POLITECNICO DI TORINO

Master's Degree in Data Science and Engineering

Master's Degree Thesis

Evaluating the entropy of physical systems using diffusion models

Supervisors

Prof. Paolo GARZA

Prof. Pietro MICHIARDI

Candidate

Giulia CORNARA

October 2024

Abstract

This work focuses on the application of diffusion models, a very well-known type of generative artificial intelligence, to the mathematical and physical problem of the computation of entropy. The main goal of this work is to show that diffusion models can be used to compute entropy in a very efficient way, exploiting their recognized ability to learn the underlying data distrbution of a system.

To this scope, first of all a good part of the presented work has been dedicated to the study of similar methods present in literature, to understand in particular which datasets are used to benchmark the task of entropy computation. This analysis brought to the result that in the physics world the preferred choice is to focus on spin systems, such as the Ising and XY models, for which the entropy has been analytically computed. Hence, these systems have been studied and methods to create samples from them have been implemented to create the corresponding datasets on which to train the diffusion models.

Moreover, an in depth study of the history and development of diffusion models to understand their functioning and how to exploit their capabilities has been carried out, with particular attention to the data preparation and to what is called the score network. To implement this last function a transformer architecture has been chosen for its versatility and powerful computational properties, that respondend well to the necessities given by the structure of the data.

Diffusion models are a fairly recent technology, and to exploit their full potential it is important that all people in the scientific community, even if not specialized in machine learning or data science, see the great benefits that the application of this technology may bring. It was for this reason and to prove the applicability of this type of generative AI in a very wide variety of fields, that the choice was made to apply diffusion models to the task of entropy computation, where they had never been employed before.

Entropy computation is a well-known problem in science since this quantity can give a lot of information on the data one is treating, but unfortunately it is often unfeasible to analytically compute it for systems or datasets. Up to now, few attempts have been made to use deep learning techniques to estimate this quantity.

The objective of this thesis has been to implement and test the proposed method employing diffusion models: all the code developed for the present work, including the entropy computations, the scripts for dataset generation and visualization, are available at this GitHub repository.

Acknowledgements

Per prima cosa vorrei ringraziare la mia famiglia per il supporto che mi ha dato da sempre. Tutto ciò che sono lo devo a loro e all'amore incondizionato che mi hanno dato per tutta la vita. Vorrei ringraziarli perché mi hanno spronata a migliorarmi continuamente e a spingermi sempre più avanti, e questo è lo spirito con cui ho affrontato tutte le sfide che ho incontrato sul mio percorso, in particolare durante l'esperienza universitaria. Grazie per avermi sempre sostenuta in ogni mia scelta, anche se mi portava lontano, di aver assecondato il mio entusiasmo e la mia voglia di fare nuove esperienze e allo stesso tempo di avermi aiutata ad affrontare anche qualche difficoltà.

Vorrei poi ringraziare i professori Paolo Garza, Pietro Michiardi e Giulio Franzese, per il loro supporto e per avermi dato la possibilità di lavorare su questa tesi in Francia ampliando i miei orizzonti.

Ringrazio tutti i miei amici: le amiche di paese che sono con me da tutta la vita e nonostante la distanza rimangono sempre vicine, il gruppo degli amici di fisica che ho conosciuto durante la triennale e che porto sempre nel cuore, i compagni di magistrale con cui ho condiviso tanto e tutte le persone meravigliose che ho conosciuto ad EURECOM. Vorrei nominarvi tutti, ma sarebbe davvero lungo.

Farò però un'eccezione per Aurora, mia cara amica da più di dieci anni, che tra telefonate e vacanze insieme non ha mai mancato di rallegrarmi e sostenermi.

Questi anni al Politecnico sono stati duri a volte; non avrei mai potuto superarli senza tutte le meravigliose esperienze che ho vissuto nel frattempo, dagli aperitivi al parco, alle scoperte di nuovi luoghi, alle "biutiful dinners" e tutte le canzoni cantate con la chitarra fino a tarda notte. Anche durante le sessioni d'esame, siamo

riusciti a spezzare lo studio con umorismo e con la gioia di stare insieme. Gli ingegneri sono spesso descritti come poco capaci da un punto di vista sociale, ma io sento di aver imparato di più sulle persone e sullo stare insieme in questi anni che in qualsiasi altro periodo della mia vita.

Potrei scegliere di citare molte delle frasi che ci siamo ripetuti per farci forza anche nei momenti difficili: *"Se insisti e resisiti il sogno conquisti"* o *"La vita è troppo importante per essere presa sul serio"*. Buoni aforismi da tenere a mente per affrontare le sfide della vita; alla fine però scelgo la frase che abbiamo menzionato più raramente, ma che mi è rimasta scolpita dentro e che spero di non dimenticare mai.

"Tu sei chi scegli e cerchi di essere"
Il gigante di ferro (ma a me l'ha detta Carla)

Table of Contents

List of Tables

List of Figures

Acronyms

AI	Artificial Intelligence
BR	Best Response
CNN	Convolutional Neural Network
DiT	Diffusion Transformer
KL	Kullback-Leibler (Divergence)
MC	Markov Chain
MCMC	Markov Chain Monte Carlo
MI	Mutual Information

PBC	Periodic Boundary Conditions
SDE	Stochastic Differential Equation
ViT	Vision Transformer
VP-SDE	Variance-Preserving Stochastic Differential Equation

Chapter 1

Introduction

The present work is based on and contributes to the large quantity of papers and projects exploring the capabilities of diffusion models. Diffusion models are a modern class of generative models, meaning their task is to generate objects of interest; they are widely used nowadays thanks to their extraordinary capacities in the generation of any type of content, may this be high fidelity images, videos, audio contents or even of specific domains of interest. Due to the large success of these models and their great capabilities, it is very important to study their functioning, the possibilities that they offer and their implications.

In particular, the present work focuses on the application of diffusion models not specifically to content generation, but to entropy computation. It has in fact been shown that diffusion models can be used to compute important mathematical quantities such as mutual information, and following this path it is interesting to try to exploit them also for the computation of entropy, which is a very important physical problem often really difficult or impossible to solve analytically. In the following chapters we will show how a specific part of the diffusion model, i.e. the score network, can be used to learn a really important quantity relative to the data distribution, i.e. the score, and starting from this computations can be made to retrieve the entropy of a chosen system. In particular, the thesis is organized as follows:

- In Chapter 2 the evolution and the functioning of diffusion models will be explained, together with all the mathematical concepts necessary to understand the following chapters. This comprehends notions of measure theory and stochastic calculus, as well as all the mathematical quantities such as KL divergence and entropy that will be used. This chapter will also introduce how entropy estimation and diffusion models are linked.

- In Chapter 3 the physical systems for which the entropy will be computed will be introduced, as well as the methods to generate the necessary corresponding datasets. This chapter will also offer a deeper focus on the system design and explain the choices that were made regarding the model architecture.
- In Chapter 4 the results of the experiments conducted to test the method will be presented. These will not be limited to the final results, but will also extend to all the checks that were made to ensure that the system is working as expected.
- In Chapter 5 the conclusions that can be drawn from this work will be exposed, discussing what worked optimally and what could still be material of future research.

Chapter 2

Related works

2.1 Mathematical background

2.1.1 Measure theory

Definition 1 *Given a set X, let $P(X)$ represent its power set, i.e. the set of all possible subsets of X. Then, a subset $\Sigma \in P(X)$ is called a **σ-algebra** if and only if the three following conditions are satisfied:*

- $X \in \Sigma$
- *If $A \in \Sigma$, then $A^c \in \Sigma$*
- *If $A_1, A_2, ... \in \Sigma$, then $\bigcup_{i=1}^{\infty} A_i \in \Sigma$*

Definition 2 *Let X be a set, and Σ be a σ-algebra over X. A function $\mu : \Sigma \to [0, +\infty]$ is called a **measure** if it satisfies the following properties:*

- $\mu(\emptyset) = 0$
- *σ-additivity: if $A_1, A_2, ... \in \Sigma$ are pairwise disjoint, then $\mu(\bigcup_{i=1}^{\infty} A_i) = \sum_{i=1}^{\infty} \mu(A_i)$*

A pair (X, Σ) is called a **measurable space**, and the members of Σ are called **measurable sets**. Moreover, a triple (X, Σ, μ) is called a **measure space**.

Definition 3 *A **probability space** $(\Omega, \mathcal{F}, \mathbb{P})$ is a measure space where*

- *Ω is the sample space, i.e. the set of all possible outcomes*
- *The σ-algebra $\mathcal{F}$ is called event space, is the set of events, an event being a set of outcomes in the sample space*
- *$\mathbb{P}$ is called probability function and assignes to every event in the event space a probability between 0 and 1, satisfying $\mathbb{P}(\Omega) = 1$*

A very special kind of measure, beside the probability measure, is the **Lebesgue measure**, which is used to assign a measure to subset of high dimensional Eucledian n-spaces. Up to three dimensions it coincides with the standard measures of length, area and volume, but generalizes the concept for every n. Considering any interval $I = [a, b] \subseteq \mathbb{R}$, and its length $l(I) = b - a$, for any subset $E \subseteq \mathbb{R}$, first the Lebesgue outer measure $\lambda^*(E)$ is defined:

$$\lambda^*(E) = \inf \left\{ \sum_{i=1}^{\infty} l(I_i) : (I_i)_{i \in \mathbb{N}} \text{sequence of open intervals with} \quad E \subset \bigcup_{i=1}^{\infty} I_i \right\} \tag{2.1}$$

The generalization to higher dimensions is done as follows: given any rectangular cuboid C which is a Cartesian product of open intervals $C = I_1 \times I_2 \times ... \times I_n$, let $vol(C) = l(I_1) \times ... \times l(I_n)$ denote its volume. For any subset $E \subseteq \mathbb{R}^n$, the Lebesgue outer measure is defined as:

$$\lambda^*(E) = \inf \left\{ \sum_{i=1}^{\infty} vol(C_i) : (C_i)_{i \in \mathbb{N}} \text{sequence of cuboids with} \quad E \subset \bigcup_{i=1}^{\infty} C_i \right\} \tag{2.2}$$

Some sets E satisfy the condition that for every $A \subseteq \mathbb{R}$

$$\lambda^*(A) = \lambda^*(A \cap E) + \lambda^*(A \cap E^c) \tag{2.3}$$

such sets are said to be Lebesgue-measurable, and the Lebesgue measure λ is defined as the restriction of the Lebesgue outer measure to the set of Lebesgue-measurable sets, moreover, the sets of all such E is a σ-algebra.

Another concept that will be used in the future sections is the **Radon-Nikodym derivative**. A measurable space is said to be σ-finite if it can be written as a countable union of measurable sets with finite measure, meaning

$$X = \bigcup_{i=1}^{\infty} X_i \quad \mu(X_i) < \infty \tag{2.4}$$

The Euclidean space is σ-finite with respect to the Lebesgue measure, because as we saw it is the union of countably many cubes with finite volume, and this is a very important property that allows to define the Radon-Nikodym derivative, which is a way to define the derivative of a measure with respect to another one, when the two measures are absolutely continuous with respect to each other.

Theorem 1 (Radon-Nikodym Theorem) *Let (X, Σ, μ) be a σ-finite measure space, and let ν be another measure on the same space such that ν is absolutely continuous with respect to μ. Then, there exists a σ-measurable function $f : X \to [0, +\infty)$ such that for every $A \in \Sigma$:*

$$\nu(A) = \int_A f \, \mathrm{d}\mu \tag{2.5}$$

the function f satisfying the above equality is called the Radon-Nikodym derivative of ν with respect to μ, and it is denoted as $\dfrac{\mathrm{d}\nu}{\mathrm{d}\mu}$. It is analogous to the derivative in calculus since it describes the rate of change of density of one measure with respect to another.

2.1.2 Stochastic Calculus

Stochastic calculus is a branch of mathematics created to deal with stochastic processes; in particular we will work with stochastic differential equations. Such equations have the form:

$$\mathrm{d}X_t = f(t, X_t)\, \mathrm{d}t + g(t, X_t)\, \mathrm{d}W_t \tag{2.6}$$

where

- X_t is the random variable defining the state of the system at time t
- $f : \mathbb{R}^m \times [0, T] \to \mathbb{R}^m$ is called the **drift coefficient**
- $g : [0, T] \to \mathbb{R}^m$ is called the **diffusion coefficient**

- $\mathrm{d}W_t$ is a Brownian motion

Definition 4 *A* ***Brownian motion*** *is a stochastic process $W_t \in \mathbb{R}$ such that:*

- $W_0 = 0$
- *W_t has independent increments for non-overlapping time intervals*
- $W_t - W_s \sim N(0, t-s) \quad \textit{for} \quad 0 \leq s \leq t$
- *W_t has continuous paths*

The diffusion equation that we will be working with is a special case of stochastic differential equation, and it is given by the following form:

$$\mathrm{d}X_t = f(t, X_t)\,\mathrm{d}t + g(t)\,\mathrm{d}W_t \tag{2.7}$$

where we can see that the diffusion coefficient only depends on time.

Given this diffusion equation, an interesting problem, solved by [1], is to find the reverse-time SDE, which is the SDE that describes the reverse of the diffusion process, reverting time. It is found that this is given too by a stochastic differential equation

$$\mathrm{d}\hat{X}_t = -f(t, \hat{X}_t)\,\mathrm{d}t + g(t)\,\mathrm{d}\hat{W}_t \tag{2.8}$$

and the reverse drift coefficient can be computed as:

$$\hat{f}(t, X_t) = f(t, X_t) - g^2(t)\nabla_x \left[\log p(t, X_t)\right] \tag{2.9}$$

2.1.3 Markov Chains Monte Carlo

The algorithms used in the following sections to model physical systems that will give the datasets of interest are based on Markov chains. Hence an introduction to this topic and an analysis of its foundations are necessary to allow to understand how some of the algorithms that have been used to the scope of this thesis work.

A Markov chain is a stochastic process that satisfies the Markov property (also, *memoryless property*), which states that the conditional probability distribution of future states of the process depends only upon the present state and not on the sequence of events that preceeded it. In other words, the future is conditionally independent from the past, given the present; for each state one can exactly compute the transition probabilities to any other configuration

A discrete-time Markov chain with state space $\mathcal{X}$ is defined by a transition probability matrix P where the rows are labeled with the entries of $\mathcal{X}$ and $P_{ij} = \mathbb{P}(X_{n+1} = j | X_n = i)$. A property that naturally follows is that all elements in each row of P sum up to 1; it is said to be a *stochastic matrix*.

A Markov chain is also characterized by its initial distribution $\pi(0)$, which gives for each state the probability of it being the first one during the system evolution $\mathbb{P}(X_0 = i) = \pi_i(0)$. Once the transition matrix and the initial probability distribution have been specified, the probability distribution of the trajectories can be computed. From the definition of P it follows that, given the probability distribution of the states at a certain time t, $\pi(t+1) = P'\pi(t)$. From this recoursive formula, which is known as Kolmogorov equation, one can derive that at every step, the probability distribution of the state of the system is given by the vector

$$\pi'(t) = \pi'(0)P^t \tag{2.10}$$

For every stochastic matrix the probability distribution π such that

$$\pi = P'\pi \tag{2.11}$$

is called *invariant probability distribution*, and it is well-known in many field such as graph theory to have very important properties and applications. It naturally follows that if the initial probability distribution $\pi(0)$ is equal to the invariant one, then for every time instant the probability distribution of the states will remain unchanged, which is the reason for which π is also called *stationary distribution*. Any stochastic matrix P can always be thought of as the normalized weight matrix of some (in general, weighted and directed) graph $\mathcal{G}_P$. The existence of a stationary distribution is not guaranteed for every stochastic matrix, but it is for all the ones that are *irreducible* and *aperiodic*. A matrix is said to be irreducible if on the graph that it defines it is possible to reach any state from any other state in a finite number of steps, and aperiodic if the greatest common divisor of the lengths of all cycles in such graph is 1.

It can be shown that if P is an irreducible and aperiodic stochastic matrix and $\pi = P'\pi$ is its unique stationary probability vector, then if at any $t \geq 0$ the probability distribution vector of a Markov chain with transition probability matrix P is called $\pi(t)$, then

$$\lim_{t\to\infty} \pi(t) = \pi \tag{2.12}$$

no matter what the initial conditions $\pi(0)$ were. This is equivalent to saying that for a Markov chain $X(t)$ with transition probability matrix P, the marginal probability distribution of $X(t)$ converges to the stationary probability vector π as t grows large. This is a first step needed to prove the following results given by the *Ergodic theorem*, which will guarantee that the empirical frequency of visits of a Markov chain in any given state i converges to the stationary probability π_i of that state.

Theorem 2 *(Ergodic theorem) Let $X(t)$, $t = 0,1,...$ be a Markov chain with irreducible transition probability matrix P. Let $\pi = P'\pi$ be the unique invariant probability distribution of P. Then, for any function f such that $\mathbb{E}_\pi[|f(X)|] < \infty$ and for any arbitrary initial probability distribution:*

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} f(X(t)) = \mathbb{E}_\pi[f(X)] \tag{2.13}$$

with probability 1.

This theorem is fundamental in the applications of Markov Chain Monte Carlo (MCMC) statistical methods: the basic concept is that in many relevant applications, the state space X is very large and one is only interested in computing weighted averages $\mathbb{E}_\pi[f(X)]$ of observable functions $f : \mathcal{X} \to \mathbb{R}$ with respect to a probability distribution π, but given the size of the system the probability vector π is hard or practically impossible to compute explicitly. However, thanks to the result of the theorem, one can compute the quantities of interest by constructing a Markov chain with invariant distribution π and, for large enough t, use its samples to approximate the expected value of f.

Other very important concepts in understanding Markov Chain Monte Carlo that simulate physical systems that obey to empirical laws come from game theory and are best response dynamics and noisy best response dynamics. They can be modeled with Markov chains and have some properites that ensure convergence to the desired results. If one considers the simulation of a physical systems were many particles interact, this can be seen as a game where each particle is an agent that interacts with all the others or at least the ones in its neighborhood which it is connected with.

In game theory, a game is defined by players $\mathcal{V}$ and for each player i a set of possible actions $\mathcal{A}_i$, potentially infinite. The set of all possible states of the game,

$$\mathcal{X} = \prod_{i \in \mathcal{V}} \mathcal{A}_i$$

is called *configuration space.* Each vector $x \in \mathcal{X}$ is a possible configuration, an assignment to each player of a specific action.

Each player has also a utility function $u_i : \mathcal{X} \to \mathbb{R}$ that assigns to each configuration a real number that quantifies the reward that player i gets from that state. The players are considered rational elements that act to maximize their utility. The triple $(\mathcal{V}, \{\mathcal{A}_i\}, \{u_i\})$ is called a *strategic form game.* Given these elements, the best response (BR) function can be defined as

$$\mathcal{B}_i(x_{-i}) = \text{argmax}_{a_i \in \mathcal{A}_i} u_i(a_i, x_{-i}) \tag{2.14}$$

i.e. the set of rational action that player i would choose in order to maximize its utility if he knew the actions of all the other players ant that those would not change. A very important concept in game theory is the one of *Nash equilibrium*, which is a configuration in which all players are playing a best response to the others. It is said to be strict if for every player such best response is unique. The interpretation of a Nash equilibrium is that in such configuration no player has any strict incentive to unilaterally change its action, as its current utility is the maximum he can get given the actions chosen by every other player. However one must note that a Nash equilibrium could still be a sub-optimal choice for the players; it is possible that coordinated deviations of more than one player from their actions in a Nash equilibrium would lead to a higher utility for these players. A game theoretic approach is widely used in optimization problems, when one must optimize a target functional

$$\Phi : \prod_{i\in\mathcal{V}} \mathcal{A}_i \to \mathbb{R} \tag{2.15}$$

This functional, that could for example represent some physical property such as the energy of a certain system, can often be decomposed in a sum of local terms

$$\Phi(x) = \sum_{F\in\mathcal{F}} \Phi^F(x_{|F}) \tag{2.16}$$

where F is a family of subsets of $\mathcal{V}$ and Φ^F is a local functional that depends only on the actions of the players in F. By defining the utilities of the players as

$$u_i(x) = \sum_{F\ni x} \Phi^F(x_{|F}) \tag{2.17}$$

one can construct $(\mathcal{V}, \{\mathcal{A}_i\}, \{u_i\})$ which is a *potential game* with potential Φ where the variables of the systems are considered the players.

Now that we know how physical systems and optimization problems can be modeled as games, we can define what a potential game is and how the potential can be maximized with noisy best response dynamics.

A game $(\mathcal{V}, \{\mathcal{A}_i\}, \{u_i\})$ is said to be an *(exact) potential game* if there exists a function $\Phi : \mathcal{X} \to \mathbb{R}$ such that for any player i, given any two configurations $x, y \in \mathcal{X}$ that differ only for the action associated with player i, the following holds true:

$$u_i(y) - u_i(x) = \Phi(y) - \Phi(x) \tag{2.18}$$

A very important property of potential games is that the configurations that belong to the set of global maximum points of the potential are Nash equilibria.

Now one can talk about evolutionary dynamics given by the series of actions of all the players; in contexts where the game is used as a modeling tool to solve optimization problems, such dynamics can be interpreted as distributed algorithms.

The best response dynamics is a game-theoretic learning process defined by a Markov chain where at each time step a player is chosen and it updates is action choosing from a uniform distribution on the set corresponding to its best response to the present configuration; this gives the transition matrix of the Markov Chain, while the state space coincides with the configuration space $\mathcal{X}$ of the game.

The best response dynamics has a very useful property: given a finite ordinal potential game with configuration space $\mathcal{X}$ and set of Nash equilibria $\mathcal{N} \subseteq \mathcal{X}$, the best response dynamics $X(t)$ is such that, for every starting configuration $X(0)$ there exists a finite time $T > 0$ such that $X(t) \in \mathcal{N}$ for every $t \geq T$. This, however, does not ensure that this dynamics will find the points of maximum of the potential, because even if all of the global maxima are Nash equilibria, it is not true that all Nash equilibria are maxima of the potential; there may be some that are "locally" the best configuration available, thus not encouraging the agents to change their strategy, but overall represent sub-optimals configurations.

This problem is resolved with the addition of noise to original dynamics; there are various way to do this, one of which gives the *logit dynamics*, or *noisy best response dynamics*. This model as before selects randomly one player whose action will be updated, but now the transition matrix for every two configurations $x, y \in \mathcal{X}$ is given by

$$\Lambda_{x,y} = \begin{cases} \dfrac{e^{\eta u_i(y_i, x_{-i})}}{\sum_{a \in \mathcal{A}_i} e^{\eta u_i(a, x_{-i})}} & \text{if } x \text{ and } y \text{ differ only in entry } i \\ 0 & \text{otherwise} \end{cases} \tag{2.19}$$

as one can see in this case there is a certain probability that the agent that changes action will make a choice that is not belonging to its current best response set. The parameter η is a parameter whose inverse corresponds to a measure of the noise. If it is put to zero, thus meaning "infinite noise", the dipendence on the utility vanishes and the agent choses its action from a uniform distribution over all its possible ones, while as η grows the noise decreases and eventually disappears and each agent will chose from a uniform distribution over its best response set, returning to the case of the best response dynamics.

Theorem 3 *Suppose that $(\mathcal{V}, \{\mathcal{A}_i\}, \{u_i\})$ is a potential game with potential function Φ. Then, for every $\eta > 0$, the noisy best response dynamics is an irreducible reversible Markov chain with invariant probability distribution*

$$\pi(x) = \frac{e^{\eta\Phi(x)}}{Z(\eta)} \qquad \textit{where} \qquad Z(\eta) = \sum_{y\in\mathcal{X}} e^{\eta\Phi(y)} \tag{2.20}$$

for every configuration $x \in \mathcal{X}$.

One can notice that in the vanishing noise limit the stationary probability distribution π converges to a uniform probability over the set $\text{argmax}\{\Phi(x) : x \in \mathcal{A}^n\}$, which gives the global maximizers of the potential function. As said before, this set will belong to but not necessarily coincide with the set of Nash equilibria: the dynamics that has been defined is able to discriminate the Nash equilibria selecting just those maximizing the potential.

The combination of this result with the Ergodic theorem (Th. 2.1.3) gives a very powerful tool: in the limit of a long time, the noisy best response Markov chain will have spent almost surely almost all of its time in $\text{argmax}\{\Phi(x) : x \in \mathcal{A}^n\}$, This is at the basis of many Markov Chain Monte Carlo algorithms widely used in statistical physics, based on the *Glauber dynamics*, which is the name of the noisy best response dynamics when applied to a physical context. In this framework, the energy is the potential function that has to be minimized by finding an appropriate configurations of the agents, π is referred to as the Gibbs distribution, Z_η is the partition function, a well-known quantity that is linked to many physical propertis amog which entropy, and the noise $\frac{1}{\eta}$ is often interpreted as the temperature of the system. These results have shown how it is possible to simulate complex physical systems relying on the mathematical properties of Markov Chains and potential games; this offers a legit basis to justify the algorithms that will be used in the following sections, in particular the Metropolis-Hastings algorithm which, apart from slight modifications and the speed of covergence, is almost identical to the Glauber dynamics.

2.2 Mathematical quantities of interest

This work is focused, rather than on the capability of diffusion model to generate content, on the exploitation of their capacity to learn what is called the score function of the input data in order to derive other quantities of interest from it to extend their domain of application. For the following discussion, the reader must be familiar with concepts such as entropy, Kullback–Leibler (KL) divergence and mutual information.

2.2.1 Entropy

Entropy is a scientific concept that describes the disorder, randomness or uncertainty of a certain state. It has applications, among others, in thermodynamics, statistical mechanics and information theory. It is often a crucial quantity of a system that is important to characterize, because it plays a key role in phase transitions and in general in the evolution of a physical system. Unfortunately, it is often difficult to compute; analytical computation of entropy is possible only for rather simple systems. Traditional definitions of entropy in classic thermodynamics are strictly related to physical characteristics such as temperature and heat. In information theory the entropy of a random variable is the average level of uncertainty inherent to the variable's possible outcomes, and it is defined for a certain measure μ as:

$$H(\mu) = -\int \mathrm{d}\mu(x) \log \bar{\mu}(x) \tag{2.21}$$

where the relationship between the probability measure μ and the corresponding probability density $\bar{\mu}$ is $\mathrm{d}\mu(x) = \bar{\mu}(x)\,\mathrm{d}x$. This expression is the same as the statistical mechanics definition, that is also focused on the degree of disorder of physical systems. The entropy can be viewed as the expected value of what is called "surprise", or "information content", which is given by $\log(1/\bar{\mu}(x))$. Such quantity is defined in information theory to measure how much a certain content is informative. This starts from the idea that the greater the "surprise" that a certain measurement brings, the higher the information that it conveys. On the other hand, registering an event that is almost certain to happen is not surprising at all and thus does not convey any useful or additional information, for this reason the surprise/information of an event with probability one is zero.

2.2.2 KL-divergence

The KL-divergence, also called relative entropy, is defined between two measures to quantify their distance, how far or equal they are to each other, and is defined as:

$$D_{KL}(\mu^A||\mu^B) = \int \mathrm{d}\mu^A \log\left(\frac{\mathrm{d}\mu^A}{\mathrm{d}\mu^B}\right) \tag{2.22}$$

if the Radon-Nikodyim derivative $\frac{\mathrm{d}\mu^A}{\mathrm{d}\mu^B}$ exists.
Notice that this quantity is not symmetric and it measures how a probability measure μ^A is different from a reference probability measure μ^B; one interpretation of the KL divergence between these two quantities is the expected excess of surprise when μ^B is used to approximate μ^A.
Because this quantity is not symmetric, it is not a metric on the space of probability distributions, while, as the name explains, it is a divergence, meaning an asymmetric, generalized form of squared distance. Two really important properties of the KL-divergence are the fact that it is always non-negative and that it is equal to zero if and only if the two probability measures under consideration are equal to each other.

2.2.3 Mutual Information

The mutual information between the random variables A, B quantifies how much they are related and they can say about each other. To measure it, one can notice that this concept is strictly related to the independence between these two variables. In fact, it is clear that if two variables are independent from each other, observing data from one will not give additional information about the second, meaning that the shared information between the variables is zero. Given these considerations, the definition of the mutual information is the following:

$$\mathbf{I}(A, B) = D_{KL}(\left[\mu^A, \mu^B\right]||\mu^A\mu^B) \tag{2.23}$$

where $\left[\mu^A, \mu^B\right]$ is the joint measure.

2.2.4 Important mathematical relations

Now that entropy, KL-divergence and mutual information have been defined, some useful equivalences must be reported which will be used in the following sections and which unveil the strict relationship between the quantities that were just introduced.

In particular it is easy to derive from the definition of KL-divergence its relationship with the entropy, where the cross entropy $H(A, B)$ has been defined

$$\begin{aligned} D_{KL}(\mu^A||\mu^B) &= \int \mathrm{d}\mu^A \log\left(\frac{\mathrm{d}\mu^A}{\mathrm{d}\mu^B}\right) = \int \mathrm{d}\mu^A \log\left(\frac{1}{\bar{\mu}^B}\right) - \int \mathrm{d}\mu^A \log\left(\frac{1}{\bar{\mu}^A}\right) \\ &= H(A, B) - H(A) \end{aligned} \tag{2.24}$$

The following set of equations instead shows that the mutual information can be thought of as the reduction in entropy of one measure when the other one is known:

$$\begin{aligned} I(A, B) &= H(A) - H(A|B) \\ &= H(B) - H(B|A) \\ &= H(A) + H(B) - H(A, B) \\ &= H(A, B) - H(A|B) - H(B|A) \end{aligned} \tag{2.25}$$

Where the conditional entropy is $H(A|B) = \int H(A_y)\,\mathrm{d}\mu_B(y)$. Let us prove the first equivalence

$$\begin{aligned}
I(A,B) &= \int_{a\in A}\int_{b\in B} p_{(A,B)}(a,b)\log\frac{p_{(A,B)}(a,b)}{p_A(a)p_B(b)}\,\mathrm{d}a\,\mathrm{d}b = \\
&= \int_{a\in A}\int_{b\in B} p_{(A,B)}(a,b)\log\frac{p_{(A,B)}(a,b)}{p_B(b)}\,\mathrm{d}a\,\mathrm{d}b - \int_{a\in A}\int_{b\in B} p_{(A,B)}(a,b)\log p_A(a)\,\mathrm{d}a\,\mathrm{d}b = \\
&= \int_{a\in A}\int_{b\in B} p_B(b)p_{A|B=b}(a)\log p_{A|B=b}(a)\,\mathrm{d}a\,\mathrm{d}b - \int_{a\in A} p_A(a)\log p_A(a)\,\mathrm{d}a \\
&= \int_{b\in B} p_B(b)\int_{a\in A} p_{A|B=b}(a)\log p_{A|B=b}(a)\,\mathrm{d}a\,\mathrm{d}b - H(A) = \\
&= -\int_{b\in B} p_B(b)H(A|B=b)\,\mathrm{d}b + H(A) = \\
&= -H(A|B) + H(A) = H(A) - H(A|B)
\end{aligned} \tag{2.26}$$

2.3 Diffusion models and generative AI

The scope of diffusion model is to learn the distribution of the data space of interest, with the aim of being eventually capable of sampling from it, creating new, unseen data that although being artificially generated still may seem plausible. For this reason, they belong to the field of generative AI. Quite often, the generation is conditioned, meaning that the model learns to generate samples conditioned on some prompt that is given: in fact, for example, it is not only useful to train a model able to reproduce credible pictures, but it is of great importance that the models responds well to user's needs, for example if the goal is to have a generated picture of a man the conditioning signal may be the label "man" and the model would have to sample not from the whole distribution of the space of all possible pictures, but only from those depicting a man.

2.3.1 Generative models

Generative AI models are widespread and rapidly improving: they succeed in generating text, audios, video and any kind of data. They are statistical models that reproduce the joint distribution of an observable and a target variable, with the final scope of generating samples from the observable, potentially even conditioned on the target. With the advent of deep learning, deep generative models have arisen, unifying statistical generative models with deep neural networks, that require large quantities of data and resources to be trained but have showed to be able to model reliably intricate mathematical relationship within the data itself that would otherwise be unfeasible to work with. A few of the most famous example of architectures for deep generative models are variational autoencoders (VAEs) and generative adversarial networks(GANs).

Variational Autoencoders

These type of models use expectation-maximization to optimize a lower bound of the data likelihood, which is usually intractable. Variational autoencoders stemmed from the autoencoder architecture, which is very successful in performing its task of dimensionality reduction. This task is fundamental for efficiency reasons because, while maintaining the most important features of the data, allows to have significant reductions in computational time and resources necessary to process data with respect to dealing with high dimensionality objects. Variational autoencoders, with respect to their predecessors, have introduced modifications to the encoder-decoder structure that allow not only the embedding of the data, $\mathbf{x}$, into a latent space, with variables $\mathbf{z}$, while retaining the most important features, but also allow generation of new data from the original distribution. The key idea behind variational autoencoders is that, in theory, if the autoencoder architecture was able to organize data in the latent space in a meaningful and "smart" way, one could sample a point in the latent distribution and decode to generate all the possible data.

A variational autoencoder is an architecture composed of both an encoder and a decoder and is trained to minimize the reconstruction error between the encoded-decoded data and the initial data. However, differently from a standard autoencoder, instead of encoding an input as a single point it encodes data in the latent space as a distribution. This is done in order to introduce some regularization of the latent space, to avoid overfitting and ensure that the latent space has good properties that enable the generative process. To train the model, firstly the input is encoded as distribution over the latent space, then a point from this space is sampled from the regular distribution and lastly, when the sampled point is decoded, the reconstruction error can be computed and backpropagated through the network.

The task is hence to try to maximized the probability that the data x is given by the parmetrized probability distribution $p_\theta(x)$, which is assumed to have the form of a Gaussian which parameters are the mean and covariance matrix. The relationship with respect to the distribution in the latent space is given by

$$p_\theta(x) = \int_z p_\theta(x|z) p_\theta(z)\, \mathrm{d}z \tag{2.27}$$

where also $p_\theta(x|z)$, called the likelihood, which will be computed by what is called a probabilistic decoder, is taken to be a gaussian distribution. The computation of $p_\theta(z|x)$ being intractable, it has to be approximated with $q_\phi(z|x)$, which will be the task of the probabilistic encoder and allow to infer the latent variable from the actual data without doing any integrals. The required regularization of the latent space comes from the assumption that the prior $p_\theta(z)$ is equal to a standard gaussian.

Variational autoencoder jointly optimize the generative models parameters θ in order to minimize the distance between the input and the corresponding reconstructed output, as all autoencoders, and minimize the distance between $q_\phi(z|x)$ and $p_\theta(z|x)$. With this last objective in mind, one can express the distance between two distributions as the Kullback-Leibler divergence

$$\begin{aligned} D_{KL}(q_\phi(z|x), p_\theta(z|x)) &= \mathbb{E}_{q_\phi(z|x)}\left[\ln \frac{q_\phi(z|x)}{p_\theta(z|x)}\right] = \\ &= \mathbb{E}_{q_\phi(z|x)}\left[\ln \frac{q_\phi(z|x)p_\theta(X)}{p_\theta(x,z)}\right] = \\ &= \ln p_\theta(x) + \mathbb{E}_{q_\phi(z|x)}\left[\frac{q_\phi(z|x)}{p_\theta(x,z)}\right] \end{aligned} \tag{2.28}$$

Re-arranging the terms, the evidence lower bound (ELBO) can be defined

$$L_{\theta,\phi}(x) := \mathbb{E}_{q_\phi(z|x)}\left[\frac{p_\theta(x,z)}{q_\phi(z|x)}\right] = \ln p_\theta(x) - D_{KL}(q_\phi(z|x), p_\theta(z|x)) \tag{2.29}$$

the ELBO is the training objective that has to be maximized by the variational autoencoder, since it can be noticed that it allows to simoultaneously maximize the log-likelihood of the observed data and minimize the divergence between the approximate posterior $q_\phi(z|x)$ and the exact posterior $p_\theta(z|x)$. It is called this way because it can be noticed that it is a lower bound for the log-likelihood.

The maximization of the ELBO is made by gradient descent, but a slight problem appears: differentiating with respect to ϕ does not allow to obtain a good formulation of the error, because it appears in the probability distribution over which the expected value is taken. To bypass this difficulty, the reparametrization trick must be used: since the latent variable z follows a normal distribution it can be reparametrized as a function of another variable distributed as a standard gaussian, ϵ. Doing this allows to obtain the following unbiased estimator for the gradient:

$$\nabla_\phi \quad \mathbb{E}_{q_\phi(z|x)}\left[\ln \frac{p_\theta(x,z)}{q_\phi(z|x)}\right] = \mathbb{E}_\epsilon\left[\nabla_\phi \ln \frac{p_\theta(x,\mu_\phi(x)+L_\phi(x)\epsilon)}{q_\phi(\mu_\phi(x)+L_\phi(x)\epsilon|x)}\right] \tag{2.30}$$

where $\mu_\phi(x)$ and $L_\phi(x)$ are the parametrized mean and standard deviation of $q_\phi(z|x)$.

The considerations made up to now for the ELBO have been used also in the training of diffusion models and are mentioned in [2].

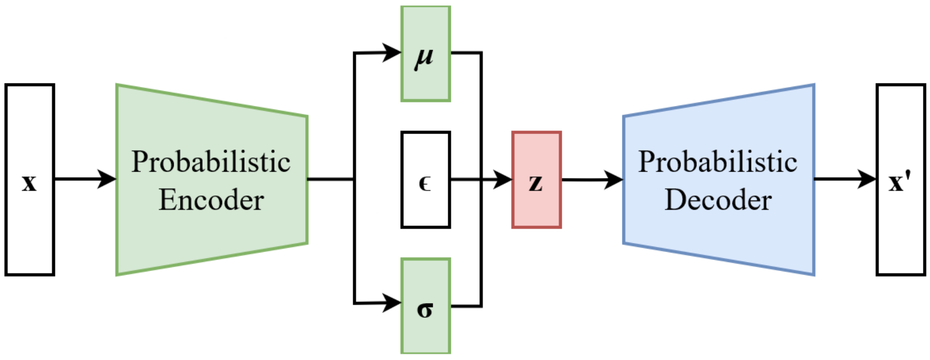


Figure 2.1: Schematic representation of reparametrized Variational autoencoder

2.4 Diffusion models evolution

Let us first give a brief overview of how diffusion models work: the core of every diffusion models is given by the **score function**,

$$s^{\mu}(x) \overset{\text{def}}{=} \nabla \log_x \bar{\nu}^{\mu} \tag{2.31}$$

where $\bar{\nu}^{\mu}$ is the probability distribution of the data in the domain of interest. The score function is often easier to reproduce than the original probability density function because it does not require the heavy calculations needed for normalization related to the partition function; nevertheless, once it is known, it allows to sample from the data distribution thanks to the Langevin dynamics. This has been the historical reason for which score-based generative models were thought.

The score is the true objective of the neural network that the diffusion model is based on, which is called **score network**. To learn and infer the score from the available data, different well-known architectures can be used: what has proven to be very effective in the past, and has been the standard score network for years is a simple U-Net, while recently [3] has introduced the transformer architecture in diffusion problems, creating the Diffusion Transformer (DiT).

To properly learn the score function it is important to be able to reproduce it also in low density regions; otherwise, since when the data dimensionality is high the initial sample to which the iterative Langevin dynamics is applied is highly likely to come from low density regions, the procedure will derail from the very beginning and will not be able to generate high quality and representative data.

To avoid this, the procedure that has been thought is to perturb the data with noise in such a way to populate also low density regions and improve the accuracy of the estimated scores. Overall diffusion models are given by the progressive corruption of the information given by the available data by the addition of noise, and the exploitation of neural network to learn the opposite process in order to be able to create new data samples starting from a prior that it is easy to sample from, for example, Gaussian noise.

Let us now describe all the steps of the development of diffusion models that are necessary to understand them nowadays.

2.4.1 Score matching

The goal of generative models is to learn the distribution that underlines the data one is interested in. This can be done by searching for a parametrized $p_\theta(x)$, but the problem is that it is only possible to approximate the probability density function up to a multiplicative constant $Z(\theta)$:

$$p_\theta(\xi) = \frac{1}{Z(\theta)} q_\theta(\xi) \tag{2.32}$$

where even with the functional form of q known it is often very difficult to compute

$$Z(\theta) = \int_{\xi \in \mathbb{R}^n} q_\theta(\xi) \, \mathrm{d}\xi \tag{2.33}$$

which is an integral that is practically impossible to compute analytically as soon as the data dimensionality n starts to grow. Hence training a model that parametrizes the probability distribution by maximizing the log-likelihood of the data becomes unfeasible for the presence of the normalizing constant $Z(\theta)$. For these reasons one can model the score function instead, which does not depend on the normalizing constant. Ideally the parametrized score function s_θ should be trained with objective function

$$\mathbb{E}_{p(x)} \left[\|\nabla_x \log p(x) - s_\theta(x)\|^2 \right] \tag{2.34}$$

but this is impossible because the true data score $\nabla_x \log p(x)$ is unknown. To circumvent this problem, various score-matching techniques have been developed over the years, one of which is denoising score matching. Considering a noise distribution $q_\sigma(\tilde{x}|x)$, the noise-corrupted data in given by $q_\sigma(\tilde{x}) = \int q_\sigma(\tilde{x}|x) p_d(x) dx$; the objective was then proved by [4] to be:

$$\frac{1}{2} \mathbb{E}_{q_\sigma(\tilde{x}|x) p_d(x)} \left[\|\nabla_x \log q_\sigma(\tilde{x}|x) - s_\theta(\tilde{x})\|^2 \right] \tag{2.35}$$

In this case, the smaller the noise scale, the better the denoising score approximates the ground-truth score.

Once the score-based model is trained a way to sample from the data distribution is given by the iterative procedure called **Langevin dynamics**, which allows to use the learned score to sample new data. This consists in starting from a sample from an arbitrary prior $x_0 \simeq \pi(X)$, that is usually taken to be a standard Gaussian distribution, and iterating the following:

$$x_{i+1} = x_i + \epsilon \nabla_x \log p(x) + \sqrt{2\epsilon} z_i \qquad i == ,1, ..., k \tag{2.36}$$

where the z_i are distributed following a standard Gaussian. If ϵ is sufficiently small and the process is carried out for a large number of steps the Markov Chain defined converges to a sample from the original distribution $p(x)$. Of course being $\nabla_x \log p(x)$ not directly available the quantity that is used in its place is the estimated score given by the trained score-network

2.4.2 Challenges of score-based generative modeling

Two main problems were found by [5] in the naive application of score-based generative modelling: the manifold hypothesis and low data density regions

- The manifold hypothesis: this is a well-known fact, often addressed in machine-learning, that holds true for many commonly employed datasets. The manifold hypothesis states that many real-world high-dimensional datasets actually lie along low-dimensional latent manifolds inside that high-dimensional space. The high-dimensional space is what is called the *ambient space.* This poses a series of difficulties for score-based generative models. The score, which requires to take a gradient with respect to the whole ambient space, is undefined when the data is confined in a low dimensional manifold. Moreover, score-matching techniques only return a consistent estimator for the score if the data distribution is supported on the whole space.

- Low density regions: In regions where the data density is low, there will not be a good enough quantity of data. This means that in such regions the score-matching objective function, that takes an expected value with respect to the available data, will not have a sufficient number of samples to be accurate in its estimate, hence the obtained scores will only be reliable in regions when the data density is high. When sampling with Langevin dynamics, the first noise sample will likely belong to a low-density region. Being that the score is not accurate there, this will derail the sampling dynamics from the very beginning. This makes the generation of high quality samples which are actually representative of the data distribution really difficult. Moreover, when the data is composed by a mixture of data distributions separated by a low density region, it has been shown that the Langevin dynamics is not able to recover exactly the relative weights of the original modes, thus not converging to the true distribution.

2.4.3 Noise conditional score networks

To overcome the difficulties exposed above, it has been found beneficial to perturb the data with random Gaussian noise. This allows to get over the manifold hypothesis, since the Gaussian noise distribution has support on the whole space, making score estimation consistent. Moreover, it allows to fill the low density regions to have more training signal. The larger the noise with which the original distribution is perturbed, the greater the previous benefits will be significant. Though on the other hand, large noise corrupts the data in a significant way and alters too much the original distribution. Using multiple noise levels gives a sequence of noise-perturbed distributions that converge to the true data one. Considering a number L of noise scales with increasing standard deviations $\sigma_1 < \sigma_2 < ... < \sigma_L$, for each of these the perturbed distribution is given by

$$p_{\sigma_i}(x) = \int p(y) N(x; y, \sigma_i^2 I) \,\mathrm{d}y \tag{2.37}$$

and samples from $p_{\sigma_i}(x)$ can be drawn by sampling x from $p(x)$ and z from $N(0, I)$ and computing $x + \sigma_i z$. Having these distributions for all noise scales, the score network can now be trained to learn the score function for each one of them, with the objective function that now is:

$$\sum_{i=1}^{L} \lambda(i)\mathbb{E}_{p_{\sigma_i}(x)}\left[\|\nabla_x \log p_{\sigma_i}(x) - s_\theta(x,i)\|^2\right] \tag{2.38}$$

where λ_i is a weighting function often chosen to be $\lambda_i = \sigma_i^2$. Once the model is trained and can reproduce scores relative to all noise scales, samples are generated starting from noise through iteratively applied Langevin dynamics with learned scores relative to decreasing noise scales: because the noise is decreasing this process is called **annealed Langevin dynamics**, which is explained in 1.

Algorithm 1 Annealed Langevin dynamics

```
 1: procedure ANNEALED LANGEVIN DYNAMICS($\{\sigma_i\}_{i=1}^{L}, \epsilon, T$)
 2:     $\tilde{x}_0 \sim N(0,1)$                                        ▷ Initialize $\tilde{x}_0$
 3:     for $i \leftarrow 1$ to $L$ do
 4:         $\alpha_i \leftarrow \epsilon \cdot \frac{\sigma_i^2}{\sigma_L^2}$
 5:         for $t \leftarrow 1$ to $T$ do
 6:             $z_t \sim N(0, I)$                      ▷ Draw noise from standard Gaussian
 7:             $\tilde{x}_t = \tilde{x}_{t-1} + \frac{\alpha_i}{2} \cdot s_\theta(\tilde{x}_{t-1}, \sigma_i) + \sqrt{\alpha_i} z_t$      ▷ Update sample
 8:         end for
 9:         $\tilde{x}_0 \leftarrow \tilde{x}_T$
10:     end for
11:     return $\tilde{x}_T$
12: end procedure
```

2.4.4 Diffusion probabilistic models

To model complex data-sets using highly flexible families of probability distributions in machine learning, [6] develop diffusion probabilistic models. Their main idea was to follow concepts studied in non-equilibrium statistical physics to systematically and slowly destroy structure in a data distribution through an iterative forward diffusion process, then learn a reverse diffusion process that restores structure in data.

To do so, a forward (or inference) diffusion process which converts any complex data distribution into a simple and tractable one was defined, and then a reversed generative diffusion process was trained to recover the original probability distribution. Calling the data distribution $q(x^{(0)})$ and the tractable and well-behaved one $\pi(y)$, the former is transformed into the latter by the repeated application of a of a Markov diffusion kernel $T_\pi(y'|y;\beta)$, where β is the diffusion rate

$$\pi(y) = \int \mathrm{d}y' T_\pi\left(y'|y;\beta\right)\pi\left(y'\right) \tag{2.39}$$

$$q\left(x^{(t)}|x^{(t-1)}\right) = T_\pi\left(x^{(t)}|x^{(t-1)};\beta_t\right) \tag{2.40}$$

The forward trajectory that defines the path of the evolving probability distribution, when starting from the data one and performing T steps of diffusion, is given by:

$$q\left(x^{(0,\ldots,T)}\right) = q\left(x^{(0)}\right)\prod_{t=1}^{T} q\left(x^{(t)}|x^{(t-1)}\right) \tag{2.41}$$

where $q\left(x^{(t)}|x^{(t-1)}\right)$ was either Gaussian or binomial diffusion; this allows to have a reversal of the diffusion process that has the identical functional form as the forward one. The generative distribution that has to be trained is given by:

$$p\left(x^{(T)}\right) = \pi\left(x^{(T)}\right) \tag{2.42}$$

$$p\left(x^{(0,\ldots,T)}\right) = p\left(x^{(T)}\right)\prod_{t=1}^{T} p\left(x^{(t-1)}|x^{(t)}\right) \tag{2.43}$$

The training of the model is carried out by maximizing the model log-likelihood, that is used to estimate the mean and covariance of the Gaussian diffusion kernel. Hence, the task of estimating a probability distribution has been reduced to the one of finding the functions defining mean and covariance of a sequence of Gaussians by means of regression.

2.4.5 Denoising Diffusion Probabilistic Models

Diffusion models and denoising score matching discussed up to now were linked by [2]. Diffusion models were different from other types of latent variable models because the approximate posterior $q(x_{1:T}|x_0)$, called forward process, is fixed to be a Markov chain that gradually adds Gaussian noise to the data according to a variance schedule $\beta_1, ..., \beta_T$. In [2] the choice was made not to learn these last parameters but to set them to constants instead. The whole diffusion process can then be re-parametrized in such a way that highlights the analogies with denoising autoencoders and Langevin dynamics: in fact, the regression task of learning the means of Gaussian kernels linking different steps of the diffusion process can be seen as learning the noise that was added to invert the diffusion process itself. The variational bound on log likelihood that must be trained then becomes:

$$L_{t-1} = \mathbb{E}_{x_0,\epsilon}\left[\frac{\beta_t^2}{2\sigma_t^2\alpha_t(1-\bar{\alpha}_t)}\left\|\epsilon - \epsilon_\theta\left(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t\right)\right\|^2\right] + C \tag{2.44}$$

where C is a constant independent of θ, $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{s=1}^{t}\alpha_s$. This objective resembles the one given by denoising score matching over multiple noise scales indexed by t.

2.4.6 Generative modelling through Stochastic Differential Equations

A very interesting result was given by [7] that allowed to reunite many of the previous approaches in a more general framework showing that the transition from images to uninformative noise can be done following the subsequent stochastic differential equation:

$$\begin{cases} \mathrm{d}X_t = f_t X_t \,\mathrm{d}t + g_t \,\mathrm{d}W_t \\ X_0 \sim \mu \end{cases} \tag{2.45}$$

where μ is the probability measure of the data, corresponding to the probability density $\bar{\mu}$ such that $\mathrm{d}\mu(x) = \bar{\mu}(x)\,\mathrm{d}x$. Given a certain f, g and initial condition, the path measure $\mathbb{P}^{\delta_x}$, i.e. the probability space of the system evolution, is determined. We can define ν_t^μ as the pushforward of the complete path measure onto time instant $t \in [0, T]$, where of course the initial probability measure is $\nu_0^\mu = \mu$. As time grows, the pushforward of the complete path measure tends to approximate a standard Gaussian. This can be seen as applying an infinite number of noise scales. The randomness and noise are given by the $\mathrm{d}W_t$ term is the standard Wiener process, i.e. a Browninan motion. What is called the *drift coefficient* of the equation is the vector-valued function f while g is a scalar function known as the diffusion coefficient.

A process like the one described by Eq. 2.45 progressively transforms data to noise; being able to reverse this process allows to start from noise and generate samples belonging to the data distribution. Luckily, the reverse of a diffusion process is also a diffusion process, given by the following reverse-time SDE:

$$\begin{cases} \mathrm{d}\hat{X}_t = f_t \hat{X}_t \, \mathrm{d}t - g_t^2 s_t^\mu(\hat{X}_t) \, \mathrm{d}t + g_t \, \mathrm{d}\hat{W}_t \\ \hat{X}_0 \sim \nu_T^\mu \end{cases} \tag{2.46}$$

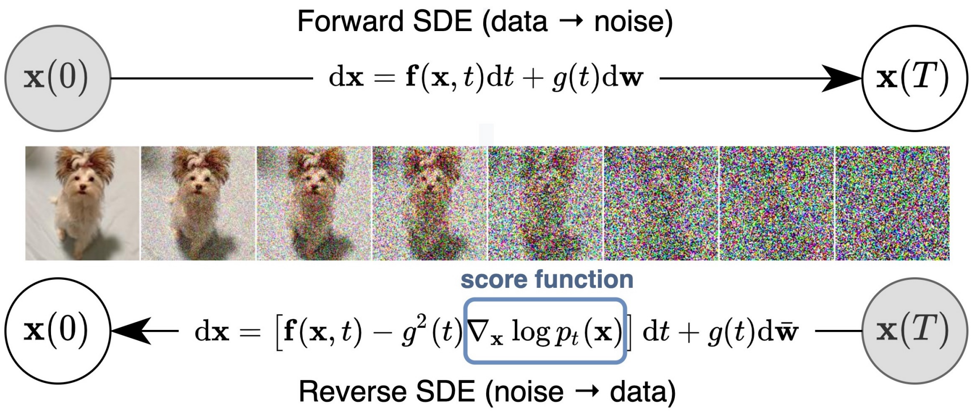
As it can be seen, the availability of the score is what makes it possible to create new samples following Eq.2.46 starting from noise; time in this framework is running from T to 0, hence $\mathrm{d}t$ is a negative timestep, and the starting point of the defined reverse diffusion process is $\nu_T^\mu \sim N(0, I)$, i.e the final distribution of the forward diffusion process. The score can be hence learned with score matching to be able to reverse the SDE and generate new samples. The overall process is schematized in Fig. 2.2.

These results offer a wide framework, and it reunites many different approaches. In practice, with any choice of appropriate f and g functions, a diffusion process can be set up. Of course, all processes defined by SDEs must be discretized to allow a practical implementation of the algorithm.

The perturbations that were introduced in the distribution by [5] using multiple noise scales $\{\sigma_i\}_{i=1}^N$ are equivalent to the following Markov chain from the data distribution to the prior one:

$$x_i = x_{i-1} + \sqrt{\sigma_i^2 - \sigma_{i-1}^2} z_{i-1} \qquad i = 1, ..., N \tag{2.47}$$

where $z_i \sim N(0, I)$. This can be shown to be a discretization of a diffusion



Forward SDE (data → noise)
$\mathbf{x}(0)$
$\mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\mathrm{d}t + g(t)\mathrm{d}\mathbf{w}$
$\mathbf{x}(T)$
score function
$\mathbf{x}(0)$
$\mathrm{d}\mathbf{x} = \left[\mathbf{f}(\mathbf{x}, t) - g^2(t)\boxed{\nabla_{\mathbf{x}} \log p_t(\mathbf{x})}\right] \mathrm{d}t + g(t)\mathrm{d}\bar{\mathbf{w}}$
$\mathbf{x}(T)$
Reverse SDE (noise → data)

Figure 2.2: Score-based generative model

process where f is the null function and

$$g_t = \sqrt{\frac{\mathrm{d}\left[\sigma^2(t)\right]}{\mathrm{d}t}} \tag{2.48}$$

where $\sigma(t)$ is now a continuous function reproducing the different noise scales. It can be demonstrated that a diffusion process with these choices has always exploding variance as $t \to \infty$, and for this reason the corresponding SDE was called Variance Exploding Stochastic Differential Equation (VE-SDE).

On the other hand the perturbation kernels of [2] give a discrete Markov chain described by:

$$x_i = \sqrt{1 - \beta_i} x_{i-1} + \sqrt{\beta_i} z_{i-1} \tag{2.49}$$

which can be shown to be a discretization of an SDE with

$$f_t = -\frac{1}{2}\beta(t)\mathbf{x} \qquad g_t = \sqrt{\beta(t)} \tag{2.50}$$

where $\beta(t)$ follows a linear schedule from β_{min} to β_{max}. This yields a process with fixed variance of one when the initial distribution has unit variance, and is hence called Variance Preserving Stochastic Differential Equation (VP-SDE). Thanks to this nice properties the VP-SDE is well-known and widely used; the present work always uses this choice to perform the diffusion process.

2.5 MINDE

The current work is a study and an extension to the MINDE model, presented in [8]. This work has shown how to exploit disintegration properties of the path measures that define the whole diffusion processes and the Girsanov theorem to obtain the KL divergence between two measures as a difference of score functions. The following disintegration properties hold for the forward diffusion process

$$\frac{\mathrm{d}\mathbb{P}^{\mu^A}}{\mathrm{d}\mathbb{P}^{\mu^B}}(\omega) = \frac{\mathrm{d}(\mathbb{P}^{\mu^A}\#_{\omega_0})}{\mathrm{d}(\mathbb{P}^{\mu^B}\#_{\omega_0})}(\omega)\frac{\mathrm{d}\mu^A(\omega_0)}{\mathrm{d}\mu^B(\omega_0)} = \frac{\mathrm{d}\mu^A(\omega_0)}{\mathrm{d}\mu^B(\omega_0)} \tag{2.51}$$

where we have conditioned the evolution of the process on the initial conditions, and for the backward diffusion process $\hat{\mathbb{P}}^{\mu}$:

$$\frac{\mathrm{d}\hat{\mathbb{P}}^{\mu^A}}{\mathrm{d}\hat{\mathbb{P}}^{\mu^B}}(\omega) = \frac{\mathrm{d}(\hat{\mathbb{P}}^{\mu^A}\#_{\omega_T})}{\mathrm{d}(\hat{\mathbb{P}}^{\mu^B}\#_{\omega_T})}(\omega)\frac{\mathrm{d}\nu_T^{\mu^A}(\omega_T)}{\mathrm{d}\nu_T^{\mu^B}(\omega_T)} \tag{2.52}$$

The first equation has as a consequence the fact that $D_{KL}\left[\mu^A \| \mu^B\right] = D_{KL}\left[\mathbb{P}^{\mu^A} \| \mathbb{P}^{\mu^A}\right]$ while from the second equation one can derive that

$$D_{KL}\left[\hat{\mathbb{P}}^{\mu^A}, \hat{\mathbb{P}}^{\mu^B}\right] = \mathbb{E}_{\hat{\mathbb{P}}^{\mu^A}}\left[\log\frac{\mathrm{d}(\hat{\mathbb{P}}^{\mu^A}\#_{\omega_T})}{\mathrm{d}(\hat{\mathbb{P}}^{\mu^B}\#_{\omega_T})}\right] + \mathbb{E}_{\hat{\mathbb{P}}^{\mu^A}}\left[\log\frac{\mathrm{d}\nu_T^{\mu^A}}{\mathrm{d}\nu_T^{\mu^B}}\right] \tag{2.53}$$

In the right hand side of this last equation, the first term can be evaluated by means of the Girsanov theorem, while the second one is the KL divergence between the results of the diffusion process of the two measures, which should both become standard gaussians; hence this term should always be in practice very small.

Unifying all these computation, with the additional information that the KL divergence between two path measures is invariant to time reversal, i.e. $D_{KL}\left[\hat{\mathbb{P}}^{\mu^A}, \hat{\mathbb{P}}^{\mu^B}\right] = D_{KL}\left[\mathbb{P}^{\mu^A}, \hat{\mathbb{P}}^{\mu^B}\right]$, one can derive the final expression for the KL divergence between two measures:

$$D_{KL}\left[\mu^A, \mu^B\right] = \mathbb{E}_{\mathbb{P}^{\mu^A}}\left[\int_0^T \frac{g_t^2}{2}\|s_t^{\mu^A}(X_t) - s_t^{\mu^B}(X_t)\|^2\,\mathrm{d}t\right] + D_{KL}\left[\nu_T^{\mu^A} \| \nu_T^{\mu^B}\right] \tag{2.54}$$

If the actual scores in the first term of the above formulas are substituted with the ones obtained by training a score network, an estimator for the KL divergence between two measures is obtained:

$$e(\mu^A,\mu^B)=\int_0^T \frac{g_t^2}{2}\mathbb{E}_{\nu_t^{\mu^A}}\left[\|\tilde{s}_t^{\mu^A}(X_t)-\tilde{s}_t^{\mu^B}(X_t)\|^2\right]\mathrm{d}t \tag{2.55}$$

Then, exploiting Eq.2.24 one can obtain the estimator for the entropy of a given measure. Since to evaluate the entropy of a given measure one must have the cross entropy and KL divergence with respect to a second measure, this last one can be taken to be a reference one such as a gaussian distribution with mean 0 and standard deviation σ, γ_σ, obtaining:

$$\begin{aligned}H(\mu^A)=H(\mu^A,\gamma_\sigma)-D_{KL}(\mu^A,\gamma_\sigma)=\frac{N}{2}\log\Big(2\pi\sigma^2\Big)+\frac{\mathbb{E}_{\mu^A}[X_0]}{2\sigma^2}-D_{KL}(\mu^A,\gamma_\sigma)\simeq\\ \simeq\frac{N}{2}\log\Big(2\pi\sigma^2\Big)+\frac{\mathbb{E}_{\mu^A}[X_0]}{2\sigma^2}-e(\mu^A,\gamma_\sigma)-D_{KL}\left[\nu_T^{\mu^A}\|\nu_T^{\gamma_\sigma}\right]\end{aligned} \tag{2.56}$$

where the last term, besides being almost null because both distribution tend to become standard gaussians, has known form equal to $\frac{N}{2}(\log(\chi_T)-1+\frac{1}{\chi_t})$, where $\chi_t=(k_t^2\sigma^2+k_t^2\int_0^t k_s^{-2}g_s^2\,\mathrm{d}s)$ with $k_t=e^{\int_0^t f_s\,\mathrm{d}s}$ and T is equal to the time corresponding to the ending of the diffusion process. This term, called diffusion time, determines a tradeoff between the efficiency of score matching, which is favoured by small diffusion times, and the fitness of the approximation of the final distribution being a standard gaussian, which instead is better for larger diffusion times. A study on this tradeoff and the selection of the optimal diffusion time has been done by [9], however in most implementations, including the current work, it is common to select this parameter to be equal to 1.

Chapter 3

Problem statement and system design

This work intends to show the applicability of diffusion models not only in what traditionally has been their principal field of employment, i.e. the generation of new data, but also in different domains of potential scientific interest. This is very important to get the whole scientific community interested in the last advancements in these topics and to allow the benefits coming from the newly-discovered capabilities of deep learning to permeate every field of knowledge, solving even problems that were deemed to be too complex to be analyzed. While it has already been shown that diffusion model can efficiently recover the mutual information between different distributions, it is interesting to assess this capabilities also in the estimation of entropy. Entropy is a crucial concept to be studied and in statistical physics is fundamental to properly describe many important processes such as phase transitions, pattern formation and protein folding. Since it is often unfeasible to evaluate the probabilities of all possible microstates of a systems, which would be required for the computation of this important physical quantity, the current ability of studying relevant thermodynamical properties of physical systems is to some extent limited. The exact, analytical computation of entropy is, in fact, limited to simple and weakly interacting systems; for all others, entropy computation quickly becomes unfeasible, with computation requirement that scale exponentially with system size. Some common methods used in the past to estimate entropy were to experimentally measure the temperature dependence of the specific heat ant then exploiting the relationship $S = S(0) + \int_0^T \frac{C_p}{T} dT$, or directly estimate the free energy of the systems. It is useful to remember the fundamental relationship that subsists between enthalpy, entropy and free energy

$$G = H - TS \tag{3.1}$$

In the above formula, G is the free energy, the state function whose changes represent the maximum amount of work that the system can perform in a process at constant temperature and determine, based on the sign, if a process is favourable or not. H is instead the enthalpy, that in turn is given by the sum of the internal energy of the system ant the product of its pressure and volume. T is the temperature of the systems and of course S is the entropy. This relationship is important because together all these quantities allow to completely describe the evolution of systems.

Given the importance of these quantities and the difficulties in their computation, the exploration of the possibilities offered by machine learning is not completely new, as it was already explored by some approaches among which [10] and [11]. However, the exploitation of diffusion models in particular in the estimation of entropy is a possibility that must be explored given also their previously proven efficacy in estimating relevant mathematical quantities.

3.1 Domains of Application

Machine learning algorithms can be exploited in a large variety of fields; with the current work, we want to prove the efficiency of diffusion models in offering precious insights in computationally difficult physical models. The first one that we are going to analyze is the Ising model.

3.1.1 Ising model

This mathematical framework allows the description of systems of magnetic dipole moments of atomic spins. The model consists of discrete variables representing the two possible values for the spins arranged in a lattice. For a general Ising model, the hamiltonian function that the system tends to minimize is:

$$H(\sigma) = -\sum_{\langle i,j \rangle} J_{ij}\sigma_i\sigma_j - \mu\sum_j h_j\sigma_j \tag{3.2}$$

where J_{ij} is the interaction between any two adjacent sites i and j and h is an external magnetic field. If the interaction between two spins is positive then the bond is said to be ferromagnetic, while if it is negative it is said to be antiferromagnetic. Very often a simplified version of Ising models is studied that has no external field interacting with the lattice and where it is assumed that all nearest neighbors have the same interaction strength J, that is commonly chosen to be equal to 1. The hamiltonian henceforward becomes the following:

$$H(\sigma) = -J\sum_{\langle i,j \rangle} \sigma_i\sigma_j \tag{3.3}$$

The peculiarity of the Ising model is that, even being relatively simple, in two ore more dimensions it starts to exhibit interesting physical properties: it is in fact characterized by a phase transition. At low temperatures the spins are in an ordered phase, where they are all aligned with each other, but as temperature increases they make a transition to a disordered phase.

The temperature of at which the phase transition can be identified, called *critical temperature*, is for the case under analysis $T_c \approx 2.269$. For this reason, we will study a range of temperatures that is developed around T_c and that completely captures the entropy evolution.

Ising models are often generated by means of a Markov chain Monte-Carlo simulation thanks to the well-known **Metropolis-Hastings** algorithm. Starting from a randomly initialized state, the evolution of the system to a configuration of minimum energy follows a Markov chain. A set of possible configuration is generated from the initial one, each of which differs just for one spin. A probability is assigned to each of the reachable states, and then once a move is selected, an acceptance probability is defined to determine wether the change in configuration will take place or not. Overall, this determines the transition probability matrix of a Markov chain. If these probabilities are defined appropriately, after a sufficiently large number of steps the system will converge to a configuration that is a minimum of the potential, i.e. minimizes the hamiltonian. The description of the Metropolis-Hastings algorithm can be found in 2.

Briefly, given a square lattice size of $N \times N$, the generation of samples for every temperature of interest is done in the following way: the lattice is initialized randomly, then at each iteration one spin is randomly flipped: if this action produces a decrease in the energy of the system it is always accepted and the configuration is updated consequentially. If instead the flipping of the spin produces an increment in the overall energy the action is accepted with probability $e^{-\dfrac{\Delta E}{T}}$. The algorithm stops when a stopping criteria such as a maximum number of iterations is met.

All the grids of spins that have been simulated for the purposes of this thesis have considered **periodic boundary conditions** (PBC), which are a type of boundary conditions very common in both computer simulations and mathematical models for physical systems. In fact, in molecular dynamics PBC are really useful when coupled with Monte Carlo modeling to calculate the bulk properties of fluids. The idea at the basis of this type of condition is to describe the world with a small unit cell, in the present case, our grid of spins of dimension $N \times N$; in small words we could say that ideally if an object moves across this unit cell and crosses one of its borders, it re-appears on the opposite side. Giving the definition formally in topological terms, the space made by periodic boundary conditions in two dimensions is equivalent to a torus. Up until now we have only considered two-dimensional systems of spins, with only nearest-neighbors interaction. In a two dimensional square lattice the nearest neighbors of each particle are four: the particle exactly above, the one below, the one to the left and that to the right. Of course, without periodic boundary conditions, this would mean that particles on the borders have a different behaviour than all the others, having a different number of nearest neighbors with which interact, and that contribute to the energy of the particle and the hamiltonian of the whole system. Since as previously said it is often interesting to study the bulk properties of physical systems, and develop different models to study separately the surface properties, with periodic boundary conditions the space is bent into a torus; in this way, the particle present in the top-left corner of the grid that represents the system under study will not only have as nearest neighbors the particles to its right and the particle below it, but will also have an interaction with the particles in the down-left corner and the top-right one. A graphical representation is given in Fig. 3.1

Algorithm 2 Metropolis - Hastings algorithm

```
procedure METROPOLIS($N, J, T, num\_iter$)
    $lattice \leftarrow rand(N \times N)$    ▷ A lattice of size N x N is initialized randomly
    $n \leftarrow 0$
    while $n < num\_iter$ do
        $n \leftarrow n + 1$    ▷ Update iteration

        ▷ Select a random spin of coordinates $i, j$
        $i \leftarrow rand$
        $j \leftarrow rand$

        $S \leftarrow$ the sum of the spin values of the neighbors of the selected spin
        $\Delta E \leftarrow 2 \cdot lattice(i, j) \cdot J \cdot S$ ▷ compute energy from flipping selected spin
        if $\Delta E < 0$ then
            $lattice(i, j) = -1 \cdot lattice(i, j)$
        else
            ▷ Accept "bad" move with a certain probability
            $r \leftarrow rand$
            if $r < e^{-\frac{\Delta E}{T}}$ then
                $lattice(i, j) = -1 \cdot lattice(i, j)$
            end if
        end if
    end while
    return lattice    ▷ The simulated configuration is returned
end procedure
```

Given its simplicity but at the same time its peculiarity, this system is often a

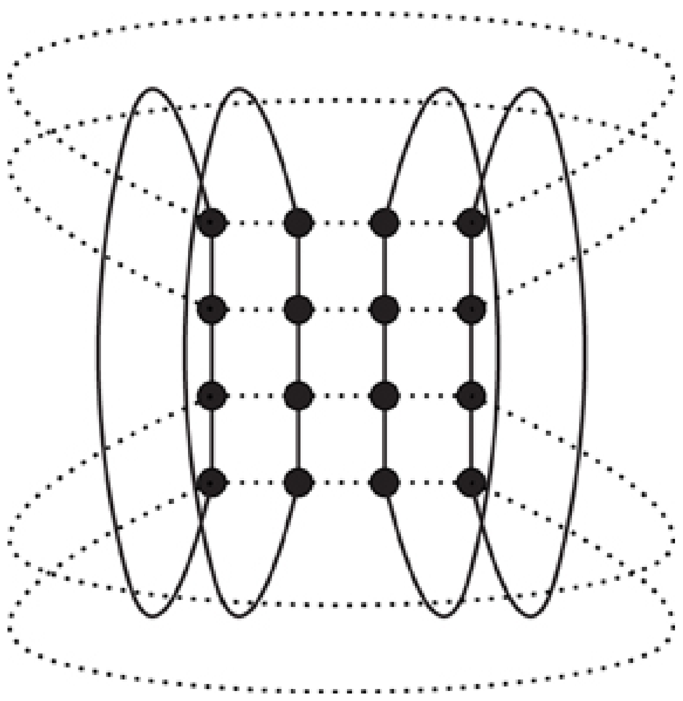


Figure 3.1: Graphical representation of periodic boundary conditions on a two-dimensional square lattice

starting point when benchmarking new algorithms and methods. The Ising model is in fact a very good testbed for the study of phase transitions and critical phenomena, and it is often used to test new methods; besides the already cited papers relative to entropy computation, other studies interested in applying machine learning to physical systems have used this dataset as the standard starting point, such as [12, 13].

3.1.2 XY model

The classical XY model, also called classical rotor (rotator) model, O(2) model or vortex model is a lattice model of statistical mechanics, and is often studied for its interesting thermodynamical properties ([14]). For every lattice site there is an assigned spin which is given by $\mathbf{s} = (cos(\theta), sin(\theta))$; in other words an angle is assigned to every lattice site.

Given a set of interaction terms between the different positions J_{ij} and an external field h_i, the hamiltonian of the system is given by

$$H(\sigma) = -\sum_{i \neq j} J_{ij} \cos(\theta_i - \theta_j) - \sum_j h_j \cos\theta_j \tag{3.4}$$

The most commonly studied case, which will be used also in all simulations, is the one in which $J_{ij} = 0$ except for couples of nearest neighbors. While in the 2D Ising model, where spins can take only values ± 1, there is a phase transition for which below a critical temperature there is long-range order, for the XY model things are different. In fact, for the Mermin-Wagner theorem, the same things would not be possible for the XY model, because this theorem states that in a number of dimension minor or equal than two, if the system is invariant under a continuous symmetry, no stable ordered phase can exist at finite temperatures. This model was however studied by Berezinskii, Kosterlitz and Thouless in the late 1970s and they showed that it nevertheless shows a quasi-long-range order. This leads to a particular phase transition that is said to be of infinite order and is named after the scientists (Berezinskii–Kosterlitz–Thouless transition, or, quite often, Kosterlitz-Thouless transition). The critical temperature for this transition has been found to be $T_c \approx 0.892$. Once again, the range of temperatures taken under analysis to study this model will develop around this value.

The model under analysis allow the creation of topological defects of the field $\theta(\mathbf{r})$ called vortices, which satisfy the Laplace equation $\nabla^2\theta(\mathbf{r}) = 0$. At low temperatures isolated vortices are unfavourable, and if they exists they are coupled in neural pairs that annihilate each other in the long distance effect while at high temperatures they are favourable. Overall the XY model shows the cited transition between a disordered high-temperature state to a quasi-ordered state below the critical temperature.

As the previous model, the XY model can be simulated by means of the Metropolis-Hastings algorithm, with the only difference that being now the spins continuous variables, once a site has been selected the flipping of its spin θ_i is done by choosing a random angle ϕ and evaluating the energy difference that one would have by substituting this new value to the previous angle assigned to the site, which is given by

$$\Delta E = -J \sum_{\theta_j \in neighbors(\theta_i)} \cos(\theta_i - \theta_j) - cos(\theta_j - \phi) \tag{3.5}$$

The acceptance probability is then given by the same formula as in the Ising model, that is $e^{-\Delta E/T}$.

However for simulating the XY model, another MCMC algorithm is often used ([15]), the Wolff algorithm, that is particularly efficient in this case. This algorithm, instead of performing local updates, relies on global ones flipping large clusters of spins at once. The idea behind the Wolff algorithm is to create a cluster of spins that are all quasi-parallel with respect to each other and are added to the cluster with a certain probability, then all the spins in the cluster are flipped at once. As explained in [16], this procedure is efficiently obtained by defining the following rule for a spin flip:

$$\theta_{i,j}^{new} = 2\phi - \theta_{i,j}^{old} \tag{3.6}$$

where an angle $\phi \in [0, \pi[$ has been randomly selected randomly.With this definition in mind, the algorithm procedure is the following: the cluster is initialized with a random site of the lattice, then as long as there are spin in the cluster that have never been updated, one of them is selected and flipped, then all its neighbors are examined and added to the cluster with a probability depending on their alignment with the spin that has just been flipped and on the temperature. Once all the spins in the cluster have been flipped, another cluster is randomly initialized and the procedure is repeated until a stopping criteria such as a maximum number of iterations is met. The algorithm is described in 3.

Algorithm 3 Wolff algorithm

```
 1: procedure WOLFF($N, J, T, num\_iter$)
 2:     ▷ to each lattice site assign a vector s.t. $\sigma_{ij} = (cos(\theta_{ij}), sin(\theta_{ij}))$
 3:     $lattice \leftarrow rand(N \times N)$    ▷ A lattice of size N x N is initialized randomly
    with a vector as described above
 4:     $n \leftarrow 0$
 5:     while $n < num\_iter$ do
 6:         $n \leftarrow n + 1$                                        ▷ Update iteration
 7:         ▷ Select a random spin of coordinates $i, j$
 8:         $i, j \leftarrow rand$
 9:         ▷ Add it to a new-forming cluster C
10:         $C \leftarrow (i, j)$
11:         for site ∈ C, site has never been updated do
12:             $\mathbf{r} \leftarrow$ random unitary vector
13:             ▷ Flip site spin
14:             $\sigma^{\mathbf{new}}_{\mathbf{i,j}} = 2\mathbf{r} - \sigma^{\mathbf{old}}_{\mathbf{i,j}}$
15:             for (k, l) in neighbors of (i,j) do
16:                 $\Delta E = -J\cos\Big(\sigma^{old}_{i,j} - \sigma_{k,l}\Big) + J\cos\Big(\sigma^{new}_{i,j} - \sigma_{k,l}\Big)$
17:                 if $\Delta E < 0$ then
18:                     $p \leftarrow rand$
19:                     $p_c \leftarrow 1 - e^{\frac{\Delta E}{T}}$
20:                     if $p < p_c$ then
21:                         $C \leftarrow C \cup (k, l)$
22:                     end if
23:                 end if
24:             end for
25:             ▷ All neighbors have been checked and possibly added to the cluster
26:         end for
27:         ▷ All sites in the cluster have been checked and flipped
28:     end while
29:     return $\sigma$                          ▷ The simulated configuration is returned
30: end procedure
```

3.2 Model architecture

The input of our network will be constituted by two "modalities": we are in fact considering two types of data, not only the spin configurations, but also their corresponding temperature. Up to now many methods explored in literature required the training of a different model for computing the entropy of the system at every different temperatures. In the present work the developed architecture is instead able to compute with just one model the entropy of the analyzed systems at every temperature. This has been done exploiting the capacity of diffusion model to learn conditional scores: indicating the conditional measure of the spin configuration given the temperature by μ^{A_T}

Then the formula to get the results for each temperature is:

$$\begin{aligned}
H(A|T) &= \int H(A_t)\,\mathrm{d}\mu^T(t) \\
&\simeq \frac{N}{2}\log\Big(2\pi\sigma^2\Big) + \mathbb{E}_{\mu^{T}(t)}\mathbb{E}_{\mu^{A_t}}\left[X_0^2\right] \\
&\quad - \int e(\mu^{A_t},\gamma_\sigma)\,\mathrm{d}\mu^T(t) - \frac{N}{2}\left(\log(\chi_T) - 1 + \frac{1}{\chi_T}\right) \\
&= \frac{N}{2}\log\Big(2\pi\sigma^2\Big) + \mathbb{E}_{\mu^{A}}\left[X_0^2\right] \\
&\quad - \int e(\mu^{A_t},\gamma_\sigma)\,\mathrm{d}\mu^T(t) - \frac{N}{2}\left(\log(\chi_T) - 1 + \frac{1}{\chi_T}\right)
\end{aligned} \tag{3.7}$$

where $e(\mu^{A_t},\gamma_\sigma)$ is the estimator for the KL divergence discussed in Sec. 2.5, for which computation it is necessary to learn the conditional score $\tilde{s}^{\mu^{A_T}}$, which is obtained by feeding the score network with both the diffused spin configurations and the information regarding the temperature of the original data. When using the VP-SDE, sampling various time instants t for the diffusion process, the perturbed configurations can be obtained by:

$$X_t(X_0,\epsilon) = \sqrt{\alpha_t}X_0 + \sqrt{1-\alpha_t}\epsilon \tag{3.8}$$

where $\alpha_t = \prod_{s=1}^{t}(1-\beta_s)$ and $\epsilon \sim \mathcal{N}(0{,}1)$. With the discussed input, the score network is then trained to reproduce the score. Traditionally as score network it has been common to use a U-Net architecture, but in this work to keep in count the fact that the input is two-dimensional and to avoid loosing information regarding the relative positions of the spins in the grid we are going to employ a diffusion transformer, whose structure is discussed in the following section.

3.2.1 Score network: Diffusion Transformer

Transformers are deep learning architectures introduced in 2017 by [17] that have had a huge success in the AI field because of their extraordinary capacities of understanding data and inferring information from it thanks to an attentive analysis of the context. Particularly useful in the analysis of sequential data, they allowed great performance improvements in the field of speech and text analysis, but are nowadays widely used almost everywhere.

They were born to process sequential data and overcome some limitations of the previous approaches, that often employed recurrent neural architectures which albeit being effective required a long training time, while transformers have no recurrent units.

The capacity of transformer to learn efficient representation for the data from which information can be extracted, is given by their excellent understanding of the context, which is implemented thanks to the attention mechanism.

The latter is a machine learning method that mimics the human skill of recognize which parts of a certain image or text are important to rapidly understand what one is facing and hence what should the attention be focused on. In practice, this is accomplished by assigning to each piece of data an attention score that is used to weight the input. The computation of this kind of weights can occur simulataneously, since unlike recurrent architectures the inputs do not have to be processed sequentially, hence the task can be parallelized, reducing the training time with respect to previous approaches.

Moreover, attention allows the exploitation of long-distance dependencies between data inputs, while recurrent architectures tended to be give more importance to nearer inputs. The main components of the attention mechanism are three sets of vectors called **queries**, **keys** and **values**. Each piece of an input sequence is equipped with its own triad of these vectors, all extracted for the input tokens by means of different linear projections. When having to determine how important each piece of input is related to all the others, the query and key vector are matched against each other to compute a score value given by their dot product. The score obtained in this way are passed through a softmax function and the final attention vector will be given by the weighted sum of this result with the value vectors. To summarize, the attention, or context vector, of a certain token with respect to all the others, given the query vector q of that token and the value and key vectors (v, k) of all tokens is given by:

$$e_{q,k_i} = q \cdot k_i$$

$$\alpha_{q,k_i} = softmax(e_q, k_i)$$

$$attention(q, K, V) = \sum_i \alpha_{q,k_i} v_{k_i}$$

Traditionally, in computer vision Convolutional Neural Networks (CNNs) have been the preferred tool to process visual data; this type of neural network are based on convolution operations and pooling layers that allow extracting information from images at different levels of resolution and recognize the most important graphical features.

Recently, with the advent of the transformer architecture, a shift in the paradigm has come through and the Vision Transformer (ViT) architecture has been created. Differently from CNNs, which leverage pixel-level data and local patterns, ViTs treat the input images as a sequence of patches and exploit the self-attention mechanism to learn meaningful relationship within images. Self-attention is a powerful mechanism thanks to which elements in a sequence are weighted based on their relevance. Since image data is two dimensional, this intrinsic structure has to be maintained to properly analyze it; CNNs used to do it thanks to the many convolutional layers. In the visual transformer instead, the image is first divided in patches of a fixed size that represent it at a local level; then, each of its patch is flattened into a single vector, which allows to create an analogy with the Natural Language Processing domain where the transformer architecture was created to process sequence of tokens. At this point, the dimensionality of the embedding can be optionally reduced by means of linear transformation. As in all transformer-based models, to keep track of the relative positions of the elements under analysis positional encodings are added; in this case it allows to keep track of the spatial relations that are present within the different patches that composed the image.

At this point the data enters the core part of the model which is equivalent to the encoder part of the original transformer architecture, composed by multiple layers of multi-head self-attention that compute the importance score of the various patches to understand which ones must be prioritized and multi-layer perceptron (MLP) blocks.

Starting from vision transformers, [3] created the Diffusion Transformer (DiT) to exploit the powerful transformer architecture also in the field of generative diffusion model, when traditionally only simpler U-Nets had been applied. This kind of neural network was composed by a contracting path (encoder) and an expanding one (decoder), characterized by residual connections. The fact that the DiT model is based on ViT and hence treats images as sequence of patches allows capturing dependencies between those patches thus allowing the understanding of long-range pixel-level interactions. During a diffusion process one does not only deal with images, but also with their progressively disrupted versions as the time of the diffusion process progresses. Summarizing, the important things to keep track of during the process are the corrupted image, the current time step and possibly a conditioning signal, that very often is a class label that can be passed to the network to make it learn how to generate samples specific to that class: these are the inputs of the diffusion transformer, that takes the role of the score network trying to predict the score function to reverse the diffusion process. The proposed architecture is graphically represented in Fig. 3.2 Note that it as been shown by [18] that the same network can be trained to learn both the conditional and marginal distribution: however, since for the present scope of the computation of entropy conditioned on the temperature only the first one was needed, this fact was not exploited in the implemented model.

As it can be seen, the first step is the division of the input image into patches that are processed by a linear embedding to have a result a sequence of tokens, each of the same dimension, as it happened in traditional transformers in NLP. The positional

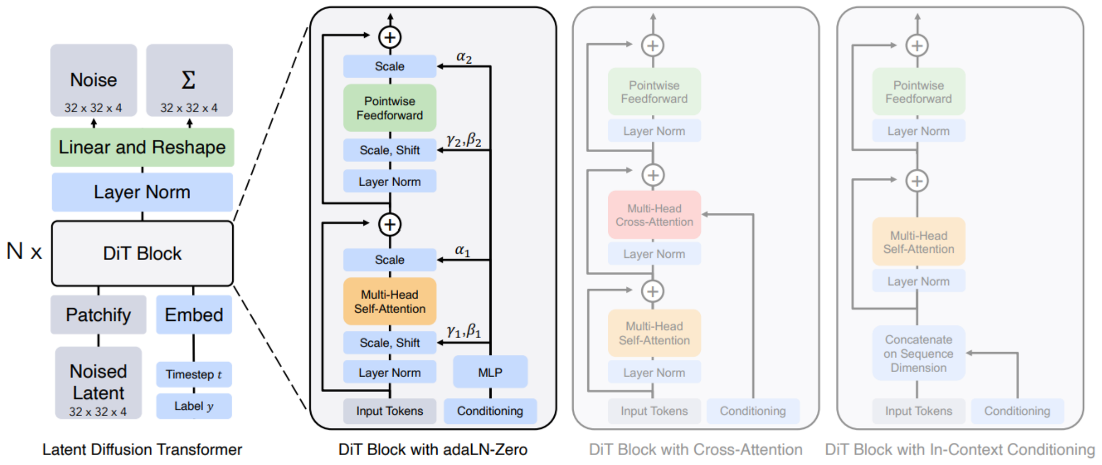


Figure 3.2: Diffusion Transformer Architecture

embedding for these tokens are created exploiting sine and cosine functions at various frequencies that allowing retaining information about the relative positions of the tokens with respect to each other. The time and conditioning information is embedded as well; they will be called t and c and together they form the conditioning that enters the true diffusion transformer block. The patchified images and this overall conditioning signal will hence undergo a series of identical diffusion transformer blocks. The transformer blocks, besides the attention mechanism, have adaptive normalization layers whose scale and shift parameters are regressed from the sum of the embedding vectors of t and c.

Since the target of the score network is to guess the random noise that was added to a certain image, the output must have exactly the same shape of the original spatial input, hence the sequence of image tokens must be decoded; this is done by means of a standard linear decoder.

Chapter 4

Experimental results

In this section the results of the many experiments carried out to assess the performances of the method explianed are presented. Besides the effective final results of the entropy computation, that will be presented in Sec. 4.2, it is really important to have a full comprehension of the intermediate steps of the process, to verify that everything is working correctly and as expected, which was done exploiting the experiments presented in Sec. 4.1.

During all the trainings necessary to perform the various experiments performed, the weights of the model were updated with an exponential moving average, which has been shown by [19] to be among the best practices for the training of the models. Then, after the stabilization of the value of the loss function, a sufficient number of iterations were waited before the checkpoints and the results were saved. This is done in order to have a smooth learning process being careful to retain only information relative to the weight that actually minimized the objective function.

The time integral necessary to obtain the KL divergence estimator in Eq. 2.55 is in general intractable and for this reason it is evaluated through Monte Carlo integration. The necessary time instants sampling can be carried out either in a uniform way over the diffusion interval or with importance sampling, following the techniques described by [20] and [21], that have been shown to reduce the variance of the estimations during the training and allow faster convergence.

4.1 Sanity checks

4.1.1 Noise prediction and content generation on spin systems

Since during the training phase the model is optimized trying to predict the noise that was added to each image, a first step in the evaluation of the model is to check if this is in fact working. One can hence compare the original image to the one obtained from the subtraction from the noisy version of the noise predicted by the neural network to evaluate their similarity. Such subtraction will be called "denoised" version. So, given the noisy image and its corresponding temperature, one wants to see if the noise predicted by the score network is actually similar to the one that was added to the real image to obtain the noisy version. This gives a first estimate of the capabilities of the model to learn the data distribution, expecially focusing on its ability to learn a conditional distribution over the set of possible labels, because of course even when given the same noisy image, the noise estimation must vary in function of the known temperature of the original image. The results for the Ising model can be visualized in Fig. 4.1, where black and white have been used to represent the two possible values for the spins (± 1).

The results seem satisfactory: it can be noticed that for temperature below the crtitical one, when the configuration results in spins that should all be aligned, with few exeptions that slowly grow in number with the temperature of the system, even if a lot of noise is added to the configuration the model is able to recover in a very good way the starting configuration: meaning that it has learned that when conditioned on that temperature the data should have aligned spins, and thus even if fed a mixed configuration it is able to understand that it consists almost entirely of added noise. One can remark that in the case under analysis the original image was almost completely recovered, but it has happened that when the noise added was very large and the temperature is lower than the crtitical the denoised image was the opposite of the real one (e.g. all spins down instead of all spins up). This is however perfectly reasonable, since the information about the temperature gives the model the knowledge that the spins should be aligned, but the mix of up and

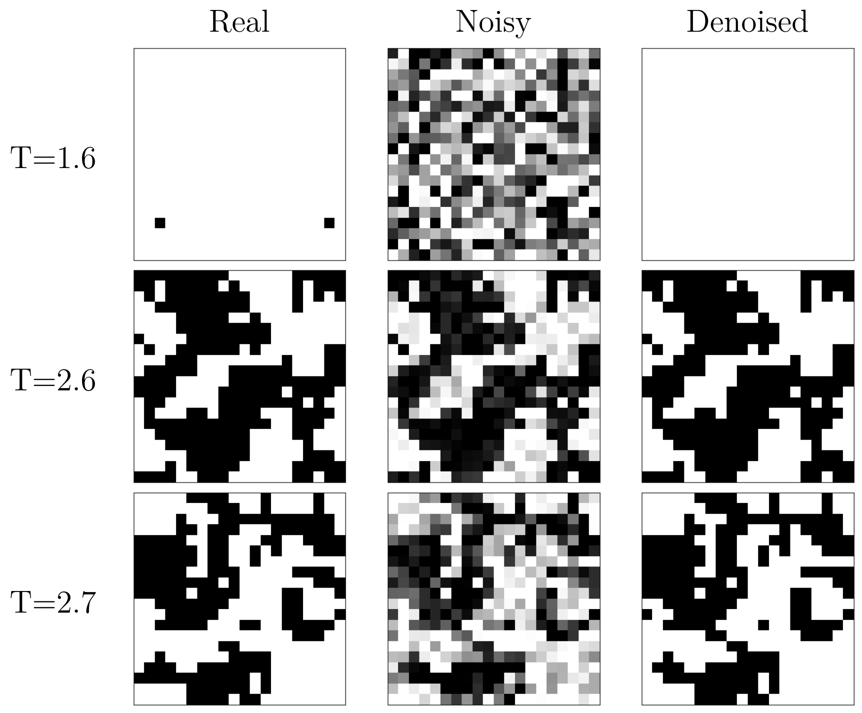


Figure 4.1: Real, noisy and denoised configurations of the Ising model

down spins in the noisy image does not allow to recover which of the two possible configurations the original image belonged to. As a further comment to the result of this experiment, one can notice that as expected for temperatures above the critical one present a large number of possibilities with mixed spins, that can be easily reconstructed in a very precise way.

After this trivial check, further ones must be carried out. Even if the scope of this work was not the generation of new data, still, the fact that diffusion model were built for this scope renders imperative to assess that everything is working as expected and that the trained model is in fact able of effectively generate consistent data, even if such data is not the ultimate target of the work.

To do so, a random subset of possible temperatures has been sampled, and for each of them, starting from a random configuration, the diffusion process was run to generate the corresponding data through a discretization of Eq. 2.46. This means that iteratively the network is fed with a noisy image and the temperature, and it computes the score, then a denoising step is performed to obtain the next

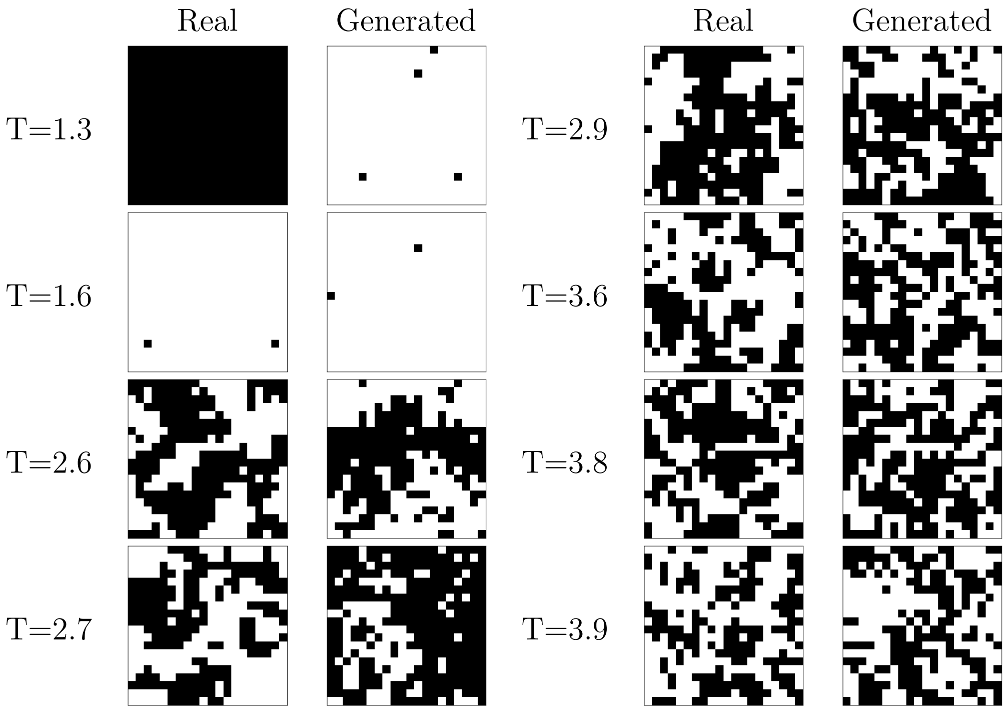


Figure 4.2: Real configurations of the Ising model at various temperature compared to the ones generated by the diffusion process

image in the sequence, until the diffusion process is completely reversed and a clean image has been generated. In Fig. 4.2 the results of this experiment are shown for the Ising model; we can see that the real data (i.e. the configurations obtained by MCMC simulations) and the ones produced by the generative model are very similar. At low temperatures almost all spins are aligned, whereas at temperatures above the critical one the spins start to assume all two possible values, firstly arranging in relatively big groups of parallel spins and then, as the temperature further increases, becoming very randomly mixed.

The same experiments and considerations of course can be carried out also for the XY model. In this case, spins can rotate on a plane, assuming continuous values, so to represent their value each spin in the 20×20 grid is represented by an arrow pointing in the direction of the angle assigned to that site. For ease of visualization, a color has been assigned to each possible value of angles according to a continuous and periodic color map, presentedi in Fig. 4.4.

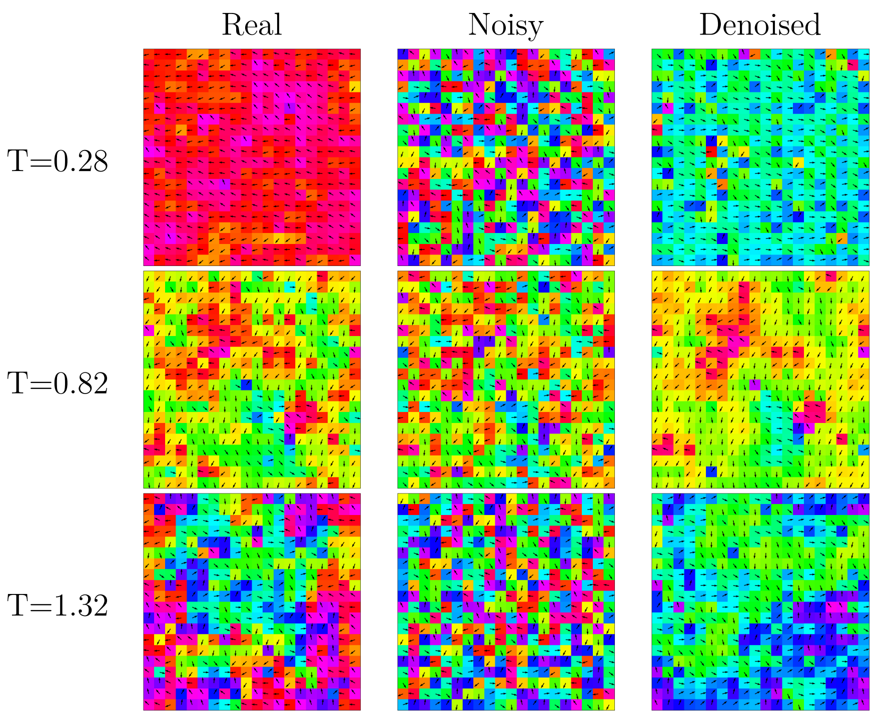


Figure 4.3: Real, noisy and denoised configurations of the XY model

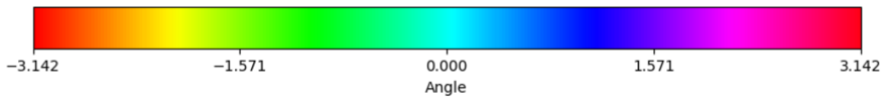
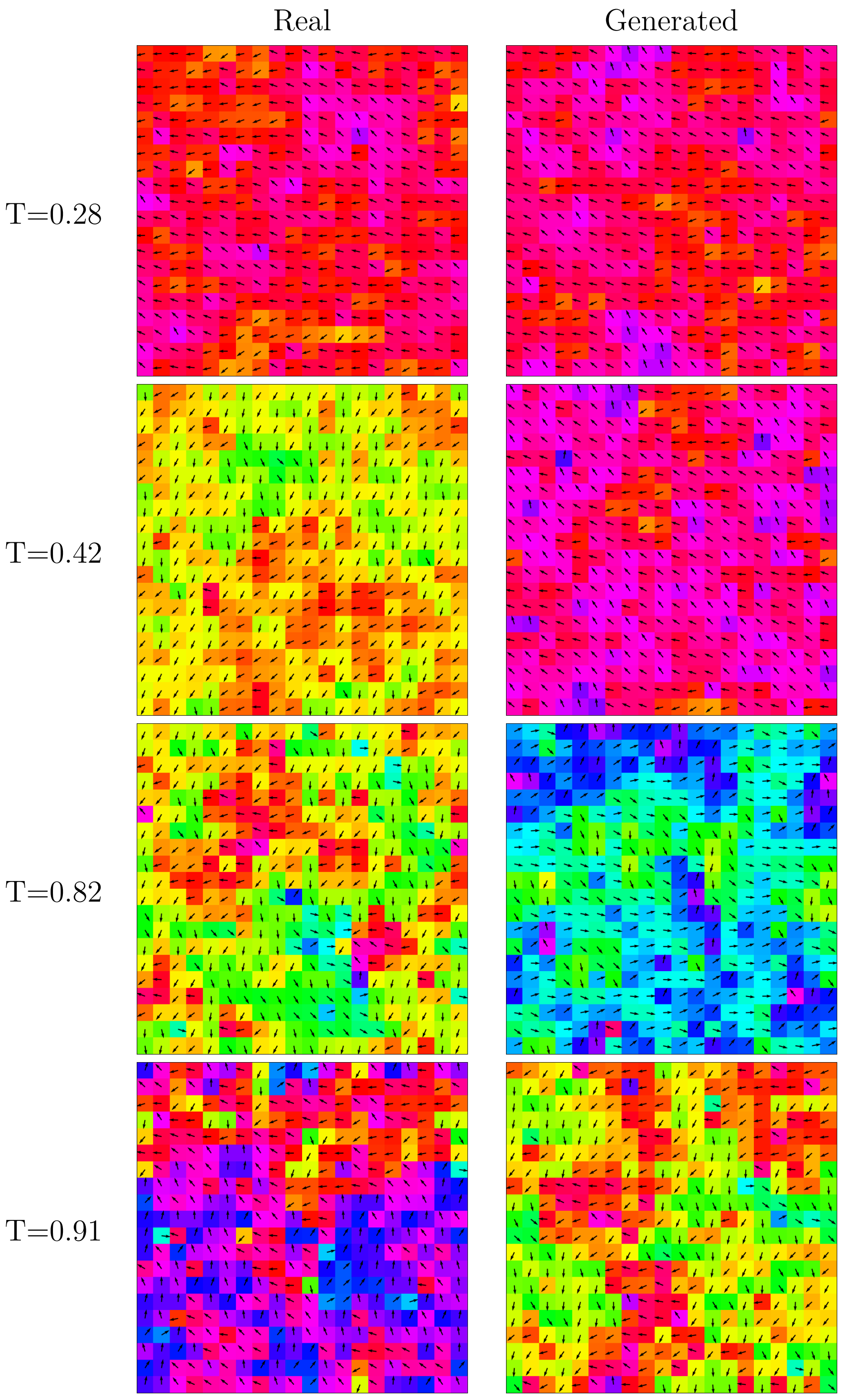


Figure 4.4: Color map used to represent the spin values of the XY model

In Fig. 4.3 it can be seen an example of what was referenced before: at a low temperature with all spins aligned if a lot of noise is added the system may fail in exacting predicting the noise, but the results are still understandable since the "denoised" image is a configuration with all spins aligned, even if not in the same direction of the original one. This happened because from the noisy image it was not possible to recover information about such direction. In the following row, around the critical temperature, we can see a case in which some noise, but not too much, has been added and the system perfectly manages to recover the original configuration. In the last row, once again a large quantity of noise was added making the original structure unrecognizable; the denoised image still is somehow similar to the original, even if at this high temperature one would expect a larger variety of spin values, as there were in the real configuration. Hence, to check that the model has learned the correct data distribution for every temperature, as before it is necessary to assess the generative capabilities of the diffusion process. This can be found in Fig. 4.5, where it can be seen also for high temperatures the generated spin configurations seem likely. As expected, for low temperatures the spins are all aligned, with more or less waves depending on how low the temperature is, and as the temperature increases the spins start to rotate more and become cahotic for the highest ones.


Real
Generated
T=0.28
T=0.42
T=0.82
T=0.91

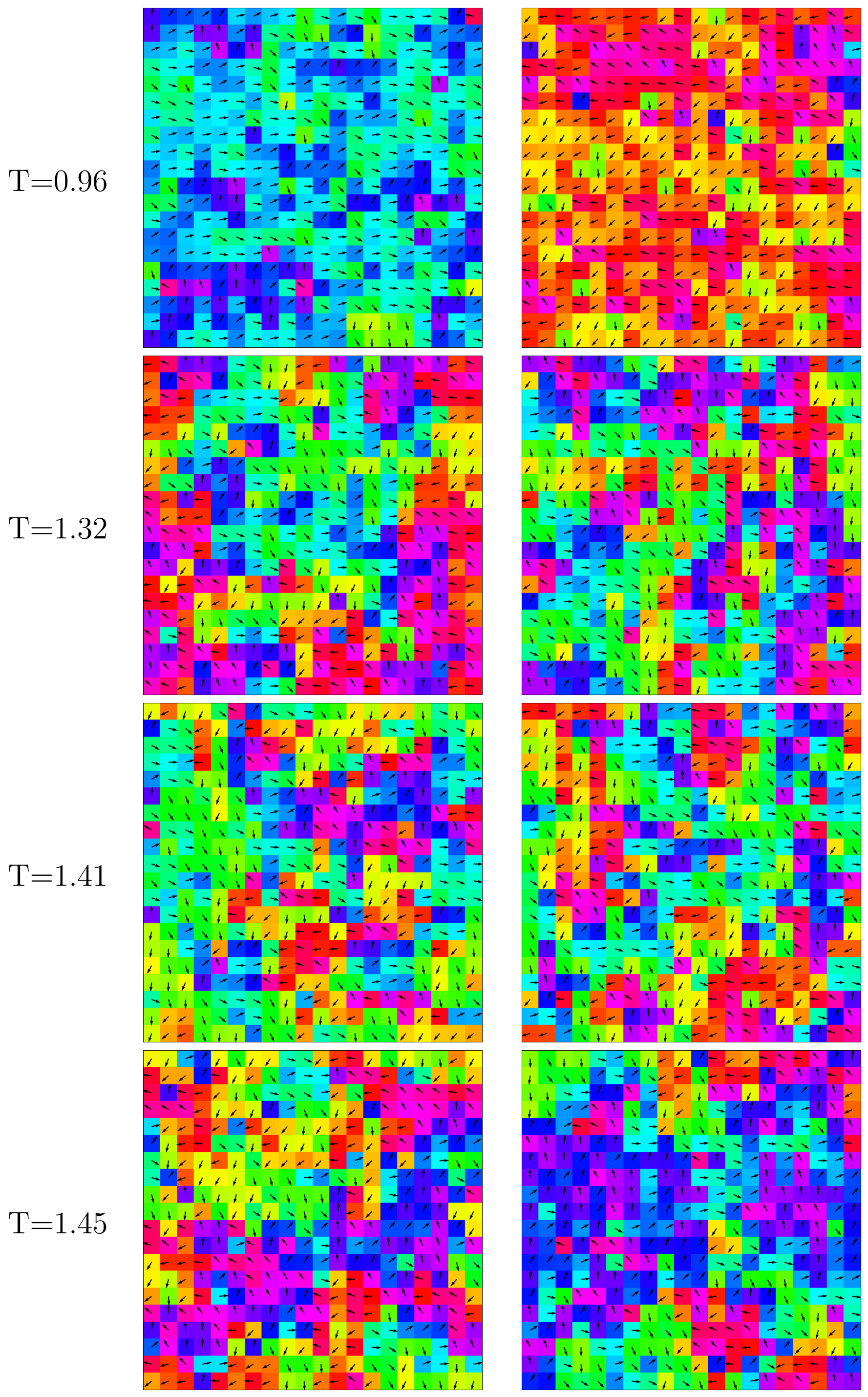


Figure 4.5: Real configurations of the XY model at various temperature compared to the ones generated by the diffusion process

4.1.2 Entropy computation on multivariate gaussians

Let us now move to the entropy computation, which is the core of the problem. In the ensemble of activities that have been carried out and constitute the material of the present work, it has found out that the model developed does not perform well on the generated dataset of the Ising model. An explaination to this behaviour was found in the fact that as explianed in Sec. 2.5 the estimator for the entropy that we are using is based on the estimate of the KL divergence of the probability measure of the data with respect to a gaussian distribution, but the results regarding this last qantity always assumed the measures to be absolutely continuous in order for the Radon-Nikodyim derivative $\frac{\mathrm{d}\mu^A}{\mathrm{d}\mu^B}$ to exist. But this is not the case for a discrete distribution, and since the spins in the Ising model can assume only values in a discrete set, the probability measure of the data would be formed by a sum of Dirac deltas, and the Radon-Nikodyim derivative and hence the KL divergence would not be defined.

For this reason, the focus was shifted on continuous datasets. As a first check, the efficiency of the method on a simple set of multivariate gaussian distributions was tested, since this is a common and relatively simple distribution for which the entropy can be computed analytically. Given a covariance matrix Σ and a mean vector μ that were generated randomly, the probability distribution is given by the formula:

$$p(x) = \frac{1}{(2\pi)^{d/2}\sqrt{\det(\Sigma)}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right) \tag{4.1}$$

and strarting from this, recalling the definition of entropy for a continuous random variable in Eq. 2.21 one can derive that the expression of the entropy is:

$$H = \frac{1}{2}\log\Big((2\pi e)^d \det(\Sigma)\Big) \tag{4.2}$$

For various possible dimensions of the multivariate gaussians, several distributions were created, chosing randomly the corresponding Σ_i and μ_i, and for each of these distributions 20000 possible configurations were sampled, each of them labeled with the corresponfing i and the ground truth entropy was computed according to the formula shown above. This last quantity was then compared to the result obtained by the application of the MINDE model when given as input the gaussian data, and the results are shown in Tab 4.1.

Dimension	i	Ground Truth	MINDE
9	0	2.8438	2.8228
	1	3.0631	3.1198
	2	2.7101	2.8784
	3	2.8453	2.9021
	4	3.0224	3.0959
16	0	3.1781	3.2369
	1	3.3124	3.3832
	2	3.1951	3.3134
	3	3.1086	3.3177
	4	3.3400	3.5103

Table 4.1: Entropy computation for multivariate gaussians

4.2 Entropy computation results on spin systems

Given the positive results on the datasets composed by multivariate gaussian distribution, one can now confidently move to the evaluation of the presented method on the XY model. One of the first things that was noticed was that, depending on the normalization of the data, a shift in the entropy result was observed. It is however a well-known fact in entropy computation that this quantity is invariant to translation but not to scaling: in a formal way, given a random variable X with probability distribution f, if the transformation $Y = \sigma X + c$ is applied then

$$H(Y) = H(X) + \log(\sigma) \tag{4.3}$$

It is a common practice in entropy computation to obtain what is called *excess entropy*, i.e. subtract to the total entropy of the system the entropy corresponding to an equilibrium configuration, which is the entropy of an ideal gas. In our case, this quantity corresponds to the entropy of a single spin. Since spins can rotate freely on the XY plane, their relative positions to each other will be determinated by the hamiltonian and behave accordingly to what described up to now, but when considering a single spin it will assume values following a uniform distribution over the range of possible angle values. The entropy of a uniform distribution is analytically known and equal to the logarithm of the amplitude of the interval over which it is defined. Subtracting this quantity also fixes the ambiguity of the shifting of the entropy when the data is rescaled. This happens because in this way if the data is scaled by a factor σ for the phoenomenon described before the entropy would increase by a factor $\log(\sigma)$, but now in the subtraction of the entropy of a single spin the interval over which the uniform distribution is defined is multiplied by σ as well, giving an overall contribution of $-\log(\sigma)$ that annihilates the shifting effect. Excess entropy is henforth invariant to scaling. For these reasons, given that the data were scaled between -1 and 1, a quantity equal to $\log(2)$ was subtracted to the results of the model in order to obtain the final result, which is shown in Fig. 4.6. To obtain the value depicted, since what was created through this work is an estimator for the entropy, several runs of the entropy computation were made and their results for each temperature were averaged. As it can be seen, the results are in perfect accordance to the ground truths and the errors are comparable with the best results present in literature, thus confirming the efficiency of the method for entropy computation and once more confirms the great capabilities of diffusion models to learn even complex data distributions.

Moreover, in Fig. 4.7 one can see that even in the Ising model case, the model, even if with less precision, is capable of correctly estimating the entropy variations with the temperature, even if it fails to retrieve the correct values. Since however entropy is often defined up to an additive constant, knowing that what is truly important for thermodynamic reasons are its variations, the results can still be considered satisfactory, and it can be seen that the phase transition, which is often the ultimate target of entropy estimations, is correctly identified.

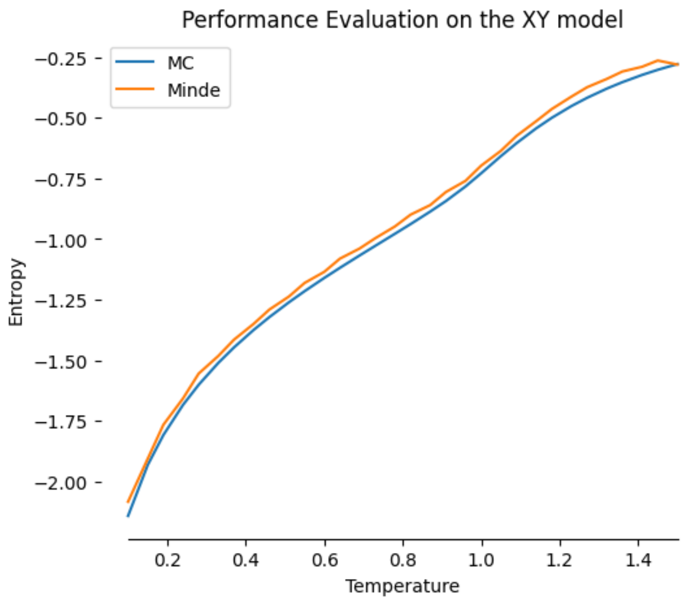


Figure 4.6: Evaluation of MINDE model on the XY dataset: entropy per temperature computed with MINDE compared to the ones obtained from Monte Carlo methods and taken as ground truth

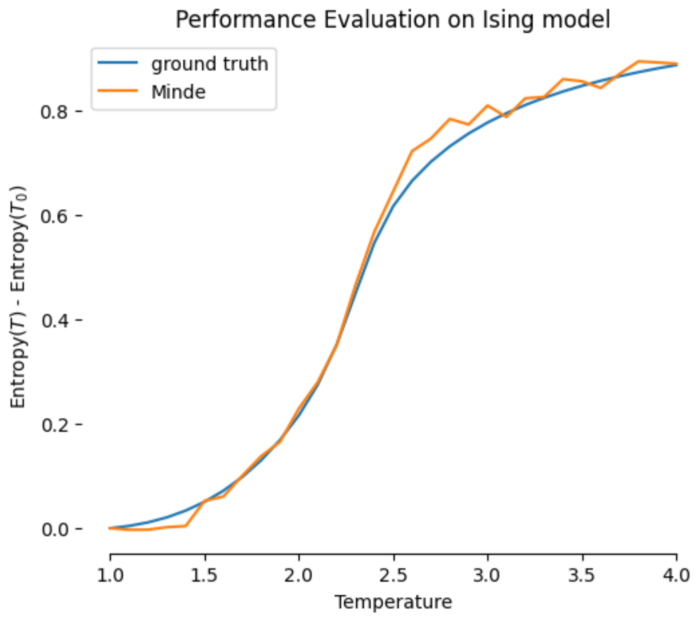


Figure 4.7: Evaluation of MINDE model on the Ising dataset: entropy variations per temperature computed with MINDE compared to the ones obtained from the analytical formula

Temperature	Entropy Ground Truth	MINDE
0.1	-2.1408	-2.0823
0.15	-1.9312	-1.9055
0.19	-1.8074	-1.7651
0.24	-1.6832	-1.6567
0.28	-1.6001	-1.5540
0.33	-1.5101	-1.4815
0.37	-1.4463	-1.4143
0.42	-1.3743	-1.3492
0.46	-1.3216	-1.2904
0.51	-1.2604	-1.2365
0.55	-1.2144	-1.1804
0.60	-1.1600	-1.1348
0.64	-1.1182	-1.0803
0.69	-1.0674	-1.0387
0.73	-1.0276	-0.9962
0.78	-0.9780	-0.9478
0.82	-0.9380	-0.8991
0.87	-0.8865	-0.8597
0.91	-0.8431	-0.8057
0.96	-0.7829	-0.7598
1.00	-0.7284	-0.6969
1.05	-0.6581	-0.6363
1.09	-0.6046	-0.5746
1.14	-0.5435	-0.5139
1.18	-0.4998	-0.4636
1.23	-0.4515	-0.4125
1.27	-0.4174	-0.3738
1.32	-0.3797	-0.3397
1.36	-0.3529	-0.3089
1.41	-0.3232	-0.2896
1.45	-0.3021	-0.2638
1.50	-0.2784	-0.2804

Table 4.2: Tabular data regarding evaluation of MINDE model on the XY dataset

Temperature	Entropy Varitions Ground Truth	MINDE
1.0	0.0000	0.0000
1.1	0.0044	-0.0030
1.2	0.0111	-0.0028
1.3	0.0207	0.0020
1.4	0.0337	0.0041
1.5	0.0505	0.0523
1.6	0.0718	0.0607
1.7	0.0981	0.1004
1.8	0.1301	0.1379
1.9	0.1689	0.1654
2.0	0.2162	0.2292
2.1	0.2749	0.2798
2.2	0.3512	0.3502
2.3	0.4495	0.4655
2.4	0.5466	0.5677
2.5	0.6167	0.6451
2.6	0.6653	0.7223
2.7	0.7020	0.7463
2.8	0.7315	0.7838
2.9	0.7560	0.7732
3.0	0.7769	0.8096
3.1	0.7949	0.7876
3.2	0.8106	0.8234
3.3	0.8243	0.8260
3.4	0.8365	0.8601
3.5	0.8473	0.8560
3.6	0.8569	0.8432
3.7	0.8656	0.8700
3.8	0.8735	0.8942
3.9	0.8806	0.8920
4.0	0.8871	0.8898

Table 4.3: Tabular data regarding the evaluation of MINDE model on the XY dataset

Chapter 5

Conclusions

This work has explored in depth both the worlds of generative AI and of entropy estimation.

Firstly, it has succesfully studied and reproduced various spin systems that have been used in literature for entropy estimation models. Samples from the data generated with Markov-Chain Monte Carlo methods have been manually inspected and have been found to follow the physical properties and characteristics predicted by the theoretical models, thus confirming the efficiency of the employed algorithms.

Diffusion models have been studied with attention to details and have once again been found very effective not only in generating new data but in truly learning the data distributions, in ways that allow to exploit them to make even elaborate computations to find mathematical quantities of interest starting from such distributions.

The data generated by the implemented method has been found to be perfectly coherent with the original ones, and the model has been able to learn and discriminate different data distributions for different temperatures of the spin system. The results for the entropy computation are satisfactory: in both the analyzed spin systems the model is able to successfully retrieve the curve of the entropy as a function of the temperature, and is thus capable of correctly identifying phase transitions and retrieving the variations of entropy that can be used to study thermodynamic properties of the systems.

While for the Ising models the results are less precise due to the considerations made in the previous sections, for the XY models the accordance between the theoretical and the diffusion model results is very good, aligned with or even better than the best machine learning entropy estimators present in literature up to now. Anyway, the results obtained in this work are very promising and show that the diffusion models can be used to estimate the entropy of a system with good precision.

As future challenges and improvements, even more physical datasets could be generated and explored, such as the Ising model on a triangular lattice. To do so and to overcome the difficulties of the presented method on discrete datasets, which was certaintly the biggest shortcoming of the method presented in this thesis, a possible further exploration would be that of trying to exploit both methods that have been found valid also with descrete datasets, such as the one presented in Appendix A, which relies on the computation of mutual information, with the diffusion model able to estimte such quantity.

Overall this work has explored a way of computing entropy that was never experimented with before, and has had a success in reproducing the entropy curves and the genertion of new data, paving the way for the exploitation of diffusion models for an ever wider range of applications and scientific problems.

Appendix A

MICE

As stated in the previous sections, it is a common practice in entropy computation to subtract a constant equal to the entropy of an "ideal gas" composed of copies of the smallest subsystem. This however wasn't something known from the beginning of the work, and for a long time there seemed to be a constant error between the output of the presented model and the ground truths provided by [22]. To find the source of this "error", and confirm wether it was a problem of the dataset, of the method or, as it turned out to be, a simple lack in the common practices in entropy calculation, another method was used to compute the entropy of the spin systems at different temperatures and compare them with the results of MINDE. The chosen method was the MICE architecture proposed in [11]. This relies once again on the relationship between entropy and mutual information. In particular, given two random variables A and B, the following relationship holds:

$$H(A, B) = H(A) + H(B) - I(A, B) \tag{A.1}$$

Given this formula, the MICE method iteratively computes the entropy of large systems by computing the entropies of its halves and their mutual information. In other words, considering the entire system X_0 of size V_0, the two halves that it can be divided in are both named X_1, since they are assumed to be statistically indistinguishable, and the mutual information between them will shortly be expressed as $I(X_1)$. By applying recursively the relationship

$$H(X_0) = 2H(X_0) - I(X_1) \tag{A.2}$$

the final formula can be obtained for the entropy of the whole system normalized by size of the system:

$$h(X_0) = \frac{H(X_0)}{V} = h_m - \frac{1}{2}\sum_{i=1}^{m} \frac{I(X_i)}{V_i} \tag{A.3}$$

where m is the number of itarations during which the system is divided in halves until the smallest subsystem of entropy h_m, formed by just one spin, is reached. Actually h_m can be ignored because it's an uninteresting addictive constant and corresponds to the quantity that is usually subtracted to obtain the "excess entropy". Implementing the MICE model, it was found that this was exactly the quantity that seemed to be an error in the MINDE results, while simply if subtracted from the total entropy results it lead to have perfect accordance with the ground truths. The authors of the MICE model employed for their method a different and older estimator for the mutual information with respect to MINDE, i.e. MINE, a model presented in [23].

Bibliography

[1] Brian DO Anderson. «Reverse-time diffusion equation models». In: *Stochastic Processes and their Applications* 12.3 (1982), pp. 313–326 (cit. on p. 6).

[2] Jonathan Ho, Ajay Jain, and Pieter Abbeel. «Denoising diffusion probabilistic models». In: *Advances in neural information processing systems* 33 (2020), pp. 6840–6851 (cit. on pp. 20, 26, 27, 29).

[3] William Peebles and Saining Xie. «Scalable diffusion models with transformers». In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, pp. 4195–4205 (cit. on pp. 21, 44).

[4] Pascal Vincent. «A connection between score matching and denoising autoencoders». In: *Neural computation* 23.7 (2011), pp. 1661–1674 (cit. on p. 22).

[5] Yang Song and Stefano Ermon. «Generative modeling by estimating gradients of the data distribution». In: *Advances in neural information processing systems* 32 (2019) (cit. on pp. 23, 28).

[6] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. «Deep unsupervised learning using nonequilibrium thermodynamics». In: *International conference on machine learning*. PMLR. 2015, pp. 2256–2265 (cit. on p. 25).

[7] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. «Score-based generative modeling through stochastic differential equations». In: *arXiv preprint arXiv:2011.13456* (2020) (cit. on p. 27).

[8] Giulio Franzese, Mustapha Bounoua, and Pietro Michiardi. «MINDE: Mutual Information Neural Diffusion Estimation». In: *arXiv preprint arXiv:2310.09031* (2023) (cit. on p. 30).

[9] Giulio Franzese, Simone Rossi, Lixuan Yang, Alessandro Finamore, Dario Rossi, Maurizio Filippone, and Pietro Michiardi. «How much is enough? a study on diffusion times in score-based generative models». In: *Entropy* 25.4 (2023), p. 633 (cit. on p. 31).

[10] Ram Avinery, Micha Kornreich, and Roy Beck. «Universal and accessible entropy estimation using a compression algorithm». In: *Physical review letters* 123.17 (2019), p. 178102 (cit. on p. 33).

[11] Amit Nir, Eran Sela, Roy Beck, and Yohai Bar-Sinai. «Machine-learning iterative calculation of entropy for physical systems». In: *Proceedings of the National Academy of Sciences* 117.48 (2020), pp. 30234–30240 (cit. on pp. 33, 62).

[12] Piotr Białas, Piotr Korcyl, and Tomasz Stebel. «Mutual information of spin systems from autoregressive neural networks». In: *Physical Review E* 108.4 (2023), p. 044140 (cit. on p. 37).

[13] Roberto C Alamino. «Explaining the Machine Learning Solution of the Ising Model». In: *arXiv preprint arXiv:2402.11701* (2024) (cit. on p. 37).

[14] Pawel Jakubczyk and Andreas Eberlein. «Thermodynamics of the two-dimensional XY model from functional renormalization». In: *Physical Review E* 93.6 (2016), p. 062145 (cit. on p. 37).

[15] Jaron Kent-Dobias and James P Sethna. «Cluster representations and the Wolff algorithm in arbitrary external fields». In: *Physical Review E* 98.6 (2018), p. 063306 (cit. on p. 39).

[16] Victor Drouin-Touchette. «The kosterlitz-thouless phase transition: an introduction for the intrepid student». In: *arXiv preprint arXiv:2207.13748* (2022) (cit. on p. 39).

[17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. «Attention is all you need». In: *Advances in neural information processing systems* 30 (2017) (cit. on p. 42).

[18] Jonathan Ho and Tim Salimans. «Classifier-free diffusion guidance». In: *arXiv preprint arXiv:2207.12598* (2022) (cit. on p. 44).

[19] Yang Song and Stefano Ermon. «Improved techniques for training score-based generative models». In: *Advances in neural information processing systems* 33 (2020), pp. 12438–12448 (cit. on p. 46).

[20] Chin-Wei Huang, Jae Hyun Lim, and Aaron C Courville. «A variational perspective on diffusion-based generative models and score matching». In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 22863–22876 (cit. on p. 46).

[21] Yang Song, Conor Durkan, Iain Murray, and Stefano Ermon. «Maximum likelihood training of score-based diffusion models». In: *Advances in neural information processing systems* 34 (2021), pp. 1415–1428 (cit. on p. 46).

[22] JF Yu, ZY Xie, Y Meurice, Yuzhi Liu, A Denbleyker, Haiyuan Zou, MP Qin, J Chen, and T Xiang. «Tensor renormalization group study of classical XY model on the square lattice». In: *Physical Review E* 89.1 (2014), p. 013308 (cit. on p. 62).

[23] Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeshwar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and Devon Hjelm. «Mutual information neural estimation». In: *International conference on machine learning*. PMLR. 2018, pp. 531–540 (cit. on p. 63).