

# Approfondimento di Intelligenza Artificiale

Tristano Munini

11 ottobre 2020

# Indice

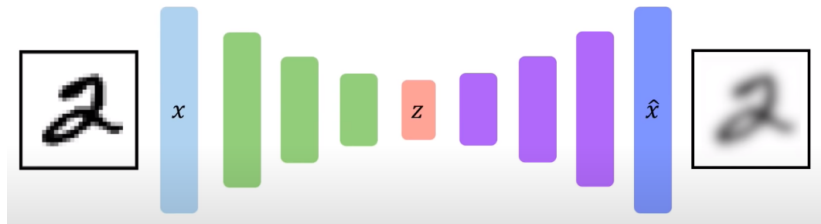
- ▶ AutoEncoders (AE) e Variational AutoEncoders (VAE)
- ▶ Generative Adversarial Network (GAN)
- ▶ Reinforcement Learning (RL)
  - ▶ QNN
  - ▶ Policy Gradient
- ▶ GAN + RL per generazione di testi
  - ▶ SeqGAN
  - ▶ LeakGAN

# AutoEncoders

$x \in \text{real data space}$

$z \in \text{latent space}$

$\hat{x} \in \text{real data space}$

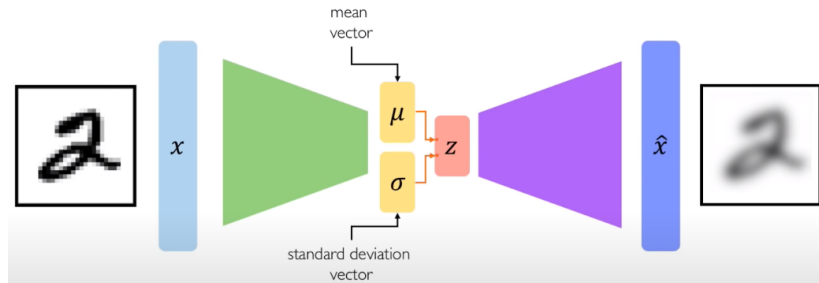


The point of maximum compression, where  $z$  is produced, is called bottleneck of the network

# Variational AutoEncoders

$z$  is sampled from the distribution with mean vector  $\mu$  and standard deviation vector  $\sigma$ .

We want an  $\hat{x}$  that is similar to  $x$  but that has some differences.  
We want a variation of  $x$ .



# Reparametrization Trick

We can not use  $z \sim \mathcal{N}(\mu, \sigma^2)$

So we use

$$z = \mu + \sigma \odot \varepsilon \quad \text{where } \varepsilon \sim \mathcal{N}(0, 1)$$

Where  $\odot$  notes the element-wise multiplication

Now  $z$  is the sum of two fixed vectors,  $\mu$  and  $\sigma$ , and a random constant  $\varepsilon$  used as a weight

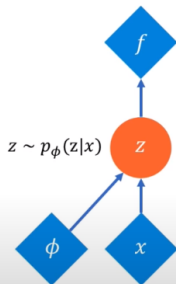
# Reparametrization Trick



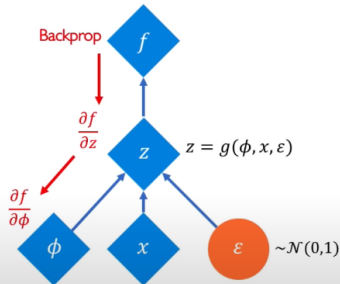
Deterministic node



Stochastic node



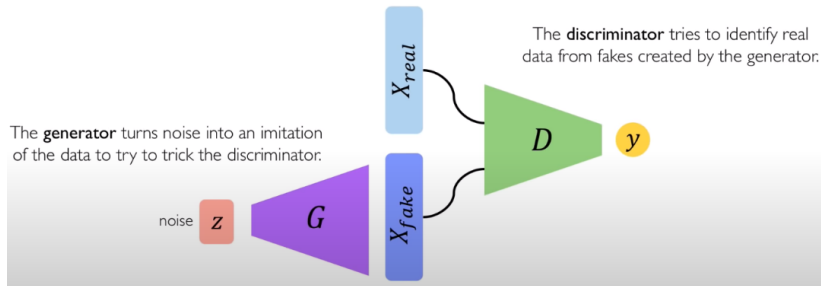
Original form



Reparametrized form

# Generative Adversarial Networks

GANs are a way to make a generative model by having two neural networks compete with each other



$z$  can be sampled from  $\mathcal{N}(0, 1)$  as in the VAEs

# GAN Training

It is a min-max game

$$\begin{aligned} \min_G \max_D V(D, G) = & \mathbb{E}_{x \sim P_{real(x)}} [\log D(x)] \\ & + \mathbb{E}_{z \sim P_z(x)} [\log(1 - D(G(z)))] \end{aligned}$$

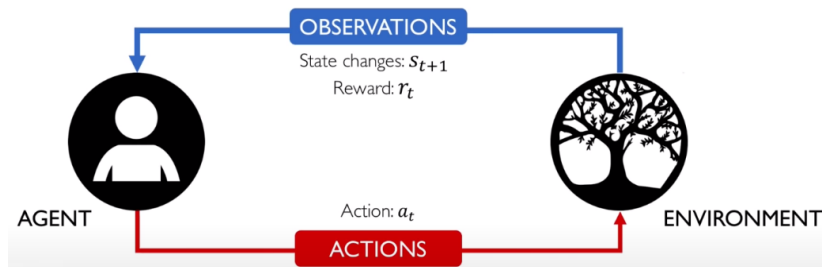
Where  $P_{real(x)}$  is the probability distribution of the real data and  $P_z(x) = \mathcal{N}(0, 1)$  in our case



# GAN Problems

- ▶ Hard to converge to a good solution
- ▶ Vanishing gradient problem
- ▶ Model collapse
- ▶ Hard to find equilibrium

# Reinforcement Learning



Total Reward (Return)

$$R_t = \sum_{i=t}^{\infty} r_i = r_t + r_{t+1} + \dots + r_{t+n} + \dots$$

Discounted Total Reward (Return)

$$R_t = \sum_{i=t}^{\infty} \gamma^i r_i = \gamma^t r_t + \gamma^{t+1} r_{t+1} + \dots + \gamma^{t+n} r_{t+n} + \dots$$

# Quality Function

The quality function, also called Q-function, is the expected total future reward an agent in state  $s_t$  can receive by doing action  $a_t$

$$Q(s_t, a_t) = \mathbb{E}[R_t | s_t, a_t]$$

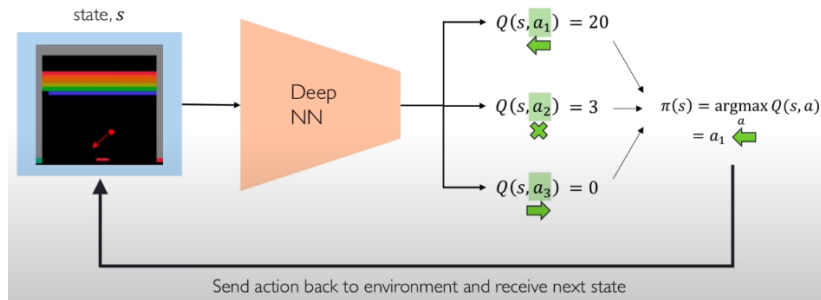
The policy that the agent follows is

$$\pi(s_t) = \operatorname{argmax}_a Q(s_t, a)$$

# QNN

It may be hard to find manually a good Q-Function

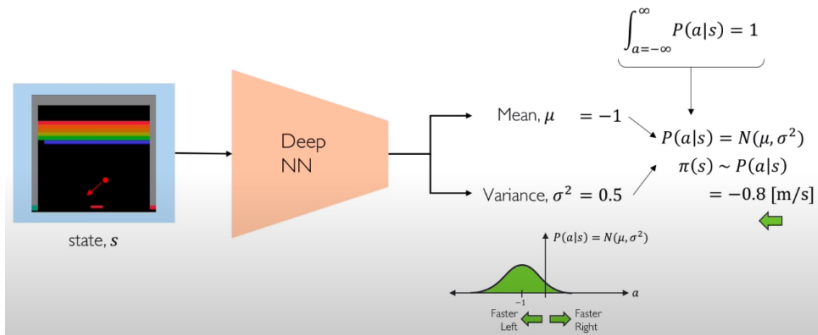
We can use an NN to estimate  $Q(s_t, a)$  for all possible actions  $a$



This solution is limited to discrete action space

# Policy Gradient

What if we need to use an agent within a continuous action space?



# Policy Gradient

The loss used in Policy Gradient is

$$loss = -\log P(a_t | s_t) R_t$$

so we weight the expected return (total reward in the future) with the probability of taking the action that gives this specific return

Remember that the network chooses the next action using  $\pi(s) \sim P(a|s)$

# N-Monte Carlo Rollouts

$$loss = -\log P(a_t | s_t) R_t$$

$R_t$  is a sum over infinity so we need to approximate it

To do so we can use the Monte Carlo Rollouts:

- ▶ use the current policy on current state  $\pi(s_t)$  to calculate  $N$  possible outcomes to state  $s_{t+n}$  in the future
- ▶ sum all the  $N$  rewards  $r_i^k$  where  $i = t, \dots, t+n$  and  $k = 1, \dots, N$
- ▶ divide by  $N$ , so get the mean of the expected future reward from state  $s_t$  with action  $a_t$

# REINFORCE

The REINFORCE Algorithm:

1. from current state  $s_t$  execute several times  $\pi(s_t)$  until termination or state  $s_{t+n}$
2. calculate the mean of the expected future reward
3. use the just calculated  $R_t$  in the loss, and update the network
4. sample an  $a_t$  from the resulting distribution  $\pi(s_t)$ , and repeat from 1

Notice that step 1 and 4 are an explore and exploit step, respectively



# GAN + RL for text

We want to generate long sequences of words (tokens) that resemble human generated sentences

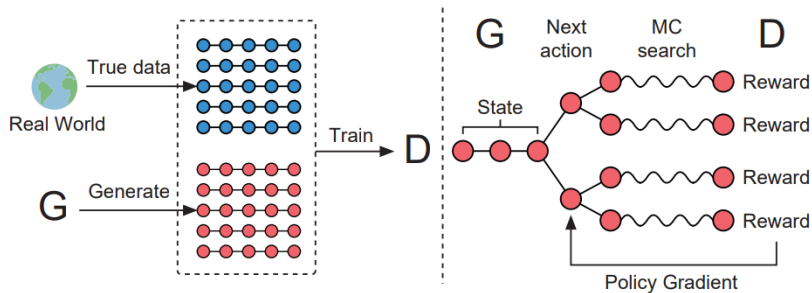
To tackle this difficult problem we use an RNN-based GAN architecture with an RL approach for the Generator

The SeqGAN are the first kind of network presented and can generate sequences up to 20 tokens

The LeakGAN are based on the previous but are more robust and can generate really good results with sentences of 40 words

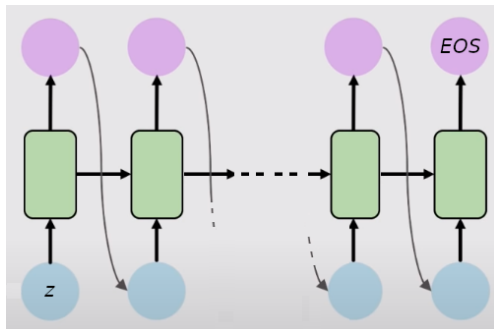
# SeqGAN

## The Sequential GAN architecture



# SeqGAN

The schema of the RNN used inside the Generator of a SeqGAN



The first blue dot is a noise  $z$  and the last pink dot is a special “End Of Sequence” token  
In the case of SeqGAN EOS is generated after 20 tokens

# Loss Function for G

$$J(\theta) = \mathbb{E}[R_t | s_0, \theta] = \sum_{y_1 \in \mathbb{T}} G_\theta(y_1 | s_0) \cdot Q_{D_\phi}^{G_\theta}(s_0, y_1)$$

where  $G_\theta$  is the generator with parameters  $\theta$  and  $Q_{D_\phi}^{G_\theta}(s, a)$  is the future reward gained by doing action  $a$  in state  $s$  following the policy given by  $G_\theta$

Note that  $\mathbb{T}$  is the action space composed by all the legal tokens (words) and  $s_0$  is the noise vector  $z$

# Loss Function for G

We need to estimate  $Q_{D_\phi}^{G_\theta}(s_{t-1}, y_t)$

To do so we use an N-Monte Carlo Search

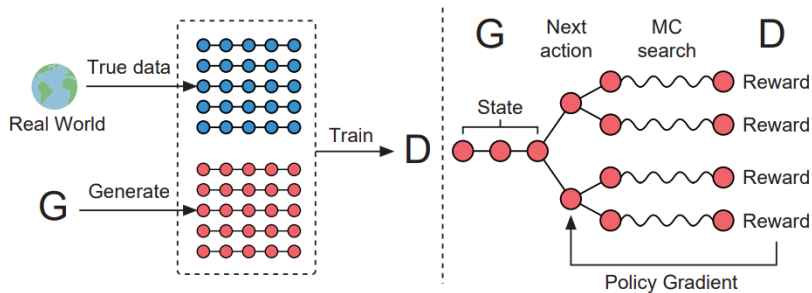
$$\{Y_{1:T}^1, \dots, Y_{1:T}^N\} = MC(Y_{1:t}; N)$$

where  $Y_{1:T}^N$  is a complete sequence composed by a fixed prefix  $Y_{1:t}^N$  and a variable suffix  $Y_{t+1:T}^N$

So we estimate

$$Q_{D_\phi}^{G_\theta}(s = Y_{1:t-1}, a = y_t) = \left\{ \begin{array}{l} \frac{1}{N} \sum_{n=1}^N D_\phi(Y_{1:T}^n), \quad Y_{1:T}^n \in MC(Y_{1:t}; N) \\ D_\phi(Y_{1:t}) \end{array} \right.$$

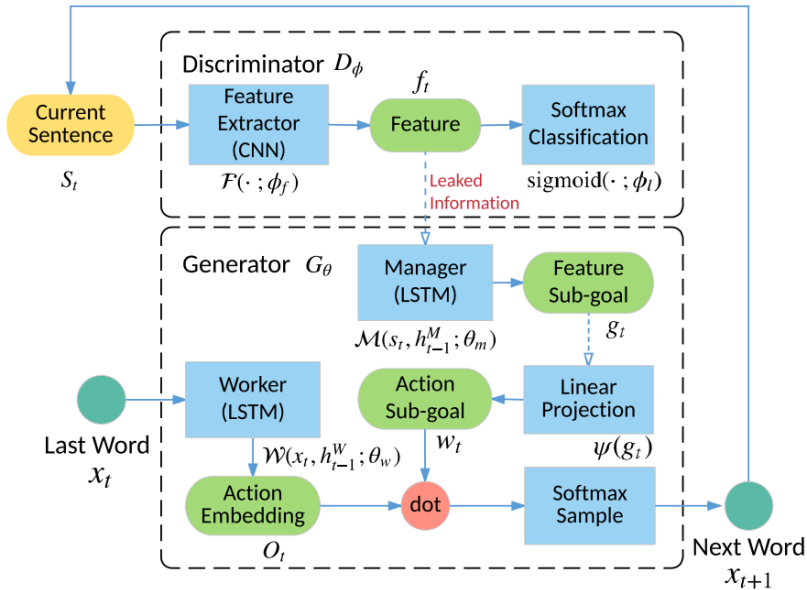
# SeqGAN



# SeqGAN Algorithm

- 1: Initialize  $G_\theta$ ,  $D_\phi$  with random weights  $\theta$ ,  $\phi$
- 2: Pre-train  $G_\theta$  using MLE on real data
- 3: Generate negative samples using  $G_\theta$  for training  $D_\phi$
- 4: Pre-train  $D_\phi$  via minimizing the cross entropy
- 5: **repeat**
- 6:     **for** g-steps **do**
- 7:         Generate a sequence  $Y_{1:T} = (y_1, \dots, y_T) \sim G_\theta$
- 8:         **for**  $t$  in  $1 : T$  **do**
- 9:             Compute  $Q_{D_\phi}^{G_\theta}(s = Y_{1:T}; a = y_t)$
- 10:         **end for**
- 11:         Update generator parameters via policy gradient
- 12:     **end for**
- 13:     **for** d-steps **do**
- 14:         Use current  $G_\theta$  to generate negative examples and combine with given positive examples
- 15:         Train  $D_\phi$  for  $k$  epochs
- 16:     **end for**
- 17: **until** SeqGAN converges

# LeakGAN





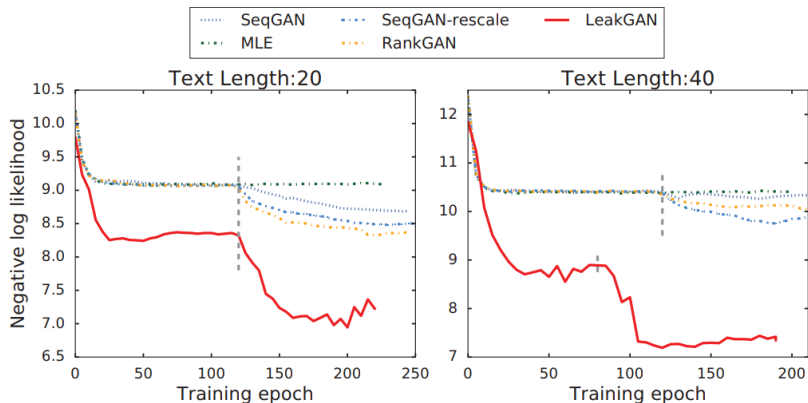
# LeakGAN Algorithm

- 1: Initialize  $G_{\theta_m, \theta_w}$ ,  $D_\phi$  with random weights  $\theta_m, \theta_w$ ,  $\phi$
- 2: Pre-train  $D_\phi$  on real data and generated data
- 3: Pre-train  $G_{\theta_m, \theta_w}$  using leaked information from  $D_\phi$
- 4: Perform the two parts of pre-training interleavingly until convergence
- 5: **repeat**
- 6:     **for** g-steps **do**
- 7:         Generate a sequence  $Y_{1:T} = (y_1, \dots, y_T) \sim G_{\theta_m, \theta_w}$
- 8:         **for**  $t$  in  $1 : T$  **do**
- 9:             Store leaked information from  $D_\phi$
- 10:            Get  $Q(f_t, g_t)$  by Monte Carlo Search
- 11:            Get the computed direction  $g_t$  from MANAGER
- 12:            Update WORKER parameters  $\theta_w$ ,  $\psi$ , softmax
- 13:            Update MANAGER parameters  $\theta_m$
- 14:         **end for**
- 15:     **end for**
- 16:     **for** d-steps **do**
- 17:         Use current  $G_{\theta_m, \theta_w}$  to generate negative examples
- 18:         Train  $D_\phi$  for  $k$  epochs on generated examples and real data
- 19:     **end for**
- 20: **until** LeakGAN converges

# Avoiding Model Collapse

- ▶ Pre-Train
- ▶ Dropout
- ▶ Regularization
- ▶ Interleaved Training

# Comparison



LeakGAN

- (1) A man sitting in front of a microphone with his dog sitting oh his shoulder.
- (2) A young man is holding a bottle of wine in his hand.

SeqGAN

- (1) A couple of kids in a bathroom that is in a bathroom.
- (2) A bathroom with tiled walls ad shower on it.