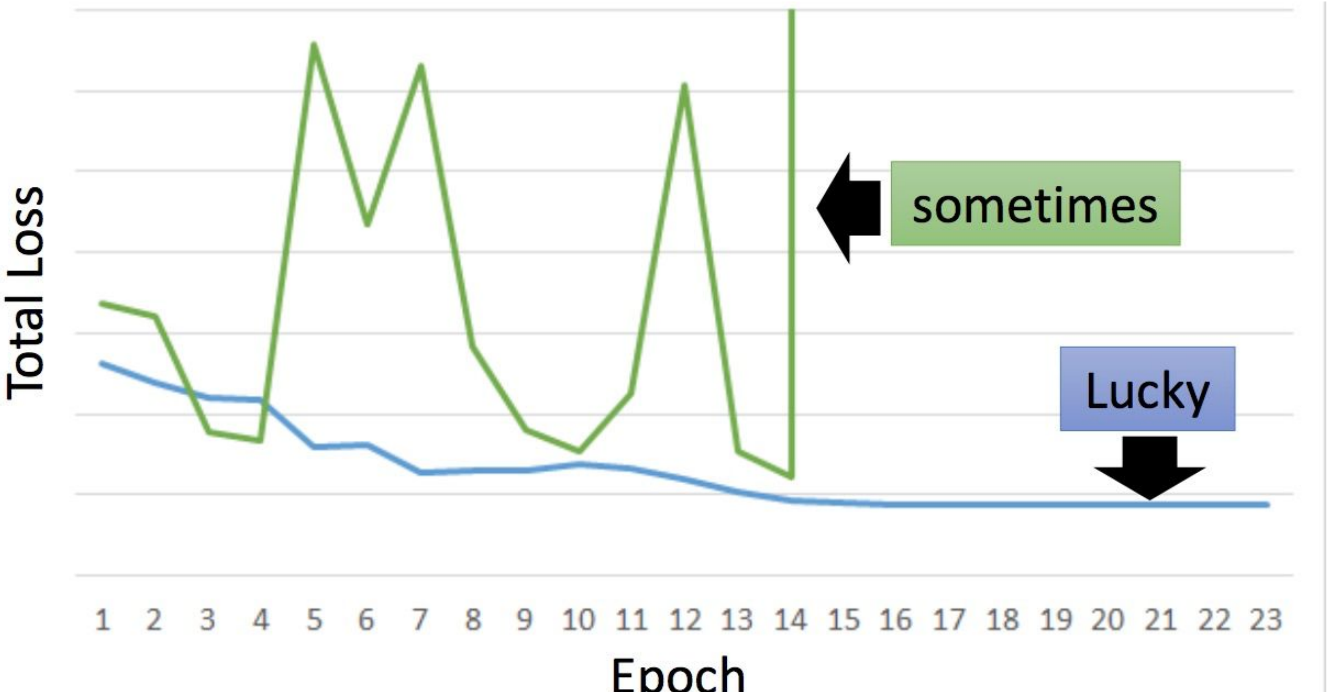
**Feature Extraction/ Downsampling**

Time, Frequency and Time-frequency domain Analysis

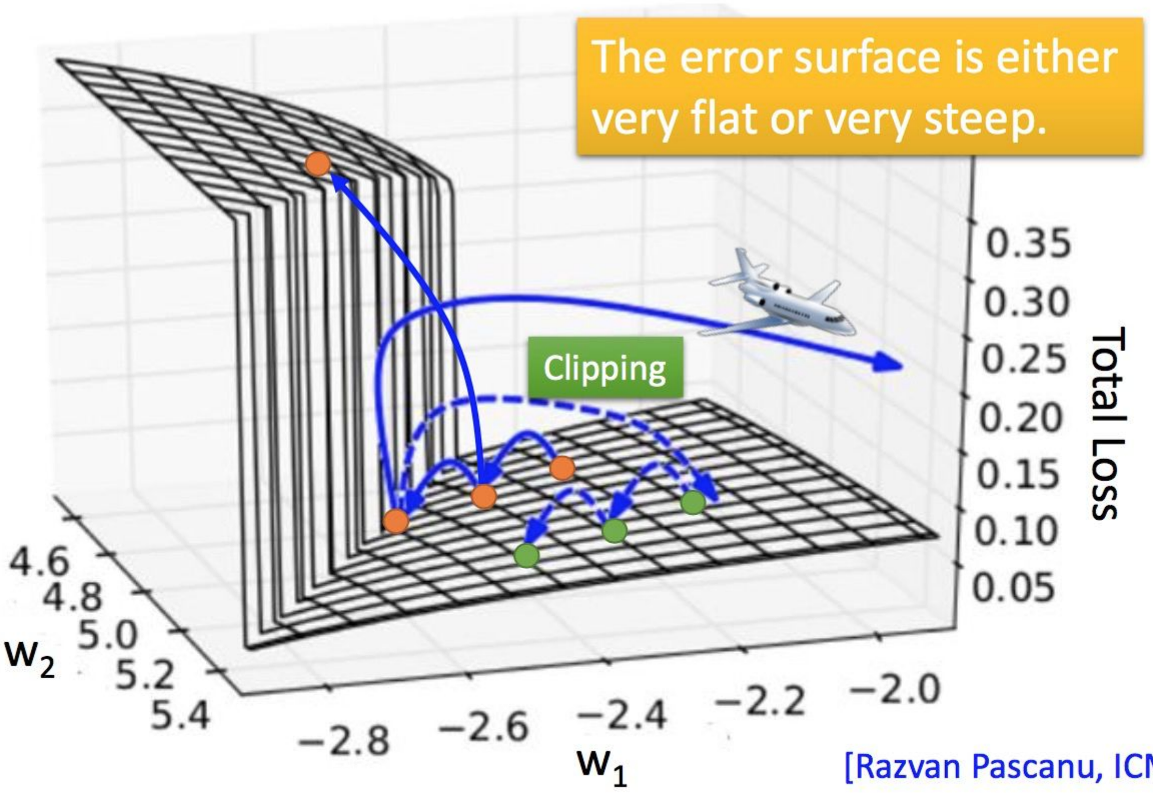
| Quantity | Equations |
|---|---|
| RMS | $z_{rms} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} z_i^2}$ |
| Variance | $\frac{1}{n} \sum_{i=1}^{n} (z_i - \bar{z})^2$ |
| Maximum | $max(z)$ |
| Skewness | $E[(\frac{z-\mu}{\sigma})^3]$ |
| Kurtosis | $E[(\frac{z-\mu}{\sigma})^4]$ |
| Peak-to-Peak | $max(z) - min(z)$ |
| Spectral Skewness | $\sum_{i=1}^{k} (\frac{f_i - \bar{f}}{\sigma})^3 S(f_i)$ |
| Spectral Kurtosis | $\sum_{i=1}^{k} (\frac{f_i - \bar{f}}{\sigma})^4 S(f_i)$ |
| Spectral Power | $\sum_{i=1}^{k} (f_i)^3 S(f_i)$ |
| Wavelet Energy | $\sum_{i=1}^{N} wt_{\phi}^2(i)/N$ |

# RNN Training is Hard

- Real experiments on Language Models



# Rough Error Surface of RNN




**Exploding/Vanishing Gradient**

[Razvan Pascanu, ICML'13]

# Toy Example

$w = 1 \Rightarrow y_{1000} = 1$

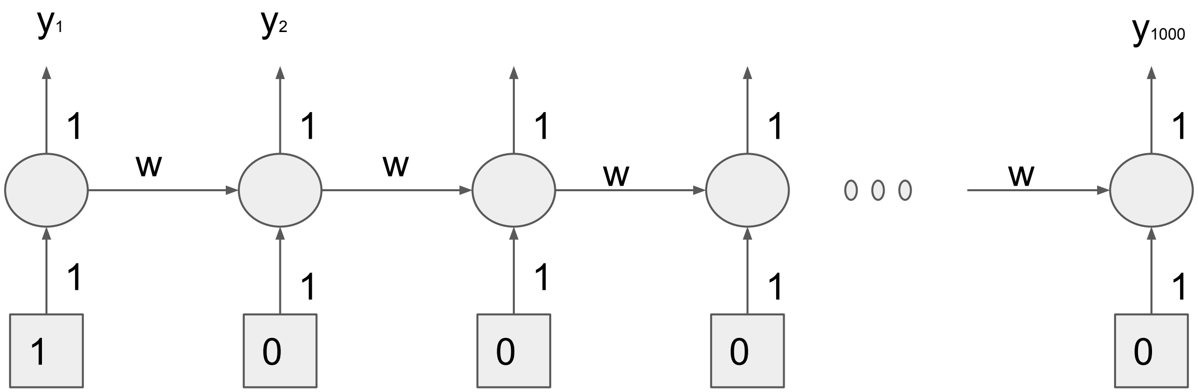
$w = 1.01 \Rightarrow y_{1000} = 20000$

$\Rightarrow$ Large Gradient: dL/dw $\Rightarrow$ **Irregular Error Surface**

$w = 0.99 \Rightarrow y_{1000} \sim= 0$

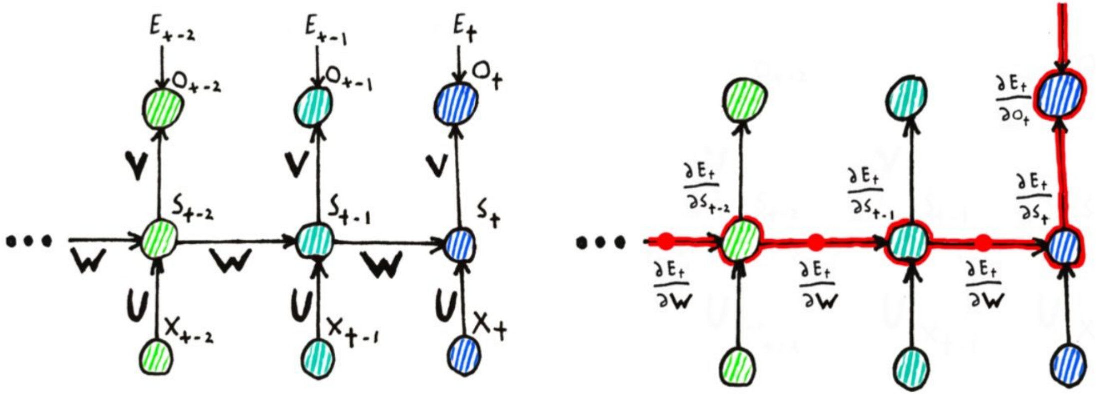
$w = 0.01 \Rightarrow y_{1000} \sim= 0$

$\Rightarrow$ Small Gradient: dL/dw $\Rightarrow$ **Irregular Error Surface**




no bias,
linear activation

# Backpropagation Through Time




$$\frac{\partial E_t}{\partial \mathbf{W}} = \sum_{k=0}^{t} \frac{\partial E_t}{\partial \mathbf{o_t}} \frac{\partial \mathbf{o_t}}{\partial \mathbf{s_t}} \boxed{\frac{\partial \mathbf{s_t}}{\partial \mathbf{s_k}}} \frac{\partial \mathbf{s_k}}{\partial \mathbf{W}}$$

Chain rule => Multiplications

Can explode or vanish

$$\prod_{j=k+1}^{t} \frac{\partial \mathbf{s_j}}{\partial \mathbf{s_{j-1}}}$$

# Exploding Gradient Solutions

- Truncated BPTT
  - Do not take the derivative all the way back to the beginning of the input sequence

$$\frac{\partial E_t}{\partial \mathbf{W}} = \sum_{\boxed{k=t-T}}^{t} \frac{\partial E_t}{\partial \mathbf{o_t}} \frac{\partial \mathbf{o_t}}{\partial \mathbf{s_t}} \frac{\partial \mathbf{s_t}}{\partial \mathbf{s_k}} \frac{\partial \mathbf{s_k}}{\partial \mathbf{W}}$$

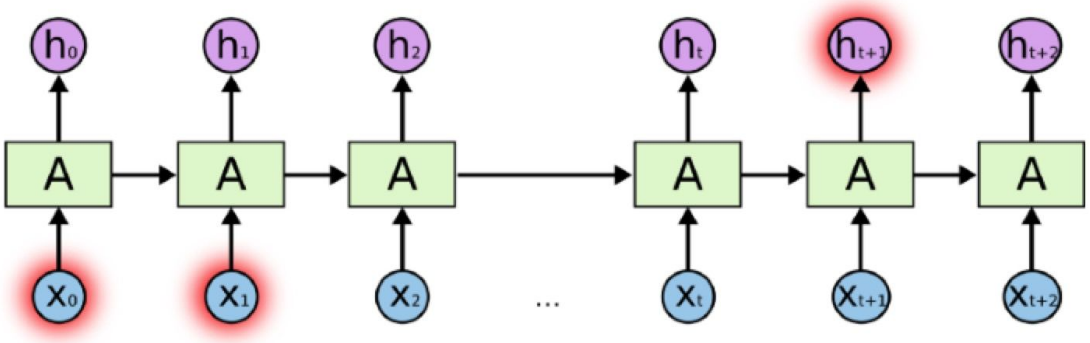
**Only through T time steps if t>= T**

- **Clip gradients at threshold**

- RMSprop to adjust learning rate
  - Adapt learning rate by dividing by the root of squared gradient

# Vanishing Gradient Problem

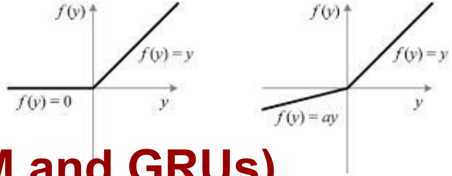
- The error at a time step **ideally** can tell a previous time step from many steps away to change during backprop
- **Can not capture long-term dependency**
- The representation from time steps 0 and t can not travel to influence the time step t+1
- **Harder to detect**



# Vanishing Gradient Solutions

- RMSprop
  - Adapt learning rate by dividing by the root of squared gradient
- Advanced activation functions such as leakyRelu function

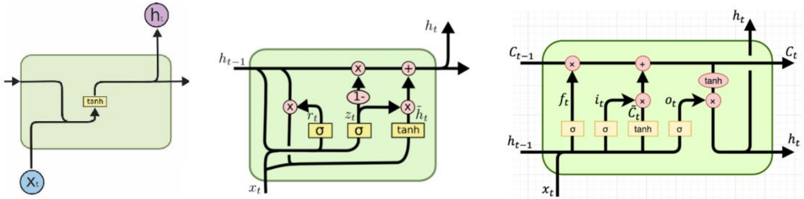
**ReLu**




Leaky ReLu

- **Gated RNN (LSTM and GRUs)**
  - Using gates in cell computation to control information flow

**Partially Solved**




RNN GRU LSTM

# RNN’s Bottleneck

- RNN is not suitable for parallel computation.

- RNN’s training is not easy
    - Gradient Vanishing
    - Gradient Exploding

# Generative Models

# Generative Models

- Given training data, generate new samples from same distribution

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


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

- Want to learn $p_{model}(x)$ similar to $p_{data}(x)$

- Address density estimation, a core problem in ? learning

# Generative Models

- Realistic samples for artwork, super-resolution, colorization, etc




- Generate data samples that can be used for simulation and planning (reinforcement learning applications)


Source Video
Detected Pose
Source to Target 1 Result
Source to Target 2 Result

# Taxonomy of Generative Models

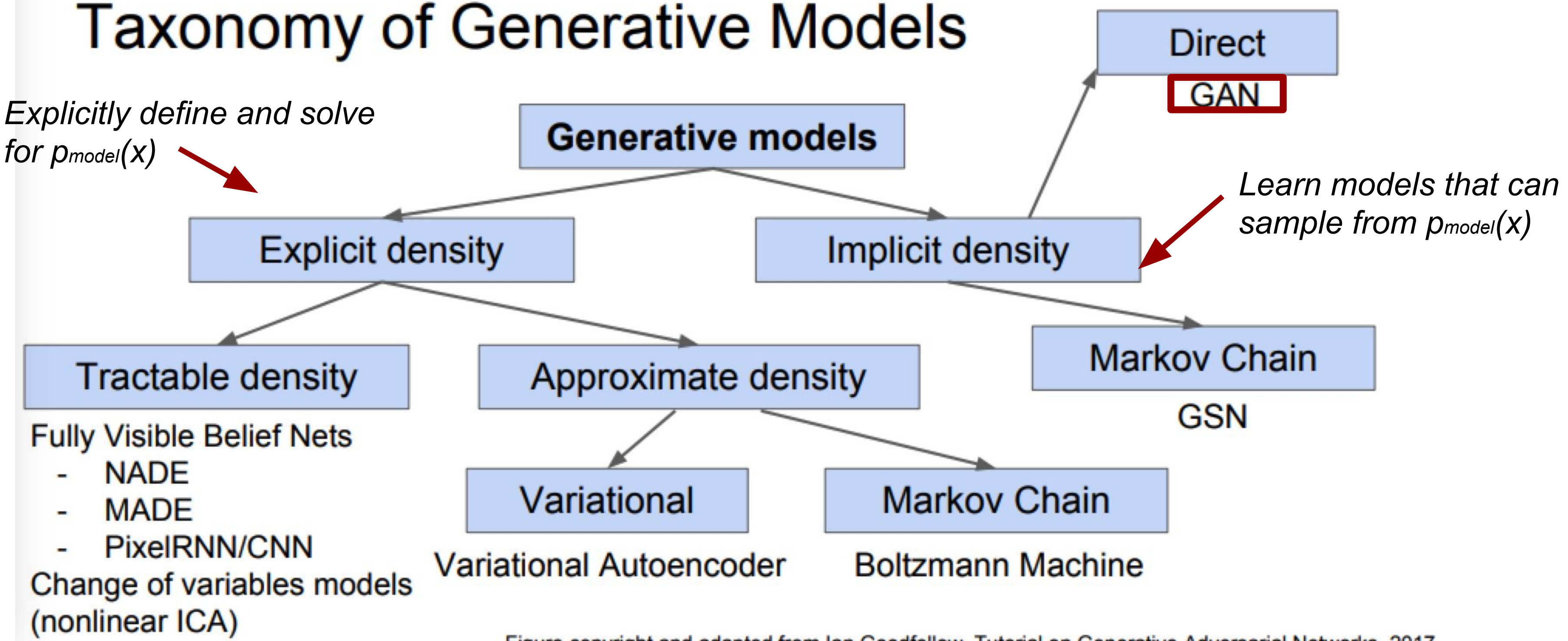



Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

# Generative Adversarial Network

# LeCun's Comment

**What are some recent and potentially upcoming breakthroughs in deep learning**

Yann LeCun, Director of AI Research at Facebook and Professor at NYU
Answered Jul 29, 2016 · Upvoted by Joaquin Quiñonero Candela, studied Machine Learning and Thamme Gowda, M.S. Computer Science, University of Southern California (2017)

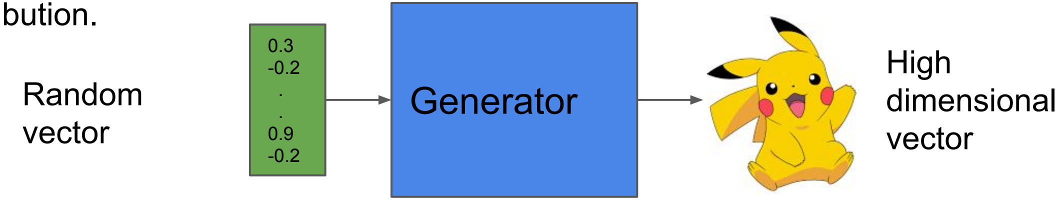
There are many interesting recent development in deep learning, probably too many for me to describe them all here. But there are a few ideas that caught my attention enough for me to get personally involved in research projects.

The most important one, in my opinion, is adversarial training (also called GAN for Generative Adversarial Networks). This is an idea that was originally proposed by Ian Goodfellow when he was a student with Yoshua Bengio at the University of Montreal (he since moved to Google Brain and recently to OpenAI).

This, and the variations that are now being proposed is the most interesting idea in the last 10 years in ML, in my opinion.

# Generative Adversarial Networks

- GANs: do not try to find density function, but only want to sample from distribution
- Two sub-modules:
    - Generator: sample from a simple distribution (random gaussian). Learn transformation to training distribution.




    - Discriminator: detect the sample whether it is from real distribution

# Generator vs Discriminator

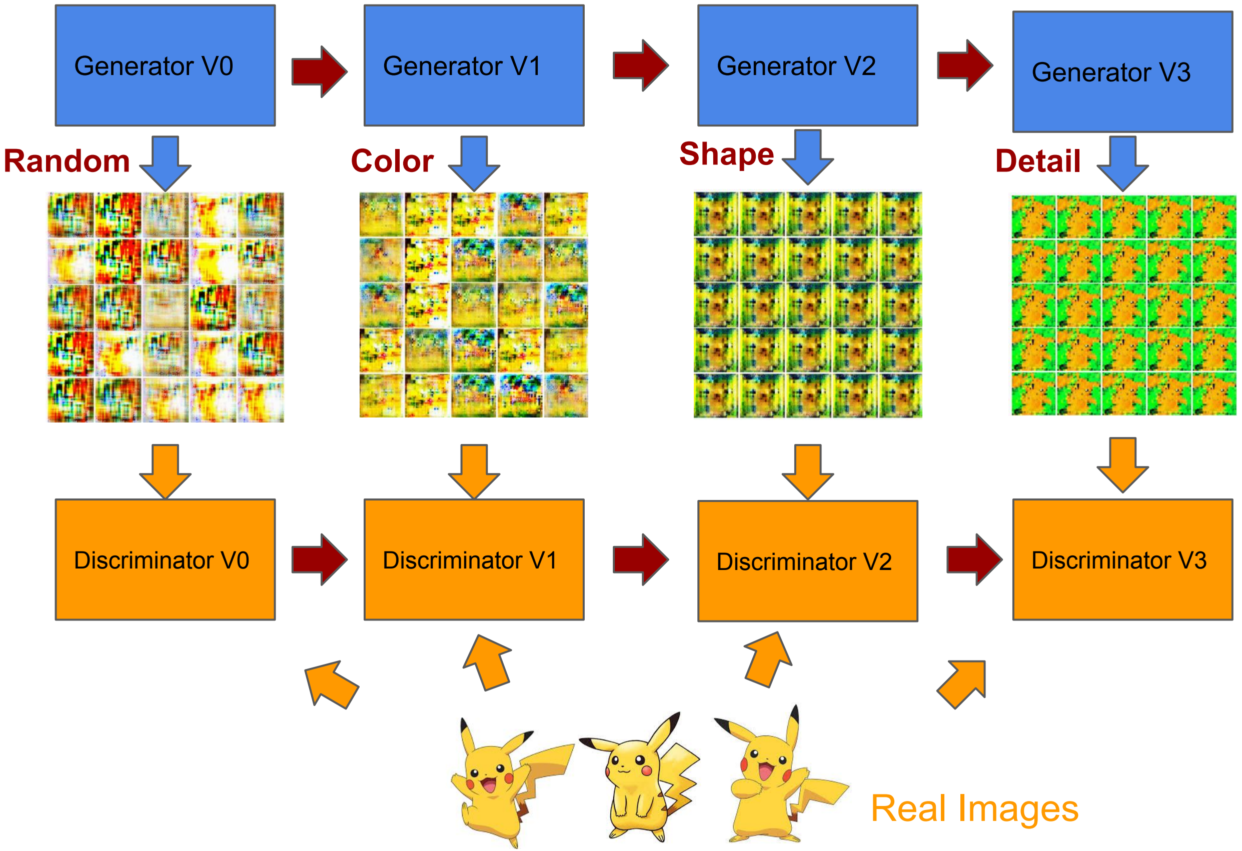
- Generator: try to fool the discriminator by generating real-looking images
- Discriminator: try to distinguish between real and fake images

# Frenemy

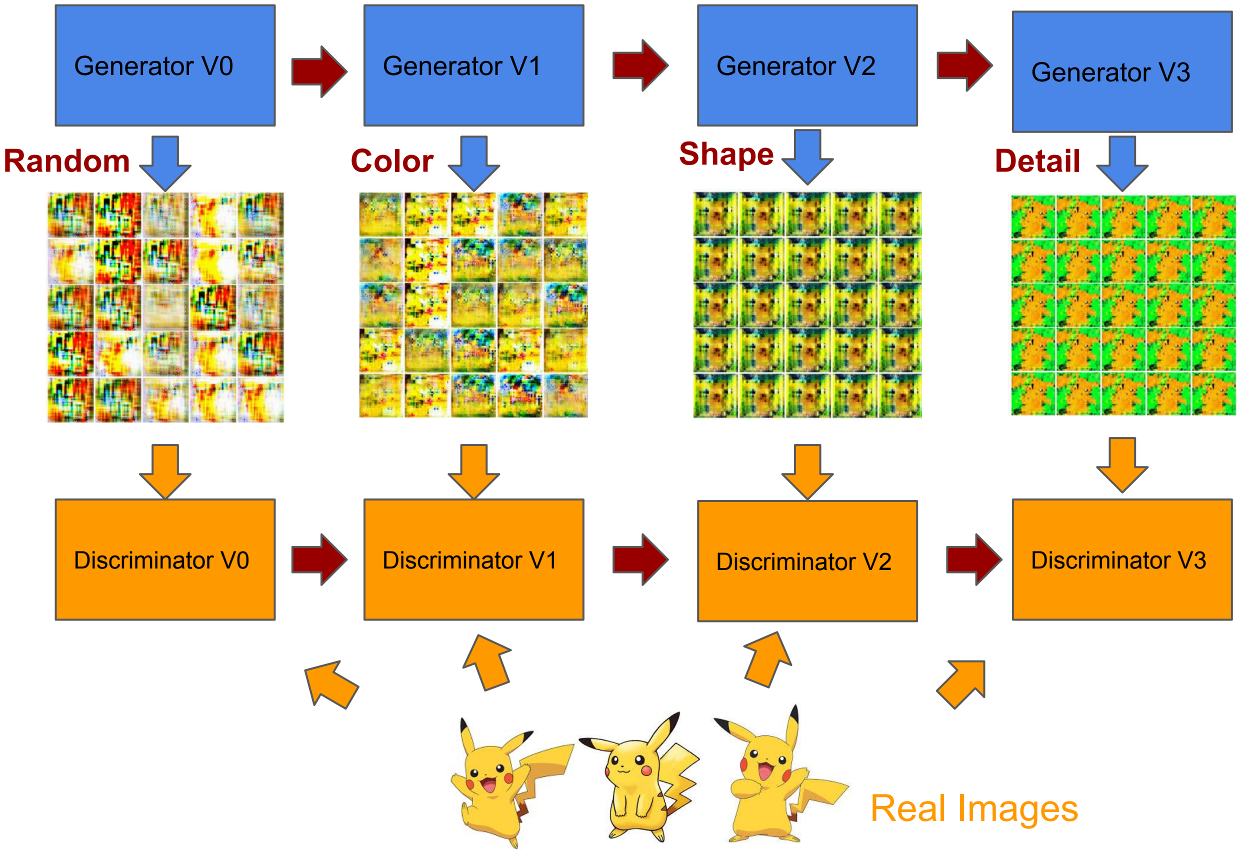
- Generator: try to fool the discriminator by generating real-looking images
- Discriminator: try to distinguish between real and fake images

Frenemy
Love-hate relationship

# Ideas behind GAN



# Ideas behind GAN




Generator and Discriminator are transformation, i.e., neural networks

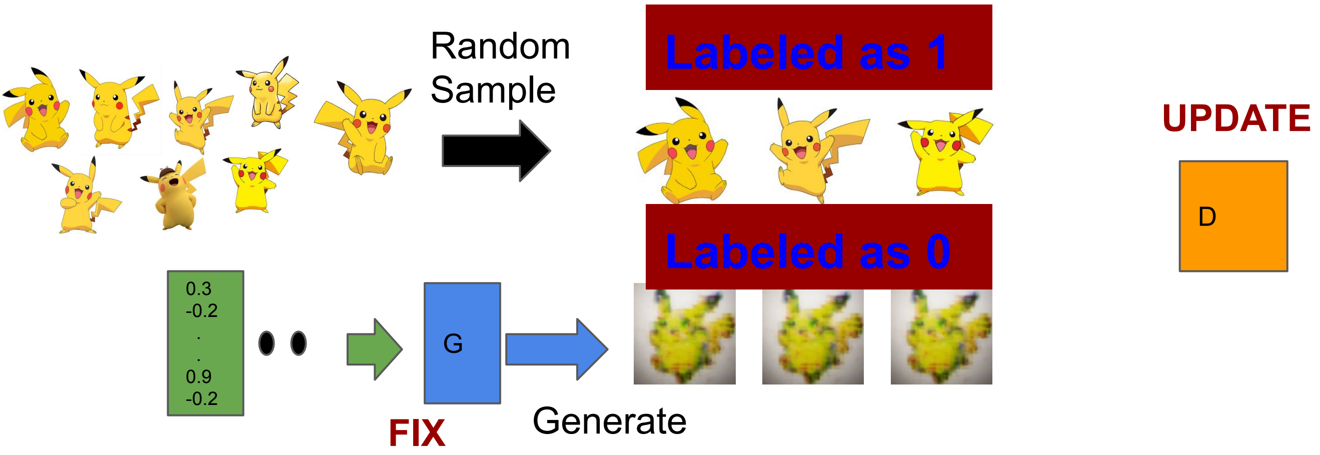
# Algorithms for GAN

- Get the real data collection:

- Randomly initialize generator network G and discriminant network D
- During training, iteratively train G and D

# Algorithms for GAN

- Step 1: Fix Generator G, Train discriminant D




Discriminator are updated to assign high scores to real objects and low scores to generated objects

# Algorithms for GAN

- Step 2: Update Generator G, Fix discriminant D

Generate

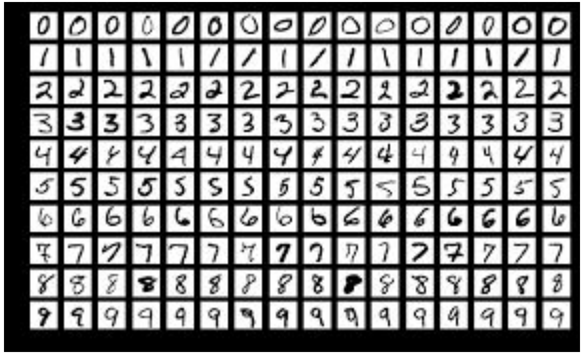

Approximate output to 1

**UPDATE**

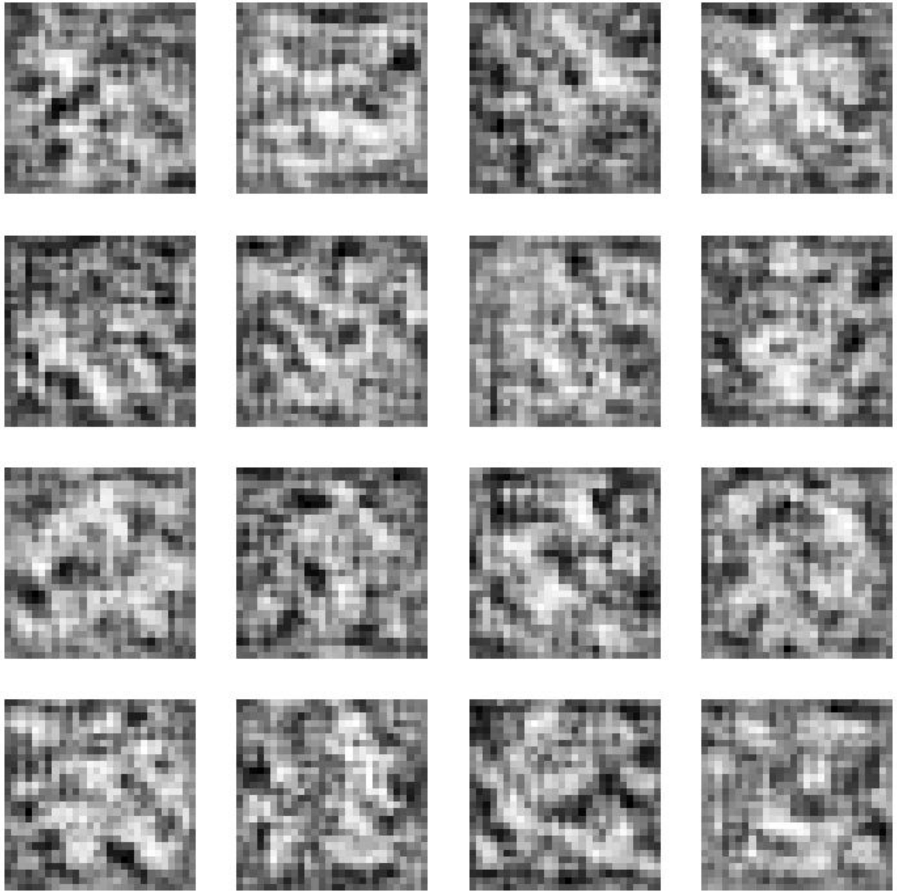
**FIX**

Generator learn to fool the discriminator

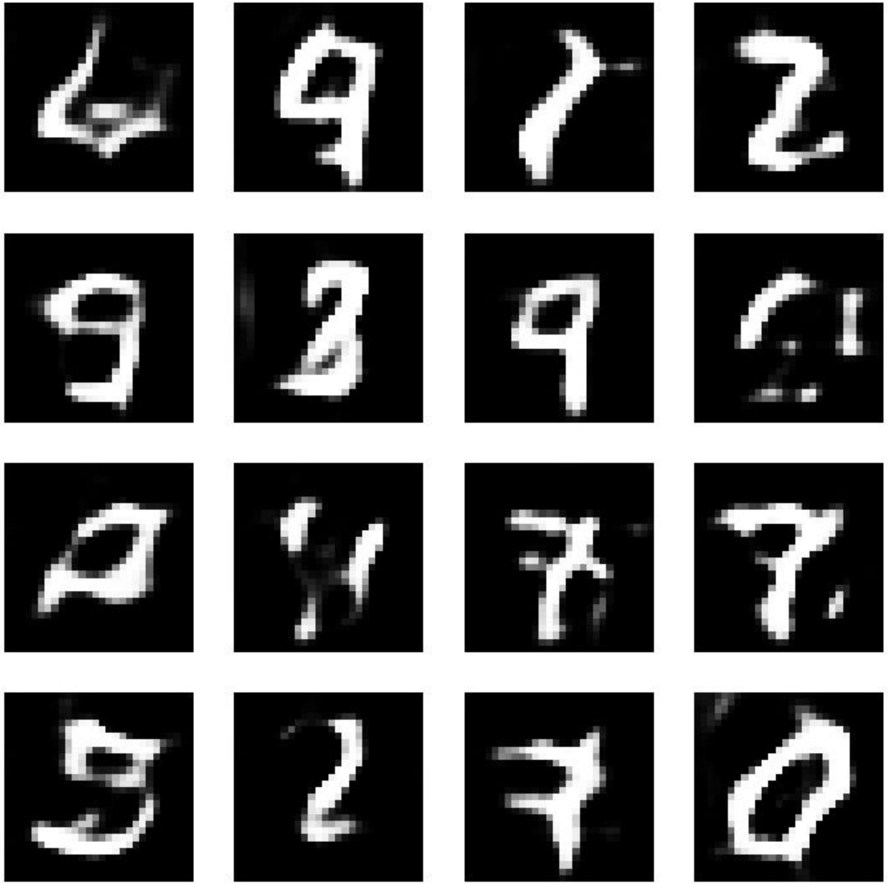
Train GAN over mnist dataset



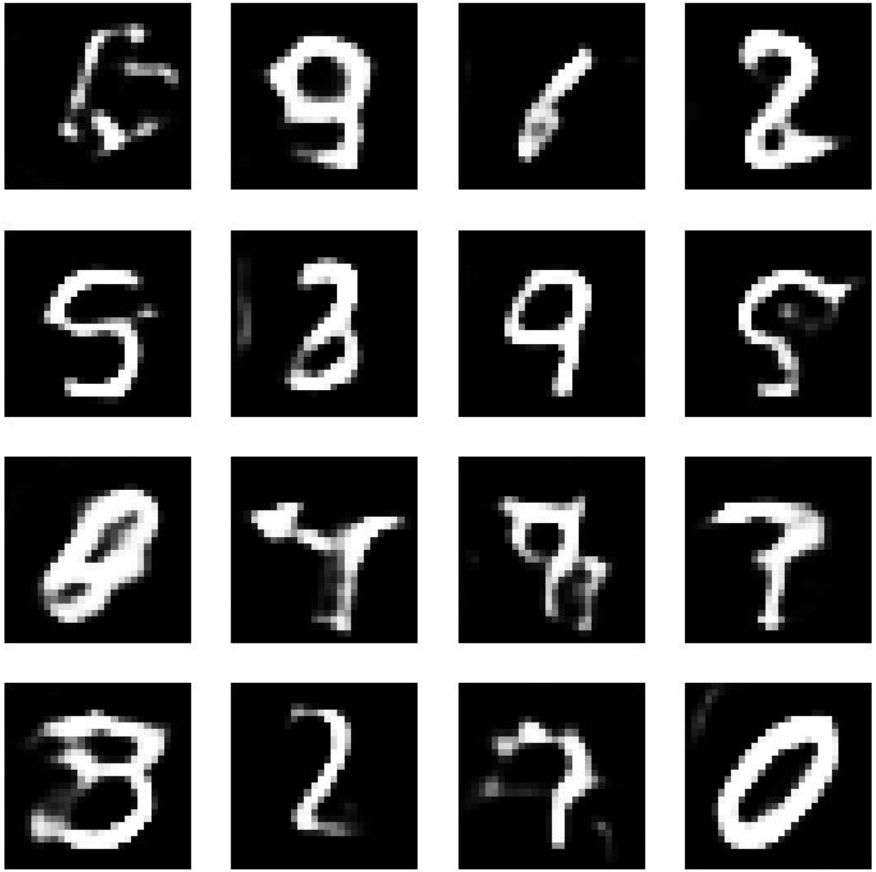
Epoch 0



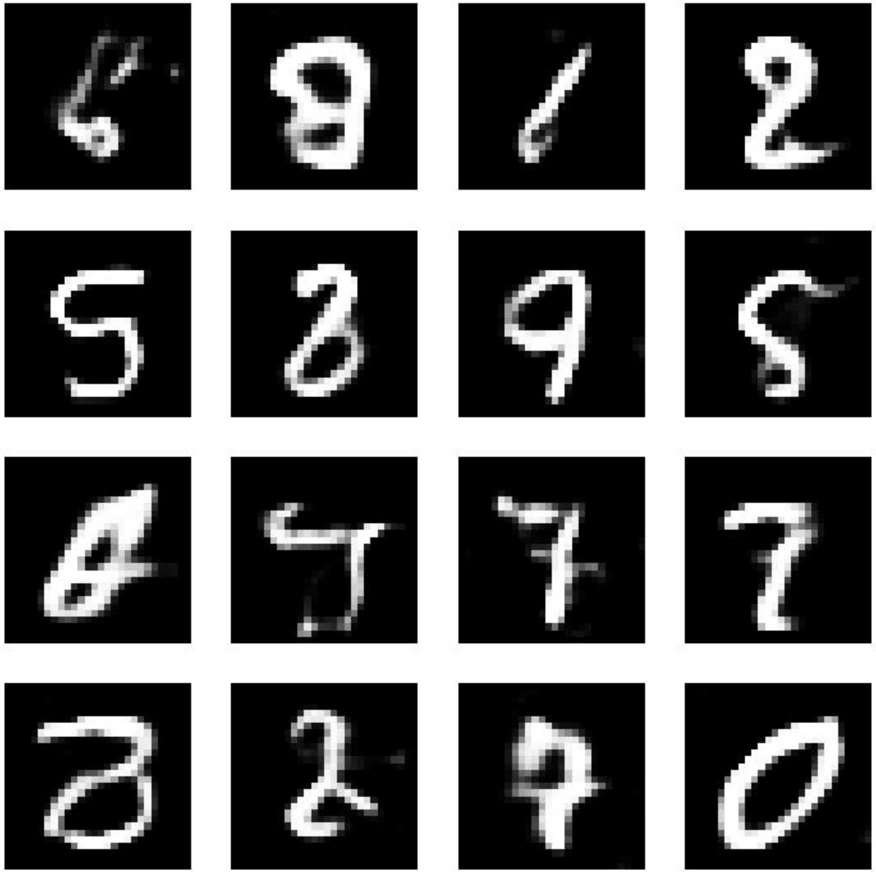
# Epoch 500



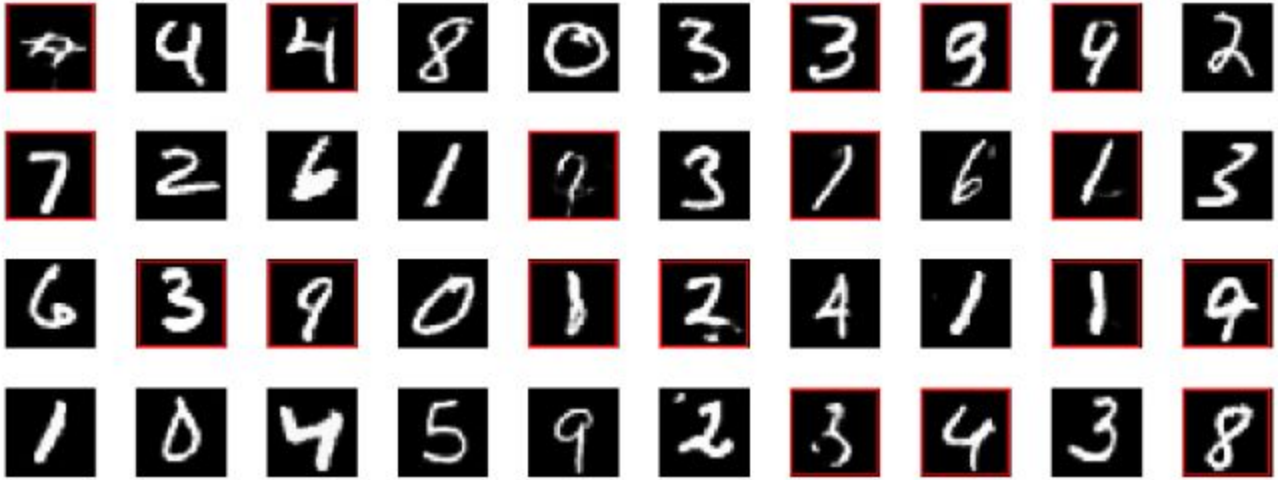
**Epoch 1000**



Epoch 1500



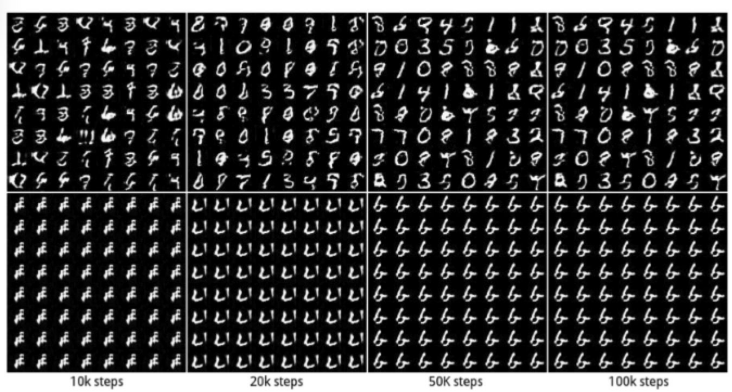
# Final Output



# It is very hard to trainGAN

- Non-convergence: the model parameters oscillate and never converge

- Model collapse: the generator collapses which produces limited modes of samples




Good GAN

Bad GAN

- Diminished gradient: the discriminator gets too successful that the generator gradient vanishes and learns nothing.

# Game Theory and GAN

- GAN is the minimax/zero-sum non-cooperative game

- GAN's minimax equation as:

$$\min_{G}\max_{D} V(D,G) = \mathbb{E}_{x\sim p_r(x)}[log D(x)] + \mathbb{E}_{z\sim p_z(z)}[log(1 - D(G(z)))]$$

Data distribution

Gaussian distribution

**Classify image as real or fake better**

**Fool the discriminator most**

- D's actions are to maximize them and G wants to minimize its actions

- In game theory, GAN model converges when the D and G reach a Nash Equilibrium

# Nash equilibrium

- In minimax game, both sides want to beat the others

- Nash equilibrium is that when one player will not change its action regardless of what the opponent may do

- Consider two players A and B which control the value of x and y, Player A wants to maximize the value xy while B wants to minimize it

$$\underset{B}{min}\underset{A}{max}V(A,B) = xy$$

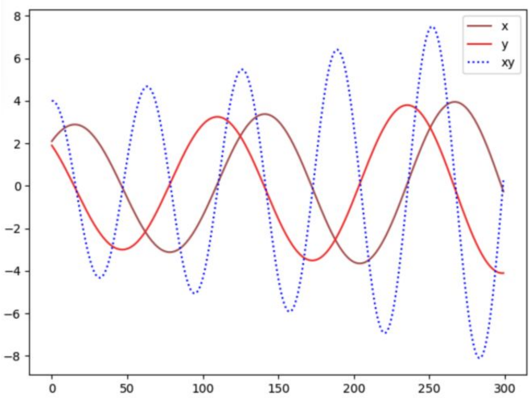
- What are the values of x and y for the nash equilibrium?

# Gradient Descent for Nash equilibrium

- Update parameter x and y following the gradient of the value function

$$\Delta x = \alpha \frac{\partial (xy)}{\partial x} \qquad \Delta y = -\alpha \frac{\partial (xy)}{\partial y}$$

- During training iterations, start with the initial guess of x=2 and y=2




**Epoch Number**

**Can not Converge i.e. x=y=0**


GAN progress on face generation from Ian Goodfellow