Advanced Machine Learning - A.A. 2022-2023

Diffusion Models

Raffaele Cerizza
845512

Giacomo Savazzi
845372

Thomas Barbera
845538

Introduzione

I **Diffusion Models** appartengono alla categoria dei **Deep Generative Models**, come *GAN*, *VAE* e *Normalizing Flow*. Recentemente hanno acquisito notorietà per il successo di alcune loro applicazioni, come **DALL·E**.

In questo lavoro sono state analizzate quattro differenti tipologie di Diffusion Models non condizionali, quali:

- *Denoising Diffusion Probabilistic Model*;
- *Denoising Diffusion Implicit Model*;
- *Noise Conditional Score Network*;
- *Score-Based Generative Model tramite Equazioni Differenziali Stocastiche*.

Esempi di immagini generate con DALL·E 2 [1]

Fondamenti Teorici

01 Catene di Markov

02 Forward diffusion

03 Reverse diffusion

04 Addestramento

Catene di Markov

Proprietà di Markov:
Una catena di Markov è un processo stocastico che soddisfa la proprietà di Markov:

$$P(X_{t+1} = x_{t+1} \mid X_t = x_t, X_{t-1} = x_{t-1}, ..., X_0 = x_0) = P(X_{t+1} = x_{t+1} \mid X_t = x_t)$$

Definizione:
Una catena di Markov è caratterizzata da:

- uno spazio degli stati:

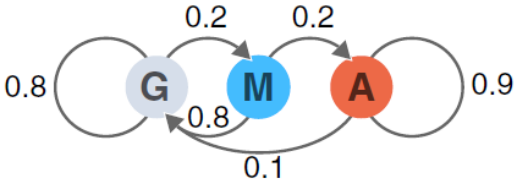
$$S = \{S_1, S_2, ..., S_N\},$$

- una matrice delle probabilità di transizione tra stati:

$$p_{mat,ij} = P(X_{t+1} = S_j \mid X_t = S_i),$$

- una distribuzione iniziale delle probabilità di transizione:

$$\pi_0(i) = P\{X_0 = S_i\}.$$



$$\mathbf{T} = \begin{bmatrix} p_{GG} & p_{GM} & p_{GA} \\ p_{MG} & p_{MM} & p_{MA} \\ p_{AG} & p_{AM} & p_{AA} \end{bmatrix} = \begin{bmatrix} 0.8 & 0.2 & 0 \\ 0.8 & 0 & 0.2 \\ 0.1 & 0 & 0.9 \end{bmatrix}$$

$$\mathbf{T}^{50} = \begin{bmatrix} 0.625 & 0.125 & 0.25 \\ 0.625 & 0.125 & 0.25 \\ 0.625 & 0.125 & 0.25 \end{bmatrix} \approx \lim_{n \to \infty} \mathbf{T}^n$$

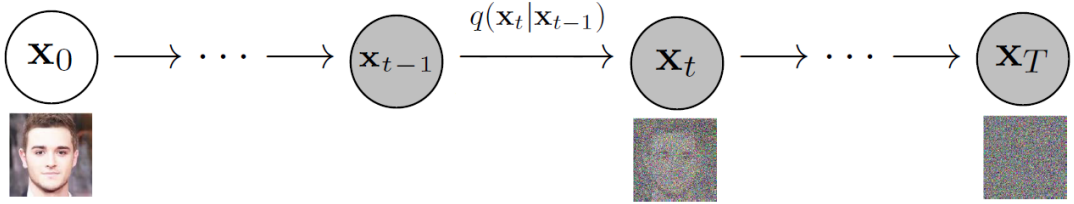
Esempio di catena di Markov [2].

Forward diffusion process

Il **forward diffusion process** converte gradualmente un dato appartenente alla distribuzione dei dati reali $q(x_0)$ in un nuovo dato appartenente a una distribuzione più semplice, tipicamente Gaussiana.
In particolare il dato viene convertito attraverso un processo stocastico che aggiunge progressivamente rumore Gaussiano in una serie di T passi. Ad ogni passo viene generato un nuovo dato avente le stesse dimensioni del dato di partenza, ma più rumoroso. Questo processo è una catena di Markov caratterizzata dal kernel di transizione:

$$q(x_t \mid x_{t-1}) = \mathcal{N}(\underbrace{x_t}_{\text{output}}; \underbrace{\sqrt{1-\beta_t}x_{t-1}}_{\text{media}}, \underbrace{\beta_t \mathbf{I}}_{\text{covarianza}})$$

Il valore β_t è il valore della *variance schedule* al tempo t. Determina il livello del rumore aggiunto.
All'ultimo passo del processo il dato assume una distribuzione Gaussiana isotropa.



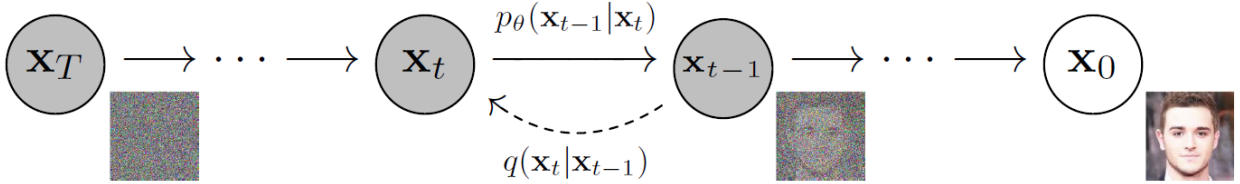
Forward diffusion process [3]

Reverse diffusion process

Il **reverse diffusion process** inverte gradualmente il forward diffusion process. Parte da un dato rumoroso $p(x_T) = \mathcal{N}(x_T; 0, \mathbf{I})$ e a ogni passo rimuove una parte del rumore. Alla fine del processo si ottiene un dato appartenente alla distribuzione originaria $q(x_0)$.
In questo caso il calcolo del kernel di transizione $p(x_{t-1} \mid x_t)$ è intrattabile. Pertanto occorre stimare un kernel di transizione parametrizzato:

$$p_\theta(x_{t-1} \mid x_t) = \mathcal{N}(\underbrace{x_{t-1}}_{\text{output}}; \underbrace{\mu_\theta(x_t, t)}_{\text{media}}, \underbrace{\Sigma_\theta(x_t, t)}_{\text{covarianza}})$$

In questo caso la media e la covarianza risultano parametrizzate. Questi parametri possono essere ottimizzati attraverso l'addestramento di uno specifico modello.
Sebbene il calcolo di $p(x_{t-1} \mid x_t)$ sia intrattabile, è invece possibile calcolare $p(x_{t-1} \mid x_t, x_0)$.



Reverse diffusion process [3]

Addestramento (1/3)

L'addestramento di un Diffusion Model ha l'obiettivo di ottimizzare i parametri del kernel di transizione inverso. Tuttavia non è possibile ottimizzare direttamente $p_\theta(x_0)$ in quanto questo calcolo è intrattabile. Per risolvere questo problema si utilizzano l'Evidence Lower Bound e la Divergenza di Kullback-Leibler.

- **Evidence Lower Bound (ELBO):**
  L'ELBO permette di trasformare un problema di inferenza intrattabile in un problema di ottimizzazione risolvibile utilizzando algoritmi noti. L'ELBO sulla *negative log-likelihood* può essere definito come:

$$\mathbb{E}[-\log p_\theta(x_0)] \leq \mathbb{E}_q\left[-\log \frac{p_\theta(x_{0:T})}{q(x_{1:T} \mid x_0)}\right]$$

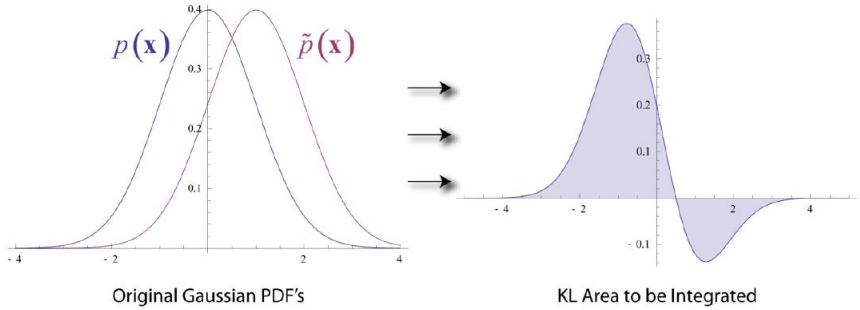
  L'ELBO è il membro destro della disequazione. Minimizzando l'ELBO si minimizza anche la *negative log-likelihood*.

Addestramento (2/3)

- **Divergenza di Kullback-Leibler:**
  I termini dell'ELBO possono essere manipolati sfruttando la divergenza di Kullback-Leibler. La divergenza di Kullback-Leibler tra due distribuzioni di probabilità è definita come:

$$D_{\mathrm{KL}}(p||q) = \sum_{x \in X} p(x) \log \frac{p(x)}{q(x)}$$

Il valore D_{KL} misura l'informazione persa quando q è usata per approssimare p.
La divergenza tra due distribuzioni Gaussiane può essere calcolata analiticamente in forma chiusa.



Esempio di calcolo della divergenza di Kullback-Leibler tra due distribuzioni Gaussiane [4].

Addestramento (3/3)

- **Loss function:**
  La generica loss function da ottimizzare è un ELBO definito come:

$$L = \mathbb{E}_q \Big[\underbrace{D_{\mathrm{KL}}(q(x_T \mid x_0) || p(x_T))}_{L_T} + \sum_{t>1} \underbrace{D_{\mathrm{KL}}(q(x_{t-1} \mid x_t, x_0) || p_\theta(x_{t-1} \mid x_t))}_{L_{t-1}} \underbrace{- \log p_\theta(x_0 \mid x_1)}_{L_0} \Big]$$

  - Il primo termine L_T specifica quanto $p(x_T)$ sia prossimo a una distribuzione Gaussiana. Questo termine non presenta parametri da ottimizzare. Pertanto può essere ignorato durante l'addestramento.
  - Il secondo termine L_{t-1} comporta che per ogni istante di tempo il modello sia addestrato in modo che $p_\theta(x_{t-1} \mid x_t)$ sia il più possibile prossimo alla probabilità a posteriori del forward diffusion process condizionata al dato originale.
  - Il terzo termine L_0 è il *reconstruction term* e misura semplicemente la probabilità logaritmica del dato originale x_0 data la prima variabile latente x_1.

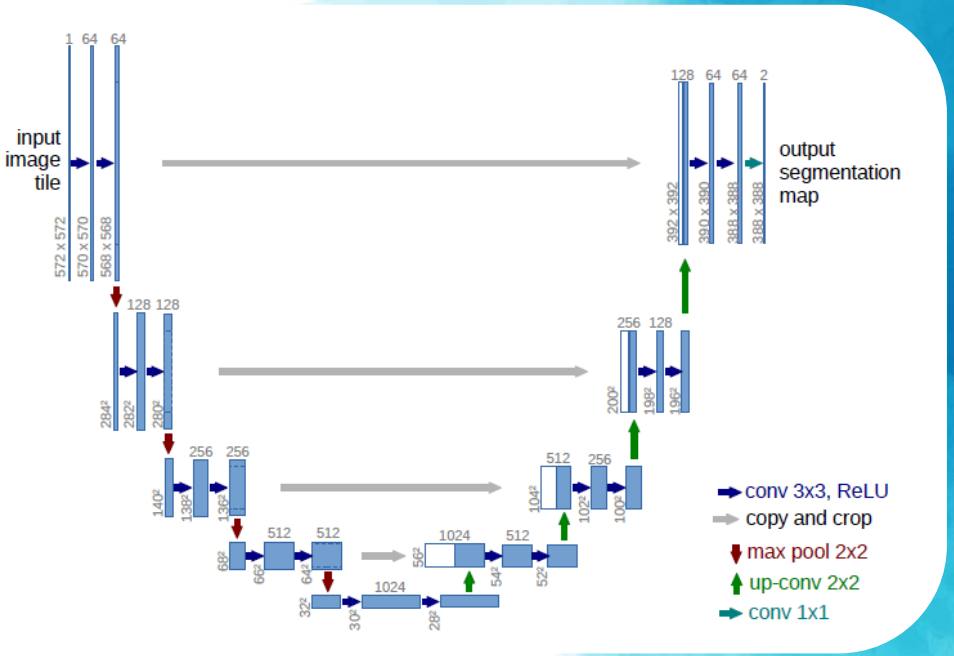
Architettura: U-Net (1/2)

Il modello tipicamente utilizzato per l'addestramento dei Diffusion Models è quello della architettura **U-Net**.

Prima Versione: la prima versione di U-Net prevede:

- *Parte Riduttiva*, utilizza una serie di layer di convoluzione per catturare le feature più importanti delle immagini;
- *Parte Espansiva*, utilizza una serie di blocchi per localizzare le informazioni estratte e ricostruire le immagini.

Ogni blocco della parte espansiva si compone di: un *layer di upsampling*, un l*ayer di up-convolution 2x2*, la concatenazione con la mappa delle feature tramite *skip connection*, e due *layer di convoluzione 3x3* con funzione *ReLU*. In tutto si hanno 23 layer convoluzionali, e nessun layer completamente connesso.

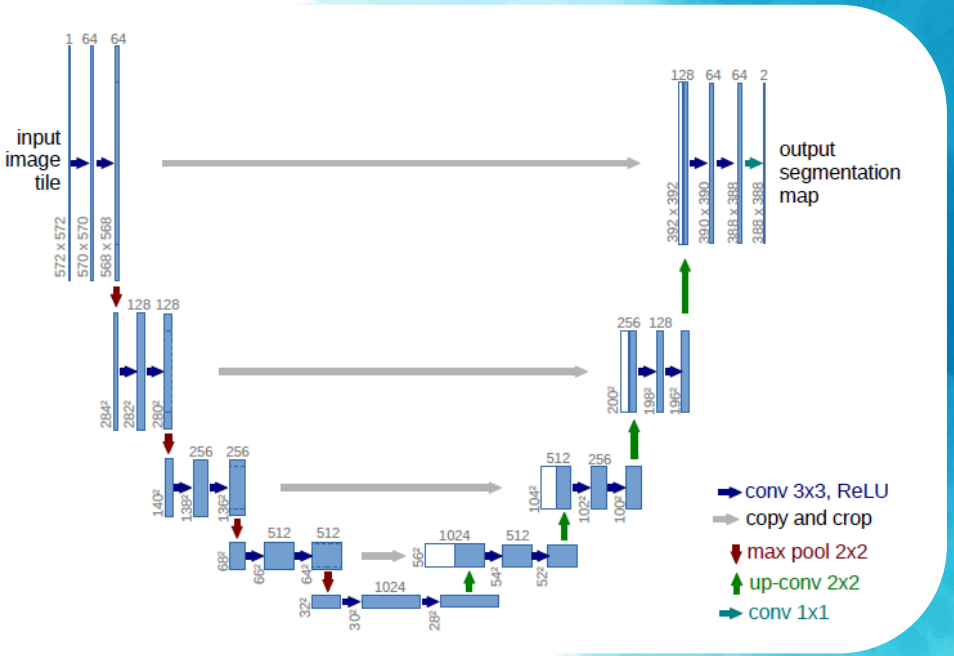


Rappresentazione dell'architettura di U-Net [5]

Architettura: U-Net (2/2)

Utilizzando un padding maggiore di 0 nei layer convoluzionali è possibile ottenere output con le stesse dimensioni dell'immagine in input. Questo risulta particolarmente utile per i Diffusion Models. Infatti, forward diffusion process e reverse diffusion process producono a ogni passo un output con stesse dimensioni del dato fornito in input, ma con diversi livelli di rumore.

Versione migliorata: nuova versione di U-Net, che offre performance migliori nella generazione di immagini. Le modifiche principali prevedono l'aggiunta di un *global attention layer*, l'utilizzo di *Adaptive Group Normalization*, modifica ai blocchi della rete ispirandosi a *BigGAN*, e riscalati i valori delle skip connections.



Rappresentazione dell'architettura di U-Net [5]

Diffusion Models

01

DDPM

Denoising Diffusion Probabilistic Model

02

DDIM

Denoising Diffusion Implicit Model

03

NCSN

Noise Conditional Score Network

04

SBGM-SDE

Score-Based Generative Model attraverso Equazioni Differenziali Stocastiche

DDPM – Peculiarità

Le pecularità dei **DDPM** sono le seguenti:

- Varianza β_t:
  I valori β_t sono fissati e crescenti nel tempo. La crescita può essere lineare o basata su una *cosine schedule*.

- Covarianza Σ_θ:
  Anche la covarianza viene fissata in modo da essere costante. In particolare $\Sigma_\theta(x_t,t)$ è fissata pari a $\sigma^2\mathbf{I}$, dove $\sigma^2 = \beta_t$.

- Loss function:
  La loss function è semplificata come:

$$L_{\text{simple}} = \mathbb{E}_{t,x_0,\epsilon}[\|\epsilon - \epsilon_\theta(x_t, t)\|^2]$$

  Il valore $\varepsilon_\theta(x_T, t)$ rappresenta il rumore del dato predetto dal modello.
  La loss function si riduce all'errore quadratico medio tra il rumore vero e quello predetto.

DDPM – Algoritmi

- Algoritmi di addestramento e campionamento:

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$$\nabla_\theta \left\| \epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t) \right\|^2$$
6: **until** converged

Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \dots, 1$ **do**
3: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

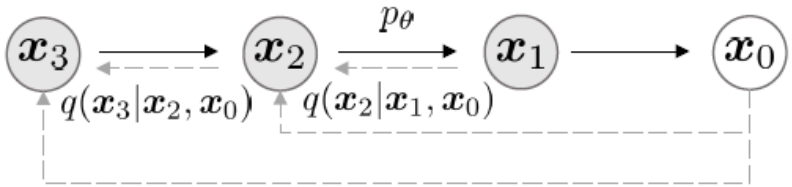
Algoritmi di addestramento e campionamento dei DDPM [3]

  - Durante l'addestramento viene campionato un dato reale a cui viene aggiunto rumore Gaussiano in corrispondenza di un istante di tempo *t*. La rete predice il rumore del dato per diversi istanti di tempo. Il rumore predetto viene usato per calcolare la loss function e aggiornare i pesi della rete.
  - L'algoritmo di campionamento genera nuovi dati a partire da rumore Gaussiano. A ogni passo viene rimossa una parte del rumore. Tipicamente si usano 1000 passi di inferenza.

DDIM – Peculiarità

Le peculiarità dei **DDIM** sono le seguenti:

- Il forward diffusion process non soddisfa più la proprietà di Markov perché il kernel di transizione dipende anche dal valore iniziale x_0.



Processo di diffusione dei DDIM [6].

- La loss function è equivalente a quella dei DDPM.
- Il campionamento per ogni istante di tempo è definito come:

$$x_{t-1} = \sqrt{\overline{\alpha}_{t-1}} \underbrace{\left(\frac{x_t - \sqrt{1-\overline{\alpha}_t}\epsilon_\theta(x_t)}{\sqrt{\overline{\alpha}_t}} \right)}_{\text{predicted } x_0} + \underbrace{\sqrt{1-\overline{\alpha}_{t-1}-\sigma_t^2} \cdot \epsilon_\theta(x_t)}_{\text{direction pointing to } x_t} + \underbrace{\sigma_t \epsilon_t}_{\text{random noise}}$$

  Il processo di campionamento diventa deterministico quando σ_t è molto piccolo per ogni istante di tempo.
- Il vantaggio principale rispetto ai DDPM consiste nella possibilità di generare campioni di buona qualità in un tempo molto inferiore.

NCSN – Peculiarità

Le **NCSN** vengono introdotte per superare le limitazioni dei modelli generativi *Score-Based*.

- Basate su stima e campionamento tramite **Score** della distribuzione dei dati. Lo Score rappresenta un vettore che punta nella direzione dove si ha più alta densità di dati;

$$\nabla_x \log p(x)$$

- La generazione si basa sull'addestramento di una rete neurale tramite *Score-Matching* per apprendere il vettore dello Score. Saranno prodotti campioni utilizzando la **dinamica di Langevin**: questo algoritmo opera muovendo gradualmente un dato random di partenza verso aree di alta densità, spostandosi lungo il vettore stimato per lo Score;
- Le *NCSN* prevedono la perturbazione dei dati con diversi livelli di rumore, e addestramento di un'unica rete per stimare lo Score rispetto a tutti i livelli di rumore considerati;
- La dinamica di Langevin partirà con lo score relativo alla distribuzione con più alto livello di rumore, che gradualmente si andrà a ridurre, fino ad arrivare a considerare lo score della reale distribuzione dei dati. Questo aiuterà a trasferire a mano a mano i benefici dati dai livelli più alti di rumore sui livelli più bassi di rumore, dove i dati perturbati sono praticamente indistinguibili da quelli reali.

NCSN - Addestramento

L'obiettivo è addestrare una rete che permetta la stima dello Score per tutti i diversi livelli di rumore.

- La sequenza di livelli di rumore deve essere tale che il primo sia abbastanza grande da permettere di evitare le difficoltà note dei modelli *Score-Based*, mentre l'ultimo sia abbastanza piccolo da rendere l'ultima distribuzione indistinguibile da quella reale;
- Come architettura della rete si considera *U-Net*, con l'utilizzo di livelli di convoluzione Dilated nella fase di subsampling, e Instance Normalization al posto di Batch Normalization;
- Per l'addestramento si considera solitamente la funzione di **Denoising Score-Matching**, perché più veloce di *Sliced Score-Matching*, e per sua natura adatta alla stima dello Score di distribuzioni perturbate;

$$\mathcal{L}(\theta; \{\sigma_i\}_{i=1}^{L}) = \frac{1}{L} \sum_{i=1}^{L} \lambda(\sigma_i) l(\theta; \sigma_i)$$

Funzione obiettivo unificata per l'addestramento di una NCSN [7].

NCSN – Inferenza

- Per la generazione di campioni viene considerata una versione annealed della dinamica di Langevin;
- L'algoritmo inizia generando un campione appartenente ad una regione ad alta densità della distribuzione più perturbata;
- Questo campione sarà mosso di iterazione in iterazione in zone ad alta densità della distribuzione perturbata successiva;
- Alla fine si potrà ottenere un campione con buona probabilità appartenente ad una regione ad alta densità nella distribuzione reale.

Algorithm 1 Annealed Langevin dynamics.

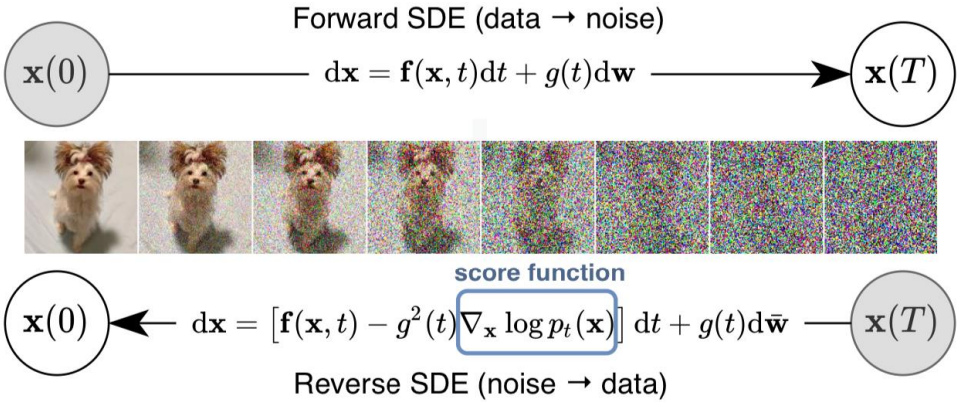
Require: $\{\sigma_i\}_{i=1}^{L}, \epsilon, T$.

Initialize $\tilde{\mathbf{x}}_0$
for $i \leftarrow 1$ to L **do**
  $\alpha_i \leftarrow \epsilon \cdot \sigma_i^2/\sigma_L^2$ ▷ α_i is the step size.
  for $t \leftarrow 1$ to T **do**
    Draw $\mathbf{z}_t \sim \mathcal{N}(0, I)$
    $\tilde{\mathbf{x}}_t \leftarrow \tilde{\mathbf{x}}_{t-1} + \frac{\alpha_i}{2}\mathbf{s}_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}_{t-1}, \sigma_i) + \sqrt{\alpha_i}\,\mathbf{z}_t$
  end for
  $\tilde{\mathbf{x}}_0 \leftarrow \tilde{\mathbf{x}}_T$
end for
return $\tilde{\mathbf{x}}_T$

Versione annealed della dinamica di Langevin [7]

SBGM-SDE

I modelli di *Score Matching* tramite dinamica di Langevin e i DDPM prevedono –implicitamente o meno– il calcolo di uno score per le varie scale di rumore. Essi appartengono quindi alla categoria degli **Score-Based generative models** (SBGM), che possono essere generalizzati con l'uso di **equazioni stocastiche differenziali** (SDE).
In questa generalizzazione non si perturbano i dati con un numero finito di distribuzioni di rumore, ma si considera uno spazio continuo di distribuzioni che evolvono nel tempo. L'intero processo è descritto da una SDE. Invertendo il processo è possibile definire una SDE *reverse-time* che permette di trasformare sequenze di dati randomici in dati che si adattano alla distribuzione del training set.



Forward SDE e Reverse SDE [10]

SBGM-SDE - Caratteristiche

- **Campionamento flessibile**: è possibile utilizzare qualunque algoritmo general-purpose per la risoluzione di SDE in modo da effettuare il campionamento. In aggiunta, è possibile utilizzare ulteriori metodi specifici come i predictor-corrector samplers.
- **Generazione Controllabile**: è possibile modulare il processo di generazione dei campioni, condizionando il processo tramite informazioni non disponibili durante l'addestramento. Questo permette l'applicazione degli SBGM-SDE per task come la generazione class-conditional, image inpainting e colorizzazione senza la necessità di effettuare più addestramenti.
- **Framework Unificato**: il metodo analizzato fornisce una modalità unificata per esplorare e adattare diverse SDE per migliorare le prestazioni degli SBGM. I metodi definiti da SMLD e DDPM possono essere visti come specializzazione di questo framework tramite la discretizzazione di due distinte SDE.

SBGM-SDE – Generazione del rumore e dei campioni

Il processo di diffusione è modellato come la soluzione di una SDE:

$$dx = f(\mathbf{x}, t)dt + g(t)d\mathbf{w}$$

L'opposto di un processo di diffusione è anch'esso un processo di diffusione che procede indietro nel tempo, descritto da una *reverse-time* SDE:

$$dx = [f(\mathbf{x}, t) - g(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x})]dt + g(t)d\overline{\mathbf{w}}.$$

SBGM-SDE - Stima degli score

Lo score di una distribuzione può essere stimato addestrando un modello score-based e time-dependent $s_\theta(x, t)$:

$$\theta^* = \underset{\theta}{\operatorname{argmin}}\, \mathbb{E}_t\{\lambda(t)\mathbb{E}_{\mathbf{x}(0)}\mathbb{E}_{\mathbf{x}(t)|\mathbf{x}(0)}[||s_\theta(\mathbf{x}(t), t) - \nabla_{\mathbf{x}(t)} \log p_{0t}(\mathbf{x}(t)|\mathbf{x}(0))||_2^2]\}$$

Il modello s_θ può essere usato per costruire la reverse-time SDE che serve a generare i campioni. A questo scopo possono essere usati diversi risolutori general-purpose.

Esperimenti

Generazione

Addestramento e Generazione attraverso i Diffusion Models descritti, con valutazione oggettiva dei campioni tramite FID.

Similarity Retrieval

Valutazione qualitativa dei campioni generati, tramite confronto con le immagini più simili presenti nel dataset di riferimento.

Il dataset considerato è **CelebA**, composto da immagini di volti di celebrità.

- Per ***DDPM*** e ***DDIM*** è stata considerata la versione *CelebA-HQ [8]*, composta da 30.000 immagini 256x256. Questi modelli sono stati analizzati sia addestrando da zero, che considerando modelli pre-addestrati;
- Per ***SBGM-SDE*** è stata considerata sempre la versione *CelebA-HQ [8]*. Sono stati analizzati solo considerando un modello pre-addestrato;
- Per **NCSN** è stata considerata una versione di *CelebA [9]* composta da 202.599 immagini 178x218. Questi sono stati analizzati considerando un modello pre-addestrato.

Per DDPM e DDIM è stato possibile definire un confronto oggettivo.

Fréchet Inception Distance (FID)

Per confrontare le prestazioni ottenute dai diversi modelli si è scelto di utilizzare la **Fréchet Inception Distance** (*FID*) in quanto tipicamente impiegata per valutare le prestazioni di modelli generativi. La *FID* permette di valutare sia la fedeltà dei campioni generati rispetto alla distribuzione dei dati reali, sia la diversità all'interno dei campioni generati.

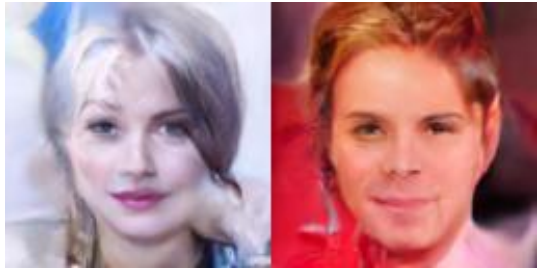
La definizione della *FID* si basa sulla nozione di **distanza di Fréchet**, utilizzata per misurare la somiglianza tra due curve.

A causa della scarsa potenza computazionale a disposizione, nei diversi esperimenti la *FID* è stata calcolata mettendo a confronto le istanze del dataset con un centinaio di campioni generati.

$$d^2((m, C), (m_w, C_w)) = ||m - m_w||_2^2 + Tr(C + C_w - 2(CC_w)^{1/2})$$

Addestramento - DDPM e DDIM

Dataset: CelebA-HQ ridotto a 3000 immagini di dimensioni 128x128
Modelli: DDPM e DDIM addestrati per 50 epoche
Architettura: U-Net con layer di normalizzazione di gruppo e attention layer
Iperparametri: variance schedule lineare e basata sul coseno
Passi di inferenza: 1000 per DDPM e 50 per DDIM



Campioni DDPM lineare

Campioni DDPM coseno

Campioni DDIM lineare

Campioni DDIM coseno

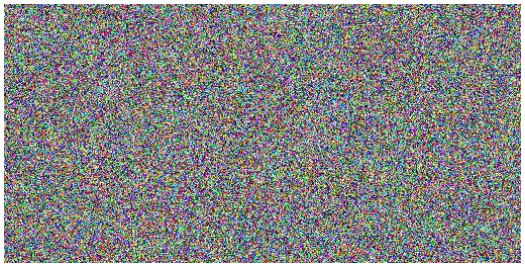
Addestramento - DDPM e DDIM

Modello	FID	Tempo addestramento	Tempo campionamento
DDPM linear	**99.802**	138 minuti	80 minuti
DDPM cosine	115.659	140 minuti	75 minuti
DDIM linear	103.322	127 minuti	**4 minuti**
DDIM cosine	113.433	**125 minuti**	**4 minuti**

- I modelli addestrati con la variance schedule lineare hanno ottenuto una FID migliore rispetto a quelli addestrati con la cosine schedule.
- A parità di variance schedule, le differenze tra DDPM e DDIM non sono significative. Con la linear schedule DDPM ha ottenuto una FID leggermente migliore. Con la cosine schedule è stata invece DDIM a ottenere una FID leggermente migliore.
- I DDIM hanno registrato un minor tempo di addestramento e di campionamento rispetto ai DDPM. Tuttavia il campionamento dei DDIM è avvenuto utilizzando solo 50 passi di inferenza rispetto ai 1000 passi utilizzati per i DDPM.

Generazione – DDPM vs DDIM

Dataset: CelebA-HQ con immagini di dimensioni 256x256
Modelli: DDPM e DDIM
Architettura: U-Net con layer di normalizzazione di gruppo e attention layer
Iperparametri: variance schedule lineare
Passi di inferenza: 50 e 1000 sia per DDPM che per DDIM



Campioni DDPM 50 passi

Campioni DDPM 1000 passi

Campioni DDIM 50 passi

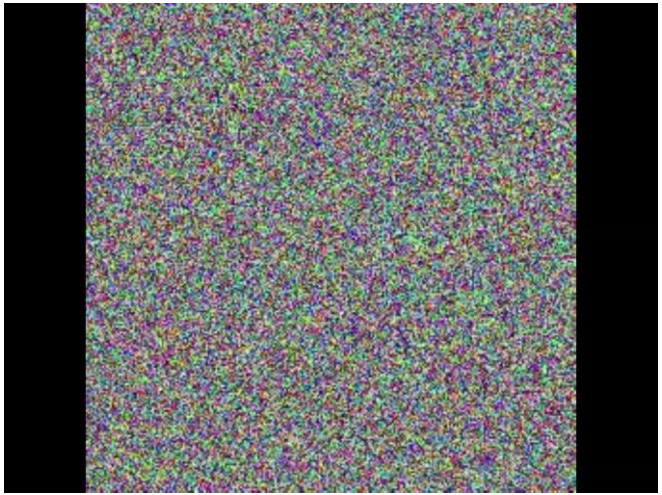
Campioni DDIM 1000 passi

Generazione – DDPM vs DDIM

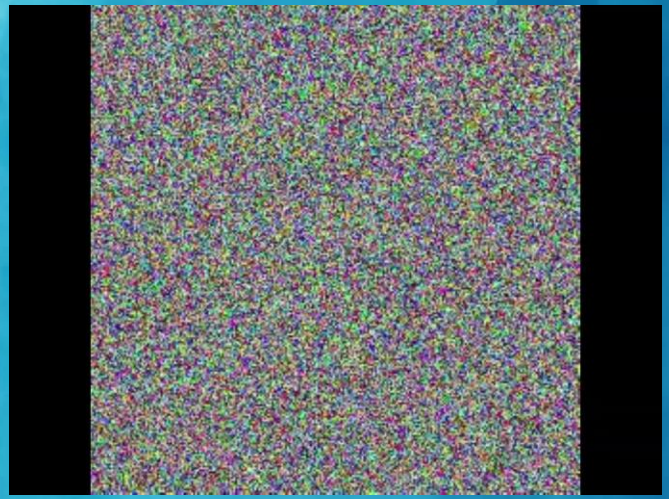
Modello	Passi	FID	Tempo campionamento
DDPM	50	459.211	**10 minuti**
DDPM	1000	**77.659**	213 minuti
DDIM	50	81.006	**10 minuti**
DDIM	1000	94.843	213 minuti

- DDPM con 1000 passi di inferenza ha ottenuto la FID migliore. Ma la FID registrata da DDIM con 50 passi di inferenza è solo di poco superiore.
- DDPM con 50 passi di inferenza ha prodotto immagini simili al rumore Gaussiano puro.
- DDIM con 1000 passi di inferenza ha registrato una FID peggiore rispetto a DDIM con 50 passi di inferenza.
- Naturalmente il campionamento con 50 passi di inferenza è più veloce di quello con 1000 passi di inferenza. E questo sia per DDPM che per DDIM.

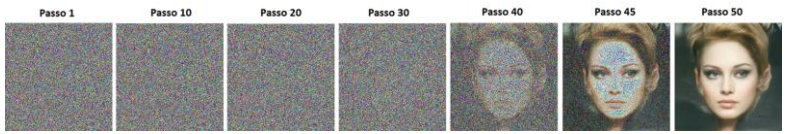
Generazione DDPM vs DDIM

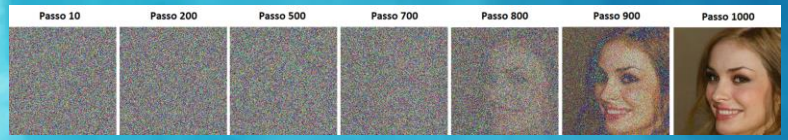


Video generazione DDIM 50 passi



Video generazione DDPM 1000 passi





Addestramento - NCSN

Dataset: CelebA
Modello: NCSN pre-addestrato considerando 200.000 iterazioni totali
Architettura: U-Net con Instance Normalization e convoluzione Dilated
Iperparametri: sequenza di 10 livelli di rumore, a partire da 1 fino a 0.01
Passi di inferenza: 1000, 100 per ogni applicazione della dinamica di Langevin

Campioni NCSN.

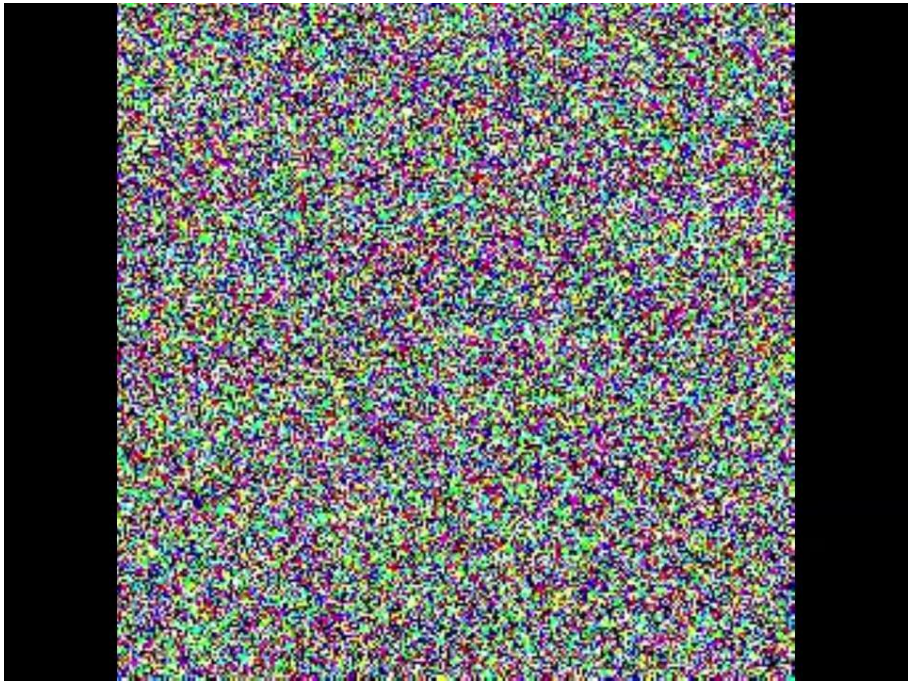
Sequenza campioni intermedi NCSN.

Generazione – NCSN

- **Tempo Inferenza**: 100 secondi;
- **FID**: 110,163

La NCSN analizzata permette di generare campioni con un tempo di inferenza molto basso, proprio perché le immagini generate sono relativamente piccole, 32x32.
Il valore di FID ottenuto è abbastanza alto. L'esperimento è stato proposto soprattutto con lo scopo di mostrare gli effettivi miglioramenti introdotti da DDPM e SBGM-SDE.

Generazione - SBGM-SDE



Modello: pre-addestrato da Google, pipeline disponibile su Hugging Face
Architettura: U-Net con dimensione dei campioni 256x256
Passi di inferenza: 2000
FID (100 immagini): 63,514

Generazione - SBGM-SDE

Alta qualità, ma con un grande costo computazionale. Per produrre una singola immagine 256x256 è necessario attendere 12 minuti.

Similarity Retrieval

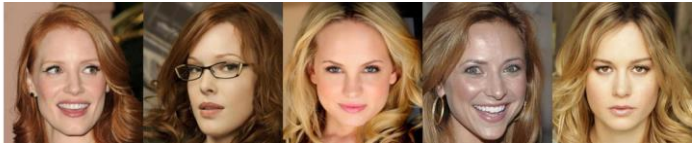
L'obiettivo di questo esperimento è stato verificare che i modelli addestrati non si fossero limitati a campionare immagini memorizzate dal dataset reale. La valutazione della similarity retrieval è stata puramente soggettiva.

L'esperimento è stato condotto utilizzando l'ultimo layer della rete Inception-V3 pre-addestrata su ImageNet per estrarre vettori di feature sia dal dataset di dati reali sia dalle immagini generate. Dopodiché si è proceduto a calcolare la distanza tra le feature delle immagini generate e le feature di ognuna delle immagini del dataset. Per calcolare questa distanza è stato utilizzato un algoritmo K-Nearest Neighbor con K pari a 5. La distanza considerata è stata quella Euclidea.

DDPM

DDIM

SDE



NCSN

***Sinistra**: immagine generata. **Destra**: le cinque immagini del dataset più simili*

Conclusioni - Esperimenti

Per quanto riguarda gli esperimenti svolti, si sono tratte le seguenti conclusioni:

- I DDIM hanno registrato il miglior compromesso tra qualità dei campioni generati e tempo computazionale;
- I SBGM-SDE hanno prodotto i campioni di migliore qualità, al costo di un tempo computazionale molto elevato;

In questo lavoro sono state vagliate le potenzialità generative dei Diffusion Models analizzati. Tuttavia, questi modelli si prestano ad applicazioni più ampie e variegate. Tra queste vi sono, a titolo meramente esemplificativo:

- deblurring;
- image inpainting;
- image colorization;
- sintesi audio;
- denoising;

Conclusioni: Diffusion Models vs GAN

Vantaggi

- NO mode collapse;
- Meno sensibili all'ottimizzazione degli iperparametri;
- NO addestramento contraddittorio;
- Funzione obiettivo adatta al confronto;
- NO vanishing gradient.

Svantaggio

- Elevato tempo di inferenza.

Grazie per l'attenzione

Raffaele Cerizza
845512

Giacomo Savazzi
845372

Thomas Barbera
845538

Bibliografia

[1] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen. Hierarchical text-conditional image generation with clip latents, 2022.
[2] Grewal, J.K., Krzywinski, M. & Altman, N., Markov models—Markov chains. Nat Methods 16, 663–664 (2019).
[3] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. Advances in Neural Information Processing Systems 33, pages 6840–6851, 2020.
[4] T. Nathan Mundhenk, Rashmi Sundareswaraa, David R. Gerwe, and Yang Chen. High precision object segmentation and tracking for use in super resolution video reconstruction. Imaging and Applied Optics, 2011.
[5] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.
[6] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. arXiv preprint arXiv:2010.02502v4, 2020.
[7] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution, 2019.
[8] CelebA-HQ: https://huggingface.co/google/ncsnpp-celebahq-256
[9] CelebA: http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html
[10] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. arXiv preprint arXiv:2011.13456v2, 2020.