POLITECNICO DI TORINO

Department of Mathematical Science

Master's Degree Thesis in Mathematical Engineering

A Multimodal Visual Language Model for Musical Encoding

Supervisors:

Prof. Giuseppe RIZZO

Eng. Angelica URBANELLI

Eng. Luca BARCO

Candidate:

Alfredo BAIONE

07 2024

Abstract

This thesis describes the main steps of a research experience (internship + thesis) conducted at LINKS foundation (Torino, Italy), regarding the implementation of a multimodal visual language model for musical encoding.
The goal of this activity was testing the powerful tools of AI generative modelling, in order to better understand the potentials of machine learning in the art field. From this perspective, the following pages will show how, in particular, a denoising diffusion probabilistic model (DDPM) can be adopted to generate artistic images from a 30 second musical input. The data preparation (collection, categorization, normalization, etc.), as well as the research and the implementation of a suitable model have been the fundamental passages around which all the project was developed.
Given the general and highly experimental nature of the task embraced in this challenge, the results obtained can be considered very promising. Objective evaluation metrics have shown that the model implemented can be seen a valid starting point from which to further investigate the AI creativity capabilities in a multimodal context.

"If, in fact, musicality rests on value relationships, these relationships form a structure. Either the recognition of this structure makes the perception of its constituents objects fade into the background [...], or else careful listening discovers in one isolated object variations in values it can appreciate musically." ([31])
Pierre Schaeffer (August 14, 1910 – August 19, 1995)

Table of Contents

List of Tables VI

List of Figures VII

List of Algorithms XII

Acronyms XIII

1 Introduction 1

2 State of the art 3
- 2.1 The necessity of deep generative modeling . . . 4
- 2.2 Deep generative models in literature . . . 5
  - 2.2.1 Autoregressive models . . . 6
  - 2.2.2 Flow-based models . . . 9
  - 2.2.3 Latent variable models . . . 12
- 2.3 Mathematical foundations of diffusion models . . . 17
  - 2.3.1 Diffusion process . . . 17
  - 2.3.2 Forward diffusion . . . 18
  - 2.3.3 Reverse diffusion . . . 20
  - 2.3.4 Training a diffusion model . . . 21
  - 2.3.5 Architecture . . . 23
  - 2.3.6 Guided diffusion . . . 36
  - 2.3.7 Scaled diffusion models . . . 37
  - 2.3.8 Score-based generative models . . . 39
- 2.4 Multimodalities involving music . . . 42

3 Dataset 44
- 3.1 Data sources . . . 44
  - 3.1.1 Datasets . . . 44
  - 3.1.2 Websites . . . 45

3.2 Data preprocessing 46
3.2.1 Images 46
3.2.2 Music 48

4 Methodology 53
4.1 Description of the model 53
4.1.1 Audio encoding 55
4.1.2 Optimization 56
4.1.3 Evaluation functions 57
4.2 Adaptation of the model 59

5 Experiments and results 60
5.1 Experiments 60
5.1.1 Music-image conditional model 1 60
5.1.2 Music-image conditional model 2 69

6 Conclusions 81

Appendix 83

Bibliography 94

List of Tables

3.1 Informations about the customized image-music dataset. Images and music samples are assigned to various subcategories. Numbers between brackets correspond to the numerosity of each subcategory. 52

5.1 AIS, IIS and FID values obtained from 335 images generated during the first experiment, with respect to the distribution of the test samples (for AIS and IIS scores) and the distribution of the ground truth images (for the FID score). 61
5.2 Ordered mean percentage values of similarities computed between all images generated from 10 music inputs belonging to the same category, with respect to the distribution of the 335 generated images, during the first experiment. 62
5.3 AIS, IIS and FID values obtained from 335 images generated during the second experiment, with respect to the distribution of the test samples (for AIS and IIS scores) and the distribution of the ground truth images (for the FID score). 70
5.4 Ordered mean percentage values of similarities computed between all images generated from 10 music inputs belonging to the same category, with respect to the distribution of the 335 generated images, during the second experiment. 71

List of Figures

2.1 Various applications of deep generative modeling ([88]). 3

2.2 An example of data (left) and two approaches to decision making: (middle) a discriminative approach, (right) a generative approach ([88]). 4

2.3 A diagram representing deep generative models. 6

2.4 An example of applying a shared MLP depending on two last inputs. Inputs are denoted by blue nodes, intermediate representations are denoted by orange nodes, and output probabilities are denoted by green nodes. The probability θ_d is not depending on x_d ([88]). . . . 7

2.5 An example of applying an RNN depending on two last inputs. Inputs are denoted by blue nodes, intermediate representations are denoted by orange nodes, and output probabilities are denoted by green nodes. Compared to the approach with a shared MLP, there is an additional dependency between intermediate nodes h_d ([88]). . 8

2.6 An example of applying causal convolutions. The kernel size is 2, but by applying dialtion in higher layers, a much larger input could be processed (red edges), thus, a larger memory is utilized. The first layers must be of type A to ensure proper processing ([88]). 8

2.7 Three examples of invertible transformations: (*top*) a volume preserving bijection, (*middle*) a bijection that shrinks the original area, (*bottom*) a bijection that enlarges the original area ([88]). 10

2.8 A combination of a coupling layer and a permutation layer that transforms $[x_a, x_b]$ to $[z_a, z_b]$. **A** describes a forward pass through the block. **B** describes an inverse pass through the block ([88]). . . 11

2.9 A schematic representation of the uniform dequantization for two binary random variables: (*left*) the probability mass is assigned to points, (*right*) after the uniform dequantization, the probability mass is assigned to square areas. Colors correspond to probability values ([88]). 12

2.10 An example of reparameterizing a Gaussian distribution: ϵ, distributed according to the standard Gaussian, is scaled by σ and shifted by μ ([88]). 15
2.11 A schematic representation of GANs ([88]). 17
2.12 An high level conceptual overview of the entire image space. 18
2.13 Forward diffusion process ([46]). 19
2.14 Reverse diffusion process ([46]) 20
2.15 The U-Net architecture([17]). 23
2.16 The FCN architecture ([16]). 24
2.17 A 3D U-Net encoder-decoder structure (source: `https://theaisummer.com/unet-architectures/`). 25
2.18 An example of skip connections via addition ([15]). 26
2.19 An exampe of a symmetrical encoder-decoder architecture provided with long skip connections ([43]). 27
2.20 An example of skip connections via concatenation ([33]). 27
2.21 Image by MC.AI. It shows how batch normalization brings the values in a compact range. 28
2.22 An illustration of batch normalization ([42]). 29
2.23 An illustration of group normalization. Here, the feature maps are divided into two groups, but the choice is completely arbitrary ([42]). 30
2.24 Seq2seq learning process representation ([44]). 30
2.25 Seq2seq encoder representation ([44]). 31
2.26 Seq2seq decoder representation ([44]). 31
2.27 Seq2seq with attention mechanism representation ([44]). 33
2.28 An example of a self-attention mechanism, modeled through an undirected weighted graph ([44]). 34
2.29 Cascade diffusion model pipeline ([55]). 38
2.30 Latent diffusion models representation ([72]). 39
2.31 Score-based generative modeling with score matching + Langevin dynamics representation ([39]). 40
2.32 Score-based generative modelling through stochastic differential equations ([61]). 41
2.33 Overview of score-based generative modelling through SDEs ([61]). 42

3.1 Graphic representation of music-inspired labels for the image dataset. 47

4.1 Architecture overview: a 30 seconds music sample is forwarded through a pre-trained audio encoder and then through a projection network. A pre-trained text encoder extracts tokens from vector representations of the tokenized prompt and the audio. Finally, the generative model is fed with the concatenated representations of these tokens. 54
4.2 Schematic representation of the projection network inside the Embedder. 56

5.1 Train (blue) and validation (purple) losses performance for the first experiment. 61
5.2 4 images generated from music inputs with label *20th century traditional figure*, during the first experiment. 63
5.3 4 images generated from music inputs with label *20th century experimental figure*, during the first experiment. 63
5.4 4 images generated from music inputs with label *20th century traditional open view*, during the first experiment. 64
5.5 4 images generated from music inputs with label *20th century experimental open view*, during the first experiment. 64
5.6 4 images generated from music inputs with label *Romanticism open view*, during the first experiment. 65
5.7 4 images generated from music inputs with label *Romanticism figure*, during the first experiment. 65
5.8 4 images generated from music inputs with label *Electronic music*, during the first experiment. 66
5.9 4 images generated from music inputs with label *Heavy metal*, during the first experiment. 66
5.10 4 images generated from music inputs with label *Renaissance secular subject*, during the first experiment. 67
5.11 4 images generated from music inputs with label *Baroque Classical secular subject*, during the first experiment. 67
5.12 4 images generated from music inputs with label *African traditional folklore*, during the first experiment. 68
5.13 4 images generated from music inputs with label *Asian traditional folklore*, during the first experiment. 68
5.14 4 images generated from music inputs with label *American traditional folklore*, during the first experiment. 69
5.15 Train (blue) and validation (purple) losses performance for the second experiment. 69
5.16 4 images generated from music inputs with label *20th century abstractionism*, during the second experiment. 72

5.17 4 images generated from music inputs with label *20th century experimental figure*, during the second experiment. 72
5.18 4 images generated from music inputs with label *20th century traditional open view*, during the second experiment. 73
5.19 4 images generated from music inputs with label *20th century experimental open view*, during the second experiment. 73
5.20 4 images generated from music inputs with label *Romanticism open view*, during the second experiment. 74
5.21 4 images generated from music inputs with label *Romanticism figure*, during the second experiment. 74
5.22 4 images generated from music inputs with label *Electronic music*, during the second experiment. 75
5.23 4 images generated from music inputs with label *Heavy metal*, during the second experiment. 75
5.24 4 images generated from music inputs with label *Renaissance secular subject*, during the second experiment. 76
5.25 4 images generated from music inputs with label *Baroque Classical secular subject*, during the second experiment. 76
5.26 4 images generated from music inputs with label *African traditional folklore*, during the second experiment. 77
5.27 4 images generated from music inputs with label *Asian traditional folklore*, during the second experiment. 77
5.28 4 images generated from music inputs with label *American traditional folklore*, during the second experiment. 78
5.29 4 images generated from music inputs with label *20th century traditional figure*, during the second experiment. 78
5.30 4 images generated from music inputs with label *Baroque Classical symbolism*, during the second experiment. 79
5.31 4 images generated from music inputs with label *European traditional folklore*, during the second experiment. 79
5.32 4 images generated from music inputs with label *Modern instrumental music*, during the second experiment. 80
5.33 4 images generated from music inputs with label *Renaissance symbolism*, during the second experiment. 80

1 An electronic music sample. The music is essentially experimental, obtained with a mixture of pre-sampled traditional and concrete sounds. 83
2 Sound of horses passing through a wood. 84
3 A melancholic piano improvisation recording. 84

4 A melodic fragment of a suite for piano and alto saxophone, named *Suite hellenique* and composed by Pedro Iturrlade (Falces, July 13 1929 – Madrid, November 1 2020). 85
5 An electronic music sample. Sounds are not traditional, but they are obtained with a digital synthesizer. 86
6 Another electronic music sample. The music is a combination of concrete (recorded voices) and digital sounds. 86
7 A fragment of a digital symphony. 87
8 A digital noise/industrial music sample. 87
9 4 images generated from a 90's dance music sample. 88
10 4 images generated from an experimental music fragment, realized with an electric piano, a classical guitar and a bass guitar. 88
11 4 images generated from a fragment of a nostalgic piano improvisation in jazz style. 89
12 4 images generated from a two violins canon sample. The music style is mainly classical, and the imitation is realized in unison. . . . 89
13 4 images generated from a melodic fragment of a tonal composition for flute and harp. 90
14 4 images generated from an experimental electronic piece of music titled *The Witchfinder*, composed by a duo called Amorphous Androgynous. 90
15 4 images generated from a recording of birds chirping in the early morning. 91
16 4 images generated from a fragment of a dynamic piano improvisation in G-flat major. 91
17 4 images generated from a fragment of a piano prelude, influenced primarily by the early piano pieces from *Mikrokosmos*, a work composed by Béla Bartók (Nagyszentmiklós, March 25, 1881 – New York, September 26, 1945). 92
18 4 images generated from an elaborately manipulated (i.e., distorting sounds) improvisation for electric piano and alto saxophone. 92
19 4 images generated from a fragment of *The Caterpillar Song*, featured in the famous Disney movie *Alice in Wonderland*. 93

List of Algorithms

1 Training algorithm of a DDPM ([46]). 22
2 Sampling algorithm of a DDPM ([46]). 22
3 Classifier guided diffusion sampling ([51]). 37

Acronyms

AI
artificial intelligence

AIC
audio-image content

AIS
audio-image similarity

ARM
autoregressive model

BN
batch normalization

CLIP
contrastive language-image pre-training

CMRA
cross-modal ranking analysis

CNN
convolutional neural network

csv
comma-separated values

DL
deep learning

DM

diffusion model

ELBO

evidence lower bound

FCN

fully convolutional network

FID

Fréchet inception distance

GAN

generative adversarial network

GB

gigabyte

GELU

gaussian error linear unit

GIILS

generated image-image label similarity

GN

group normalization

GPT

generative pre-training transformer

GPU

graphic processing unit

IIS

image-image similarity

JPEG

joint photographic experts group

KL

Kullback-Leibler

LDM

latent diffusion model

LLM

large language model

MHSA

multi-head self-attention

MLP

multi-layer perceptron

MM

multi-modal

MSE

mean squared error

NCSN

noise conditional score network

NVP

non-volume preserving

PE

position embeddings

PNG

portable network graphics

ReLU

rectified linear unit

RL

reinforcement learning

RNN

recurrent neural network

SDE

stochastic differential equation

SSL

self supervised learning

VAE

variational autoencoder

WRN

wide residual network

ZSL

zero-shot learning

Chapter 1

Introduction

One of the reasons why automatic analysis of image content in data is important relates to the development of human-centric intelligent systems. For instance, in recent years, significant progress has been made toward the analysis of emotion in individual modalities ([20], [26], [19]). However, emotion recognition is a challenging task due to huge variability and subjectivity involved in the expression and perception of human emotion. For this reason, some studies related to this topic focus on human perception, specifically consisting of cross-modal works involving audio and visual data. Based on neuroscientific studies around *synaesthesia*, Li et al. ([7]) built a prototype cross-media retrieval system to establish a semantic correspondence between music and images. Synaesthesia is also the main theme around which Xing et al. ([41]) develop a cross-synaesthesia-aware model based on music and image similarity in the emotion space. In [25], instead, authors propose a method able to learn the non-linear relationship between music and image, and to integrate heterogeneous ranking data from different modalities into a unified space.
However, little attention has been paid to another human resource, that can be identified with the name of "creativity". In a context like automatic learning, the powerful tools of multimodal applications in the field of audio and visual data, whose literature is consistent (e.g., [40], [11], [10], [82], [29], [28], [57], [77], [79]), represents a solid starting point for investigating the creativity potential of automatic systems. Generative modelling is, then, a possible direction to take. In fact, this approach, which has recently become very common, seems to provide the most natural framework for conductiong this type of research due to its extreme flexibility in adapting to different contexts. For instance, Hao et al. ([29]) developed a specific network for visual-audio mutual generation, based on the common informations shared by the two kinds of data. Chen et al. ([28]) extended this idea working on two specific scenarios: instrument-oriented generation and pose-oriented generation. Zhu et al. ([82]) take a different route by conditioning the generation process of

music and images to dances, text and labels.

This thesis focuses on the application of a particular generative model, namely a *diffusion model* (whose main literature includes [46], [69], [63], [55], [67], [73], [72], [18], [58], [70]), in order to generate artistic images from music inputs. Music and artistic images (like paintings, murals, logos, etc.) share not only semantic relations, but also creative structures which have developed in history according to social and cultural traditions. Exploring them through the lens of multimodal generative learning models can lead to a deeper understanding, through appropriate measurements, of those creative mechanisms governing the human mind under specific circumstances. In order to achieve that, the use of a diffusion model provides several advantages. In fact, these models generate high-resolution images at a level beyond the reach of other generative models (as shown by Saharia et al in [73]). Moreover, as shown by Nichol et al. ([58]), they are extremely prone to conditioning, which can be very helpful in determine dependences between inputs and outputs. Lastly, as highlighted by Rombach et al. ([72]), they are efficiently scalable.
The model proposed here is based on an architecture named *AudioToken*: a novel method utilizing latent diffusion models, trained for text-to-image-generation, to generate images, conditioned on audio recordings ([80]). Using a pre-trained audio encoding model ([64]), the chosen method encodes audio into a new token which can be considered as an adaptation layer between the audio and text representations ([80]). Such a modelling paradigm, moreover, requires a small number of trainable parameters, making the proposed approach appealing for lightweight optimization. The dataset adopted addresses three key issues: a customized preparation of the data (following the approach of Chen et al.[45]), a high cardinality and a refined categorization. It contains 24216 artistic images (mainly taken from the WikiArt dataset [89] and Pinterest [85]) and 24216 30 seconds music samples (taken from YouTube [90]). Both images and audio files are organized into 18 labelled categories. AudioToken adaptation to the music-image dataset involves the development of a new training-validation framework, together with the introduction of a customized objective metric, named GIILS (Generated Image-Image Label Similarity): a metric that evaluates the semantic similarity among generated images obtained from music inputs belonging to the same category. Results for other common evaluation metrics (AIS, IIS and FID) are also provided in this context, as well as images generation from music inputs taken outside the dataset.

Chapter 2

State of the art

Deep generative modelling is nowadays very popular in the AI field. The term "deep" in the AI context refers to the extensive use of neural networks architectures in order to build machine learning models (also called "deep learning" models, or DL models), as well explained by Zhang et al. ([81]). Not to mention the most famous large language models (LLM) like ChatGPT, BERT, Claude, etc., there is a plethora of multimodal generative tools (as DALL-E, Midjourney, Sora, etc.) capable of carrying out several task and ready to be further improved or renewed. In fact, AI applications can vary from typical modalities considered in machine learning, i.e., text analysis ([21], [52], [59]), image analysis ([50], [71]), audio analysis ([64], [9], [74]), to problem in active learning ([38]), reinforcement learning ([32]), graph analysis ([34]) and medical imaging ([17], [27]).

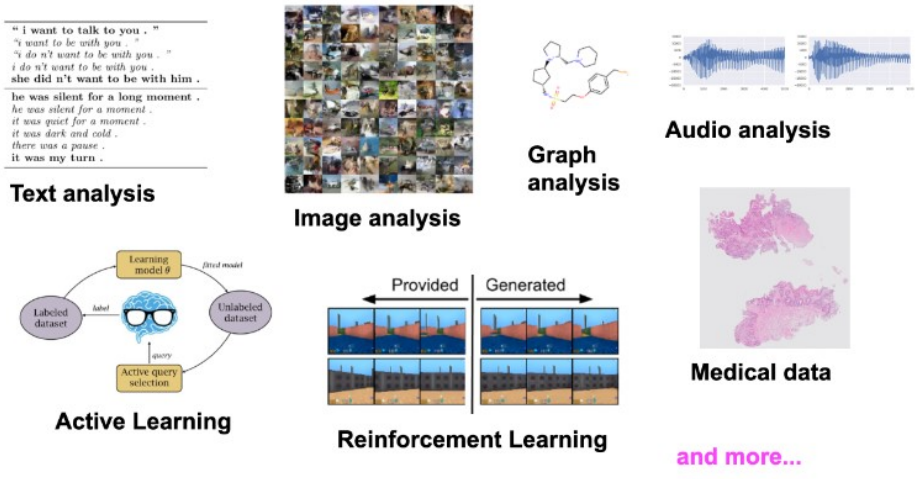

Figure 2.1: Various applications of deep generative modeling ([88]).

2.1 The necessity of deep generative modeling

As stated in [86], today's AI aim is to built intelligent entities, capable of acting efficiently and safely in a variety of new situations. The possibility to have access to proper hardwares (like the latest NVIDIA GPUs) by the biggest AI tech companies (Open AI, Google, Amazon, etc.) in the last decade is making this dream come true ([87]).
The reason why AI generative modelling is considered the most suitable sub-field of AI for realizing the aforementioned dream can be explained with the following example, taken from [88].
Let us consider a system that classifies objects into two classes: orange and blue. Assuming two-dimensional data (Figure 2.2, left) a new datapoint has to to be classified (a black cross in Figure 2.2). Decisions can be taken using two approaches:

1. a classifier could be formulated explicitly by modeling the conditional distribution $p(y|x)$ (Figure 2.2, middle);

2. a joint distribution $p(x, y)$, that could be further decompose as $p(x, y) = p(y|x)p(x)$ (Figure 2.2, right), could be taken into cosideration.

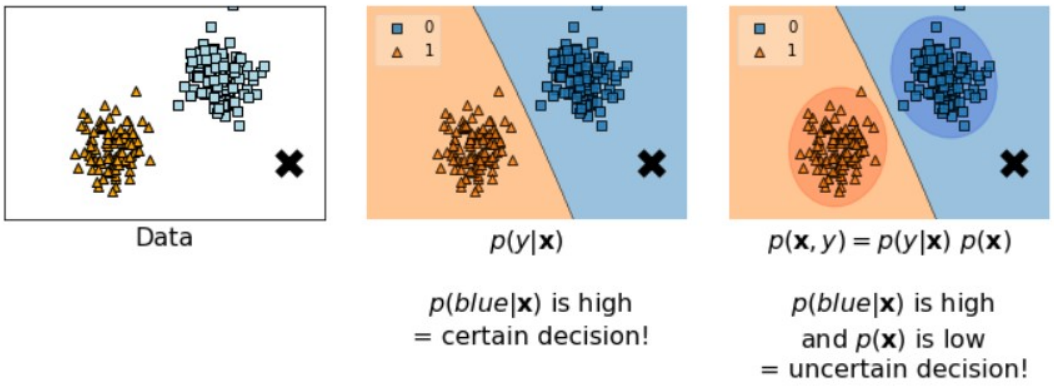

Figure 2.2: An example of data (left) and two approaches to decision making: (middle) a discriminative approach, (right) a generative approach ([88]).

After training a model using a discriminative approach, namely, the conditional distribution $p(y|x)$, a clear decision boundary is obtained. Given that black cross is farther away from the orange region, the classifier assigns a higher probability to the blue label. As a result, it is certain about the decision. On the other hand, additionally fitting a distribution, $p(x)$, implies that the black cross is not only farther away from the decision boundary, but it is also distant from the region

where blue datapoints lie. In other words, the black point is far away from the region of high probability mass. As a result, the probability of the black cross, $p(x = black\,cross)$, is low, the joint distribution $p(x = black\,cross, y = blue)$ will be low as well and, thus, the decision is uncertain.
This simple example clearly indicates that building AI systems that make reliable decisions and communicate with human beings, requires understanding the environment first. For this reason, this systems cannot simply learn how to make decisions, but they should be able to quantify their beliefs about their surrounding using the language of probability. In order to do that, estimating the distributions over objects, $p(x)$, is crucial and this is the fundamental task generative models try to accomplish. From a generative perspective:

- $p(x)$ can be used to asses whether a given object has been observed in the past or not;

- $p(x)$ can help to properly weight the decision;

- $p(x)$ can be used to asess uncertainty about the environment;

- $p(x)$ can be used to actively learn by interacting with the environment (e.g. by asking for labeling objects with low $p(x)$);

- $p(x)$, eventually, can be used to generate new objects.

2.2 Deep generative models in literature

Deep generative models can be divided into 3 main groups (see Figure 2.3):

- autoregressive generative models (ARM);

- flow based models;

- latent variables models.

ARMs, flows, and prescribed models like VAEs are all likelihood-based models, meaning that their optimization functions can be obtained by the so-called *likelihood* function, defined as $L(\theta|x) = f(x;\theta)$, where $f(x;\theta)$ is a statistical model, x represents the observed data and θ represents the parameters of the model ([56]). On the other hand, implicit models like GANs suffer from instability, so they can not be trained directly using the likelihood function ([88]).
Next subsection describes briefly the mathematical structure of these models, following the approach of [88].

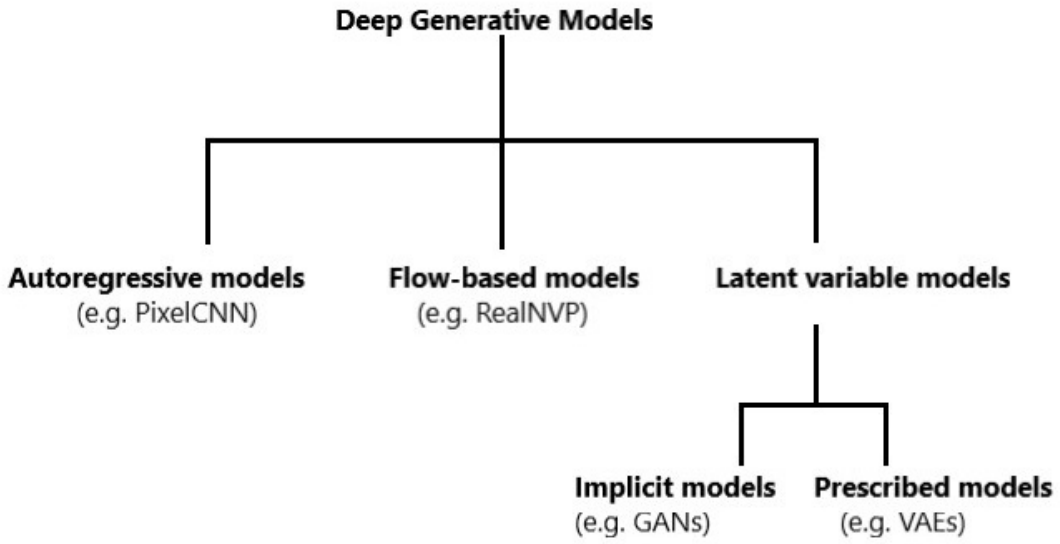

Figure 2.3: A diagram representing deep generative models.

2.2.1 Autoregressive models

Let us consider a high-dimensional random variable $x \in \mathcal{X}^D$, $\mathcal{X} = \{0,1,...,255\}$ or $\mathcal{X} = \mathbb{R}$, and D is the variable dimension. Thanks to the product rule, the joint distribution $p(x)$ can be expressed as:

$$p(x) = p(x_1) \prod_{d=2}^{D} p(x_d|x_{<d})$$

where $x_{<d} = [x_1, x_2, ..., x_{d-1}]^\mathsf{T}$.

As shown, the product rule applied multiple times to the joint distribution provides a principled manner of factorizing the joint distribution into many conditional distributions. However, modelling all conditional distributions $p(x_d|x_{<d})$ separately is simply infeasible. Doing that would lead to D separate models, and the complexity of each model would grow due to varying conditions.

The first attempt to limit the complexity of a conditional model is to assume a finite memory. For instance, it is possible to assume that each variable is dependent on a maximum of two other variable, namely:

$$p(x) = p(x_1)p(x_2|x_1) \prod_{d=3}^{D} p(x_d|x_{d-1}, x_{d-2}).$$

Then, a small neural network, e.g., a Multi-layered Perceptrons (MLP), can predict the distribution of x_d. If $\mathcal{X} = \{0,1,...255\}$, the MLP takes x_{d-1}, x_{d-2} and outputs

probability from the categorical distribution of x_d, θ_d. The MLP could be of the following form:

$$[x_{d-1}, x_{d-2}] \rightarrow \text{Linear}(2, M) \rightarrow \text{ReLU} \rightarrow \text{Linear}(M, 256) \rightarrow \text{softmax} \rightarrow \theta_d$$

where M denotes the number of hidden units, e.g., $M = 300$. ReLU, standing for Rectified Linear Unit, is an activation function, commonly used in artificial neural networks. It is defined as $\text{ReLU}(x) = \max(0, x)$ ([81]). softmax is a common output function adopted for multiclassification problems. It is defined as $\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{c=0}^{N_c-1} e^{z_c}}$, where N_c is the number of classes and z_i is the logit function ($\ln(\frac{p}{1-p})$) associated to the ith class ([62]).
An example of this approach is depicted in Figure 2.4.

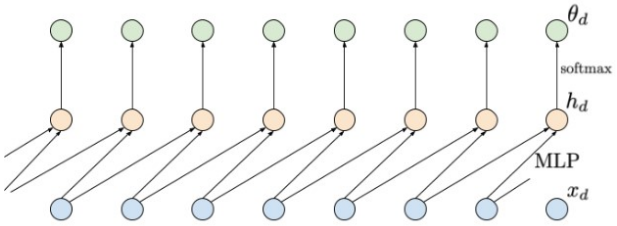

Figure 2.4: An example of applying a shared MLP depending on two last inputs. Inputs are denoted by blue nodes, intermediate representations are denoted by orange nodes, and output probabilities are denoted by green nodes. The probability θ_d is not depending on x_d ([88]).

The obvious drawback, in this case, is a limited memory. In many problems, e.g., image processing, learning long-range statistics is crucial to understand complex patterns in data. A possible solution to this problem relies on applying a recurrent neural network (RNN [12]). In other words, the conditional distributions can be modeled as follows:

$$p(x_d|x_{<d}) = p\left(x_d|\text{RNN}(x_{d-1}, h_{d-1})\right)$$

where $h_d = \text{RNN}(x_{d-1}, h_{d-1})$, and h_d is a hidden context acting as a memory that allows learning long-range dependencies. An example of using an RNN is presented in Figure 2.5.

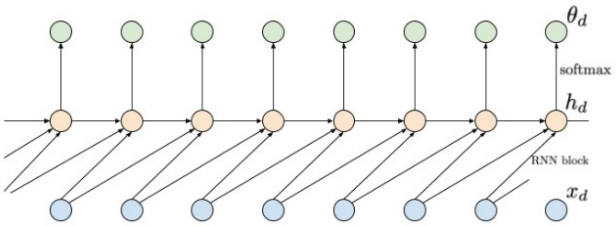

Figure 2.5: An example of applying an RNN depending on two last inputs. Inputs are denoted by blue nodes, intermediate representations are denoted by orange nodes, and output probabilities are denoted by green nodes. Compared to the approach with a shared MLP, there is an additional dependency between intermediate nodes h_d ([88]).

This approach gives a single parametrization, thus, it is efficient and also solves the problem of a finite memory. Unfortunately, RNNs suffer from other issues, namely: they are sequential, hence, slow, if they are badly conditioned and they suffer from exploding or vanishing gradients. In [14] authors notice that convolutional neural networks (CNNs) could be used instead of RNNs to model long-range dependencies. To be more precise, one-dimensional convolutional layers (Conv1D) could be stacked together to process sequential data. To ensure proper processing, moreover, the first Conv1D layers must be dependent on all the inputs but the current one (type A), while the other Conv1D layers can depend only on the current one (type B). Additionally, in order to increase the effective kernel size, long-range dependencies can be learned using dilation.
Figure 2.6 presenst an example of a neural network consisting of 3 causal Conv1D layers. The first causal Conv1D is of type A, while the next two layers are of type B, with dilation 2 and 3.

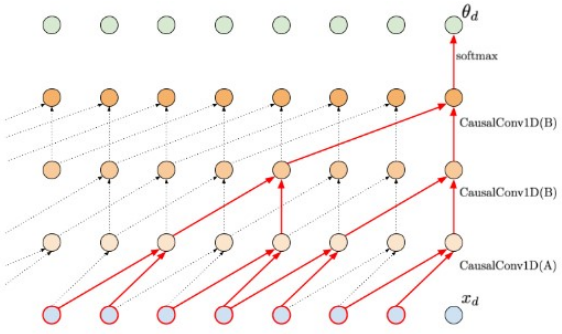

Figure 2.6: An example of applying causal convolutions. The kernel size is 2, but by applying dialtion in higher layers, a much larger input could be processed (red edges), thus, a larger memory is utilized. The first layers must be of type A to ensure proper processing ([88]).

There is a drawback of applying autoregressive model parametrized by causal convolutions. In fact, sampling new objects requires D full forward passes. That is a big waste, but it is also the price for all the positive effects following from the convolutional-based parametrization of the ARM.

2.2.2 Flow-based models

The main advantages of ARMs is that they can learn long range statistics and, as a consequence, powerful density estimators. However, their drawback is that they are parametrized in an autoregressive manner, hence, sampling is rather a slow process. To alleviate this uncomfortable situation it is possible to come up with a different approach to directly model the data distribution $p(x)$.
Let us consider a random variable $z \in \mathbb{R}$, with $\pi(z) = N(z|0,1)$ (i.e., z is normally distributed with 0 mean and variance equal to 1) and a linear transformation, e.g., $x = a \cdot z + b$, with $a, b \in \mathbb{R}$. Remembering the *change of variables* formula, the distribution of x can be calculated as:

$$p(x) = \pi\left(z = f^{-1}(x)\right)\left|\frac{\partial f^{-1}(x)}{\partial x}\right| \tag{2.1}$$

where f is an invertible function (a bijection, i.e., a function that maps one point to another, distinctive point, and it is always possible to invert it, in order to obtain the original point).
It is easy to verify that x still follows a normal distribution and, in particular, $\left|\frac{\partial f^{-1}(x)}{\partial x}\right|$ is responsible to normalize the distribution $\pi(z)$ after applying the transformation f. In other words, $\left|\frac{\partial f^{-1}(x)}{\partial x}\right|$ counteracts as a possible change of volume caused by f.
This example indicates that it is possible to calculate a new distribution of a continuous random variable by applying a known bijective transformation f to a random variable x, with a known distribution $z \sim p(z)$. x and z can also be multiple variables, e.g., $x, z \in \mathbb{R}^D$, for some D. In this case, it holds

$$\left|\frac{\partial f^{-1}(x)}{\partial x}\right| = |\det \mathbf{J}_{f^{-1}}(x)|$$

where $\mathbf{J}_{f^{-1}}$ is the jacobian matrix of f, defined as follows:

$$\mathbf{J}_{f^{-1}} = \begin{bmatrix} \frac{\partial f_1^{-1}}{\partial x_1} & \cdots & \frac{\partial f_1^{-1}}{\partial x_D} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_D^{-1}}{\partial x_1} & \cdots & \frac{\partial f_D^{-1}}{\partial x_D} \end{bmatrix}.$$

The *inverse function* theorem yields

$$|\mathbf{J}_{f^{-1}}(x)| = |\mathbf{J}_f(x)|^{-1}$$

that allows rewriting 2.1 as:

$$p(x) = \pi\left(z = f^{-1}(x)\right)|\mathbf{J}_f(x)|^{-1}.$$

To get some insight into the role of the Jacobian-determinant, it is possible to look at Figure 2.7. Here, there are three cases of invertible transformations that play around with a uniform distribution defined over a square.

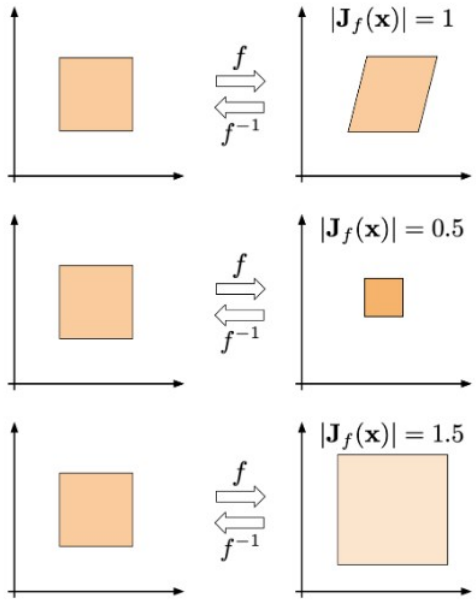

Figure 2.7: Three examples of invertible transformations: (*top*) a volume preserving bijection, (*middle*) a bijection that shrinks the original area, (*bottom*) a bijection that enlarges the original area ([88]).

A natural question is whether it is possible to utilize the idea of the change of variable to model a complex and high-dimensional distribution over images, audio or other data sources. Let us consider a sequence of invertible transformations, $f_k : \mathbb{R}^D \to \mathbb{R}^D$. Starting with a known distribution, $\pi(z_0) = N(z_0|0, \mathbb{I})$, the invertible transformations are used to obtain a flexible distribution (using the notation of a Jacobian for the ith transformation):

$$p(x) = \pi\left(z_0 = f^{-1}(x)\right)\prod_{i=1}^{K}|\mathbf{J}_{f_i}(z_{i-1})|^{-1}.$$

Let $\pi(z_0)$ be $N(z_0|0,\mathbb{I})$. Then, the logarithm of $p(x)$ is the following:

$$\ln p(x) = \ln N\left(z_0 = f^{-1}(x)|0,\mathbb{I}\right) - \sum_{i=1}^{K} \ln |\mathbf{J}_{f_i}(z_{i-1})|.$$

The first part, $\ln N(z_0 = f^{-1}(x)|0,\mathbb{I})$, corresponds to the *Mean Squared Error* (MSE) loss between 0 and $f^{-1}(x)$. The second part, $\sum_{i=1}^{K} \ln |\mathbf{J}_{f_i}(z_{i-1})|$, ensures that the distribution is properly normalized and can be considered as a kind of regularizer for the invertible transformations $\{f_i\}$. Models consisting of invertible transformations (usually neural networks) with tractable Jacobian-determinants are referred to as *normalizing flows* or *flow-based-models*.
A very important class among these kind of models is the REALNVP (*Real-valued Non-Volume Preserving flows* [22]). The main features of a REALNVP model are the followings:

- **coupling layers**: the idea behind this transformation is considering an input to the layer divided into two parts: $x = [x_a, x_b]$. Then, the invertible transformation is defined as

$$\begin{aligned} y_a &= x_a \\ y_b &= \exp(s(x_a)) \odot x_b + t(x_a) \end{aligned}$$

  where $s(\cdot)$ and $t(\cdot)$ are arbitrary neural networks called, respectively, *scaling* and *transition*, whose Jacobian-determinant logarithm is easy to calculate;

- **permutation layer**: it is an effective transformation that could be combined with a coupling layer. Since permutation preserves volumes, i.e., its Jacobian-determinant is equal to 1, it can be adopted each time after the coupling layer. For instance, it is possible to reverse the order of the variables, as shown in Figure 2.8.

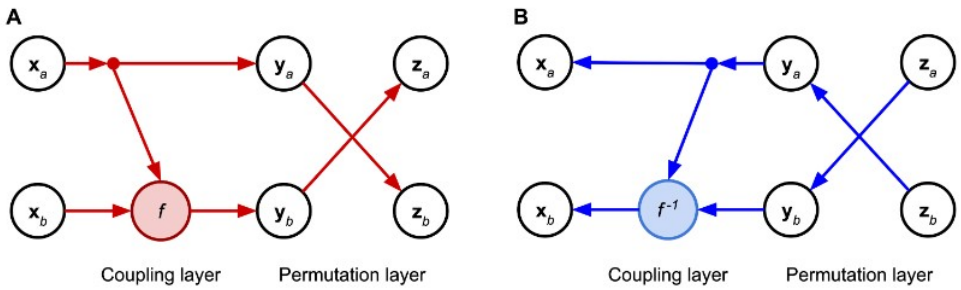

Figure 2.8: A combination of a coupling layer and a permutation layer that transforms $[x_a, x_b]$ to $[z_a, z_b]$. **A** describes a forward pass through the block. **B** describes an inverse pass through the block ([88]).

- **dequantization**: flow based models assume that x is a vector of real-valued random variables. However, in practice, many objects are discrete. For instance, images are typically represented as integers taking values in $\{0,1,...,255\}^D$. Theis et al. ([24]) outline that adding a uniform noise, $u \in [-0.5, 0.5]^D$, to the original data, $y \in \{0,1,...,255\}^D$, allows applying density estimation to $x = y + u$. This procedure is known as *uniform dequantization*. An example of uniform dequantization applied to two binary random variables is depicted in Figure 2.9:

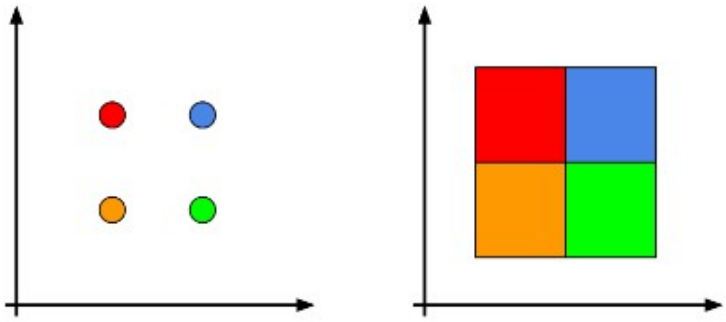

Figure 2.9: A schematic representation of the uniform dequantization for two binary random variables: (*left*) the probability mass is assigned to points, (*right*) after the uniform dequantization, the probability mass is assigned to square areas. Colors correspond to probability values ([88]).

2.2.3 Latent variable models

The idea behind latent variable models is to assume a lower-dimensional latent space and the following generative process:

$$z \sim p(z)$$
$$x \sim p(x|z).$$

In other words, the latent variables correspond to hidden factors in data, and the conditional distribution $p(x|z)$ could be treated as a generator.

Variational Auto-Encoders

The approach underlying the utilization of latent variable models relies on the presence of certain crucial factors within the data that can be exploited to enhance the learning of $p(x)$ (and, consequently, generate more precise objects).
Namely, given an high-dimensional object of interest, $x \in \mathcal{X}^D$ (e.g., for images, $\mathcal{X} \in \{0,1,...,255\}$) it is possible to introduce some low-dimensional latent variables,

$z \in \mathcal{Z}^M$ (e.g. $\mathcal{Z} = \mathbb{R}$), called "hidden factors". $\mathcal{Z}^M$ can be seen as a low-dimensional manifold (i.e., a topological space that locally resembles Euclidean space near each point, as stated in [2]). Then, the generative process could be expressed as follows:

- z is sampled from $z \sim p(z)$;
- then, x is sampled from the conditional distribution $x \sim p(x|z)$.

Therefore, the mathematical idea behind latent variables model is that, once introduced the latent variable z, the joint distribution is factorized as follows: $p(x, z) = p(x|z)p(z)$. This naturally expresses the genereative process described above. However, only x is trainable. Therefore, according to probabilistic inference, summing out (or marginalizing out) the unknown z is necessary. As a result, the (marginal) likelihood function is the following:

$$p(x) = \int p(x|z)p(z)\, dz.$$

A natural question is how to calculate this integral. In general, it is a difficult task. The simplest approach would be to use the Monte Carlo approximation ([8]):

$$\begin{aligned} p(x) &= \int p(x|z)p(z)\, dz \\ &= \mathbb{E}_{z\sim p(z)}[p(x|z)] \\ &\approx \frac{1}{K}\sum_{k=1}^{K} p(x|z_k) \end{aligned}$$

where, in the last line, samples from the prior over latents are used ($z_k \sim p(z)$). Such approach is relatively easy; however, if z is multidimensional, the *curse of dimensionality* trap arises. This curse, generally, refers to issues that arise when the number of datapoints is small (in a suitably defined sense) relative to the intrinsic dimension of the data ([56]).
A possible approach, here, is using a specific approximated inference called *variational inference* ([5]). Let us consider a family of variational distributions parametrized by Φ, $\{q_\Phi(z)\}_\Phi$. For instance, Gaussian distributions with means and variances given by μ and σ^2, respectively, can be taken into consideration. The form of these distributions is well known and, assuming that they assign non-zero probability mass to all $z \in \mathcal{Z}^M$, the logarithm of the marginal distribution could be approximated

as follows:

$$\begin{aligned}
\ln p(x) &= \ln \int p(x|z)p(z)\,dz \\
&= \ln \int \frac{q_\Phi(z)}{q_\Phi(z)} p(x|z)p(z)\,dz \\
&= \ln \mathbb{E}_{z\sim q_\Phi(z)} \left[\frac{p(x|z)p(z)}{q_\Phi(z)}\right] \\
&\geq \mathbb{E}_{z\sim q_\Phi(z)} \ln \left[\frac{p(x|z)p(z)}{q_\Phi(z)}\right] \\
&= \mathbb{E}_{z\sim q_\Phi(z)} \left[\ln p(x|z) + \ln p(z) - \ln q_\Phi(z)\right] \\
&= \mathbb{E}_{z\sim q_\Phi(z)} \left[\ln p(x|z)\right] - \mathbb{E}_{z\sim q_\Phi(z)} \left[\ln q_\Phi(z) - \ln p(z)\right].
\end{aligned}$$

In the fourth line of the above equation Jensen's inequality holds. This famous inequality is generally stated in the following form: if X is a random variable and φ is a convex function, then $\varphi(\mathbb{E}[X]) \leq \mathbb{E}[\varphi(x)]$.
Considering an amortized variational posterior, namely, $q_\Phi(z|x)$ instead of $q_\Phi(z)$ for each x, leads to:

$$\ln p(x) \geq \mathbb{E}_{z\sim q_\Phi(z)} \left[\ln p(x|z)\right] - \mathbb{E}_{z\sim q_\Phi(z)} \left[\ln q_\Phi(z|x) - \ln p(z)\right].$$

Amortization could be extremely useful, because the training of a model, in this case, returns parameters of a distribution for given inputs.
This results is an auto-encoder like model, with a stochastic encoder $q_\Phi(z|x)$, and a stochastic decoder, $p(x|z)$. The lower-bound of the log-likelihood function is called *Evidence Lower Bound* (ELBO). The first part of the ELBO, $\mathbb{E}_{z\sim q_\Phi(z)} \left[\ln p(x|z)\right]$, is referred to as the (negative) *reconstruction error*, because x is encoded to z and then decoded back. The second part of the ELBO, $\mathbb{E}_{z\sim q_\Phi(z)} \left[\ln q_\Phi(z|x) - \ln p(z)\right]$, could be seen as a regularizer and it coincides with the Kullback-Leibler divergence (D_{KL}) between the variational posterior and the prior ($D_{KL}\left(q_\Phi(z|x)||p(z)\right)$). In this case, this divergence can be seen as a gap between the ELBO and the true log-likelihood.
Encoders and decoders are typically modeled using neural networks, with careful consideration given to ensuring that their distributions are appropriate for the specific problem at hand. For instance, if data are images, a possible distribution for $x|z$ is the categorical distribution (considered to be the generalization of the Bernoulli distribution for a categorical random variable, as stated in [8]). The choice of a distribution for the latent variables depends on the way latent factors are expressed in data. For convenience, typically z is taken as a vector of continuous random variables, $z \in \mathbb{R}^M$. Then, Gaussians, for both the variational posterior and

the prior, can be adopted:

$$q_\Phi(z|x) = N\left(z|\mu_\Phi(x), \text{diag}\left[\sigma^2_\Phi(x)\right]\right)$$
$$p(z) = N(z|0, \mathbb{I})$$

where $\mu_\Phi(x)$ and $\sigma^2_\Phi(x)$ are outputs of a neural network, while diag $[\sigma^2_\Phi(x)]$ is a square matrix having $\sigma^2_\Phi(x)$ on every entry of the main diagonal and zeros elsewhere. In order to avoid the integral represented by the expected value, a Monte Carlo strategy can be adopted. In this way, z can be sampled from $q_\Phi(z|x)$, plugged into the ELBO, and gradients can be computed with respect to the parameters of a neural network Φ. The reason why samples are taken from $q_\Phi(z|x)$ instead of $p(z)$ is that the variational posterior assigns typically more probability mass in a smaller region than the prior. Practically, even when following this strategy, the variance of the gradient may still be quite large. A possible solution to that is the idea of reparameterizing the variational posterior with the so called *reparameterization trick* ([3]). The idea is that a random variable can be expressed as a composition of primitive transformations (e.g., arithmetic operations, logarithms, etc.) of an indipendent random variable with a simple distribution. For instance, for a Gaussian random variable z with mean μ and variance σ^2 and an independent random variable $\epsilon \sim N(\epsilon|0,1)$, it holds:

$$z = \mu + \sigma \cdot \epsilon.$$

Now, sampling ϵ from the standard Gaussian, and applying the above transformation, lead to a sample from $N(z|\mu, \sigma^2)$.

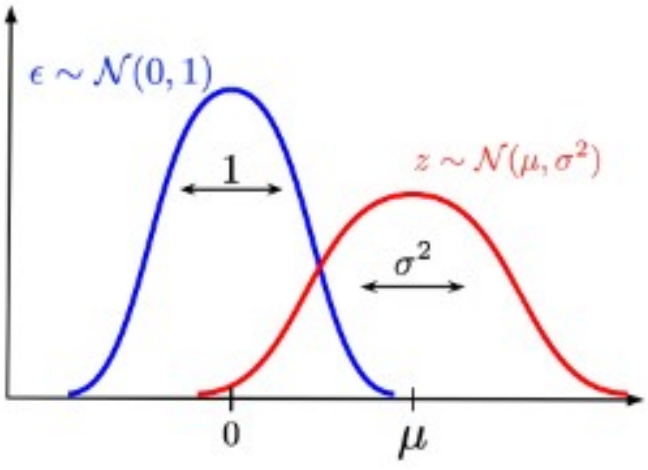

Figure 2.10: An example of reparameterizing a Gaussian distribution: ϵ, distributed according to the standard Gaussian, is scaled by σ and shifted by μ ([88]).

Generative Adversarial Networks

When issues with dimensionality become particularly restrictive for the generative tasks, another option, in addition to the previously mentioned variational inference, is to abandon the likelihood-based approach. This is done in favour of a so-called *adversarial loss*, on which generative adversarial networks rely.
In order to understand this kind of optimization framework, it is possible to introduce a function called *discriminator*, which takes an object x and returns the probability whether it is real (i.e., coming from the empirical distribution), $D_\alpha : \mathcal{X} \to [0,1]$. Then, another function, called *generator*, that takes noise and turns it into an object x, i.e., $G_\beta : \mathcal{Z} \to \mathcal{X}$, can be introduced. All x's coming from the empirical distribution $p_{data}(x)$ are called *real* and all x's generated by $G_\beta(z)$ are called *fake*. The objective function can be constructed as follows:

- there are two sources of data: $x \sim p_\theta(x) = \int G_\beta(z)p(z)\,dz$ and $x \sim p_{data}(x)$;

- the discriminator solves a classification task by assigning 0 to all fake datapoints and 1 to all real datapoints;

- since the discriminator can be seen as a classifier, the binary cross-entropy loss function (as described in [56]) can be adopted, in the following form:

$$l(\alpha, \beta) = \mathbb{E}_{x \sim p_{real}} \left[\ln D_\alpha(x)\right] + \mathbb{E}_{z \sim p(z)} \left[\ln \left(1 - D_\alpha \left(G_\beta(z)\right)\right)\right]$$

  where the left part corresponds to the real data source, and the right part contains the fake data source;

- $l(\alpha, \beta)$ is maximized with respect to α (i.e., the discriminator);

- the generator tries to fool the discriminator, thus, it tries to minimize $l(\alpha, \beta)$ with respect to β (i.e., the generator).

Eventually, the optimization problem is:

$$\min_\beta \max_\alpha \mathbb{E}_{x \sim p_{real}} \left[\ln D_\alpha(x)\right] + \mathbb{E}_{z \sim p(z)} \left[\ln \left(1 - D_\alpha \left(G_\beta(z)\right)\right)\right].$$

As always, the generator and the discriminator are parametrized using deep neural networks learnable using the adversarial loss (i.e., optimizing the min-max problem). The resulting class of models is called Generative Adversarial Networks (GANs [13]). Figure 2.11 shows the idea of GANs. The generator part constitutes an implicit distribution, i.e., a distribution from an unknown family of distributions. Its analytical form is also unknow; nevertheless, it is possible to sample from it.

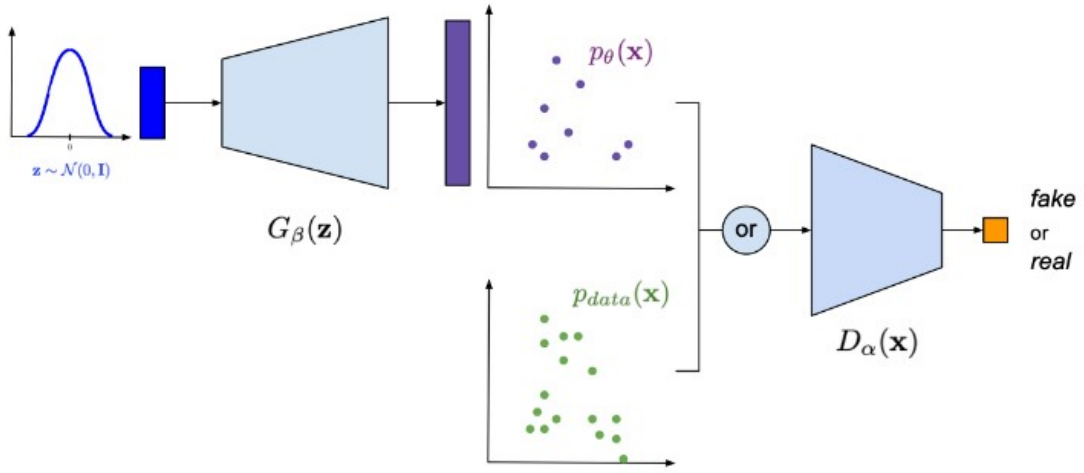

Figure 2.11: A schematic representation of GANs ([88]).

2.3 Mathematical foundations of diffusion models

Diffusion models are a class of state-of-the-art generative models that generate diverse high resolution images. They are extensively utilized by OpenAI, Nvidia and Google managed to train large-scale models. Examples architectures that are based on diffusion models are GLIDE, DALLE-2, Imagen and the full open-source stable diffusion. Next subsections, following the approach of [69], describe the mathematical concepts as well as the main typologies of diffusion models, starting from the DDPM as initialized by Sohl et al. ([18]) and then proposed by Ho et al. ([46]), up to the score-based generative ones (described in [39]).

2.3.1 Diffusion process

The basic idea behind diffusion models is rather simple. They take the input image x_0 and then add Gaussian noise to it through a series of T steps. This process is called *forward diffusion process*, but this is unrelated to the forward pass (i.e., the *forward propagation*, that refers to the calculation and the storage of intermediate variables, including outputs, for a neural network in order from the input layer to the output layer, as stated in [81]). Afterwards, a neural network is trained to recover the original data by reversing the noising process. By being able to model the reverse process, it is possible to generate new data. This is the so-called *reverse diffusion process*.

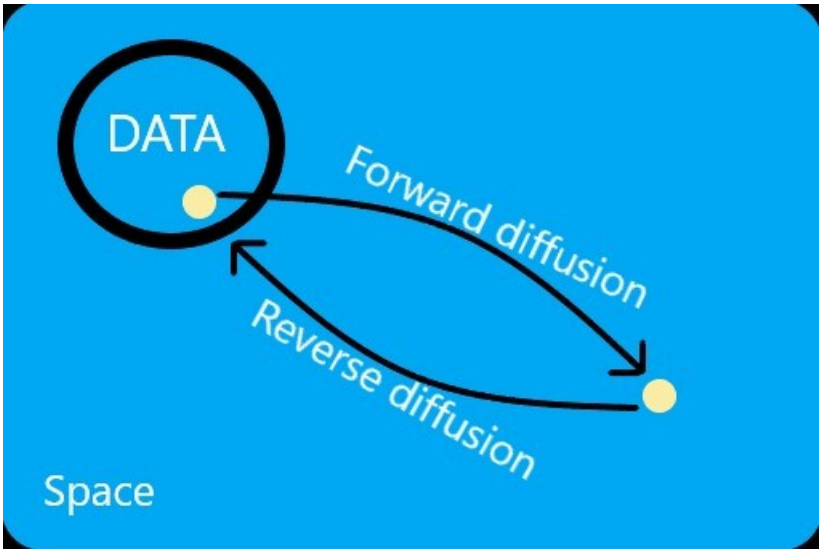

Figure 2.12: An high level conceptual overview of the entire image space.

2.3.2 Forward diffusion

Diffusion models can be seen as latent variable models, given that they always refer to a hidden continuous feature space. In practice, they are formulated using a Markov chain of T steps. A Markov chain is a stochastic process $\{X_n, n = 0,1, ...T\}$ such that the conditional distribution of any future state X_{n+1}, given the past states $X_0, X_1, ..., X_{n-1}$ and the present state X_n, is independent of the past states and depends only on the present state ([4]). Importantly, there is no need to use a specific type of neural network, unlike with flow-based models.
Given a data-point x_0, sampled from the real data distribution $q(x)$ ($x_0 \sim q(x)$), one can define a forward diffusion process by adding noise. Specifically, at each step of the Markov chain a Gaussian noise with variance β_t is added to x_{t-1}, producing a new latent variable x_t with distribution $q(x_t|x_{t-1})$. This diffusion process can be formulated as follows:

$$q(x_t|x_{t-1}) = N(x_t; \mu_t = \sqrt{1-\beta_t}x_{t-1}, \Sigma_t = \beta_t\mathbb{I})$$

with $\beta_t \in (0,1)$, $t = 1, ..., T$.

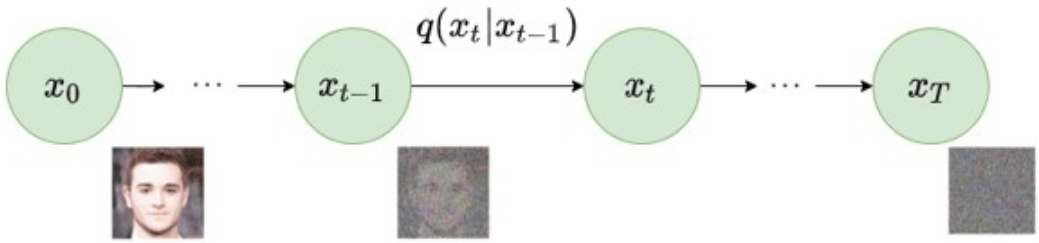

Figure 2.13: Forward diffusion process ([46]).

Thus, it is possible to go in a closed form from the input data x_0 to x_T in a tractable way. Mathematically, this is the posterior probability and it is defined as:

$$q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1}).$$

In order to sample x_t, insted of applying q t times, it is possible to use the already mentioned reparameterization trick. Being $\alpha_t = 1 - \beta_t$, $\bar{\alpha}_t = \prod_{s=0}^{t} \alpha_s$ and $\epsilon_0, ..., \epsilon_{t-1} \sim N(0, \mathbb{I})$, it holds:

$$\begin{aligned} x_t &= \sqrt{1-\beta_t}x_{t-1} + \sqrt{\beta_t}\epsilon_{t-1} \\ &= \sqrt{\alpha_t\alpha_{t-1}}x_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\epsilon_{t-2} \\ &= \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon_0 \end{aligned}$$

where the passage from the first to the second line leverages the fact that the sum of two independent Gaussian random variables remains a Gaussian with mean being the sum of the two means, and variance being the sum of the two variances (a detailed proof is given by [70]). Interpreting $\sqrt{1-\alpha_t}\epsilon_{t-1}$ as a sample from the Gaussian $N(0, (1-\alpha_t)\mathbb{I})$, and $\sqrt{\alpha_t - \alpha_t\alpha_{t-1}}\epsilon_{t-2}$ as a sample from the Gaussian $N(0, (\alpha_t - \alpha_t\alpha_{t-1})\mathbb{I})$, their sum can be considered as a random variable sampled from the Gaussian $N(0, (1-\alpha_t+\alpha_t-\alpha_t\alpha_{t-1})\mathbb{I}) = N(0, (1-\alpha_t\alpha_{t-1})\mathbb{I})$. A sample from this distribution can then be represented, using the reparameterization trick, as $\sqrt{1-\beta_t}x_{t-1}$.
Thus, to produce a sample x_t the following distribution can be adopted:

$$x_t \sim q(x_t|x_0) = N(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)\mathbb{I}).$$

Since β_t is an hyperparameter, α_t and $\bar{\alpha}_t$ can be precomputed for all timesteps. This means that it is possible to sample noise at any timestep t and get x_t in one go. Hence, the latent variable x_t can be sampled at any arbitrary timestep. This

provides also the target in order to calculate the tractable objective loss L_t.
The variance parameter β_t can be fixed to a costant or chosen as a schedule over the T timesteps. In fact, one can define a variance schedule, which can be linear, quadratic, cosine, etc. In [46], authors utilizes a linear schedule increasing from $\beta_1 = 10^{-4}$ to $\beta_T = 0.02$. In [58], authors show that employing a cosine schedule works even better.

2.3.3 Reverse diffusion

As $T \to \infty$, the latent x_T is nearly an isotropic Gaussian distribution. This means that its probability density is equal in every direction. For a Gaussian distribution this can be achieved with a $\sigma^2\mathbb{I}$ covariance matrix. Therefore, learning the reverse distribution $q(x_{t-1}|x_t)$, gives the possibility to sample x_t from $N(0, \mathbb{I})$, run the reverse process and obtain a sample from $q(x_0)$ (generating a novel data point from the original data distribution). Practically, $q(x_{t-1}|x_t)$ in unknown because it is intractable, since statistical estimates of $q(x_{t-1}|x_t)$ require computations involving the data distribution. Therefore, $q(x_{t-1}|x_t)$ is approximated with a parametrized model p_θ (e.g., a neural network). Since $q(x_{t-1}|x_t)$ is also Gaussian, for small enough β_t, p_θ can be chosen to be Gaussian. Its mean and variance can be parametrized as:

$$p_\theta(x_{t-1}|x_t) = N(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)).$$

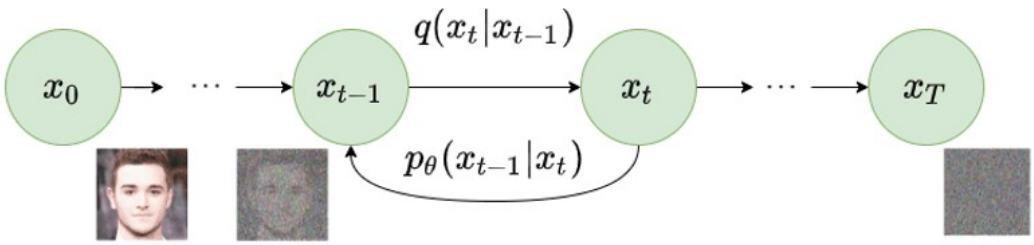

Figure 2.14: Reverse diffusion process ([46])

Applying the reverse formula for all timesteps leads from x_T to the data distribution:

$$p_\theta(x_{0:T}) = p_\theta(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t).$$

By additionally conditioning the model on the timestep t, it learns to predict the Gaussian parameters (the mean $\mu_\theta(x_t, t)$ and the covariance matrix $\Sigma_\theta(x_t, t)$) for each timestep.

2.3.4 Training a diffusion model

Given that the combination of q and p is very similar to a variational autoencoder (VAE), a DDPM can be trained by optimizing the negative log-likelihood of the training data. Remembering the formula 2.2.3, it holds:

$$\begin{aligned}\ln p(x) \geq & \mathbb{E}_{q(x_1|x_0)}[\ln p_\theta(x_0|x_1)] - D_{KL}(q(x_T|x_0)||p(x_T)) - \\ & \sum_{t=2}^{T} \mathbb{E}_{q(x_t|x_0)}[D_{KL}(q(x_{t-1}|x_t, x_0)||p_\theta(x_{t-1}|x_t))] \\ & = L_0 - L_T - \sum_{t=2}^{T} L_{t-1}.\end{aligned}$$

In [46], authors show that the first term can be learned separately and that the second term has no trainable parameters. Then, maximizing the likelihood boils down to learning the denoising steps $L_t, t = 1, ..., T-1$.
Even though $q(x_{t-1}|x_t)$ is intractable, [46] illustrates that by additionally conditioning on x_0 makes it tractable. Intuitively, a painter (the generative model) needs a reference image (x_0) to slowly draw (reverse diffusion step $q(x_{t-1}|x_t, x_0)$) an image. In other words, x_t can be sampled at noise level t, conditioned on x_0.
It is possible to prove ([70]) that:

$$q(x_{t-1}|x_t, x_0) = N(x_{t-1}; \widetilde{\mu}(x_t, x_0), \widetilde{\beta}_t \mathbb{I})$$

where

$$\begin{aligned}\widetilde{\mu}(x_t, x_0) &= \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t} x_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t} x_t \\ \widetilde{\beta}_t &= \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t.\end{aligned}$$

Notice that α_t and $\bar{\alpha}_t$ depend only on β_t, so they can be precomputed. This little trick provides a fully tractable ELBO.
Using the expression of x_0 obtained with the reparameterization trick and substituting it in the equation of $\widetilde{\mu}(x_t, x_0)$ leads to:

$$\widetilde{\mu}_t(x_t) = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon \right), \quad \epsilon \sim N(0, \mathbb{I}).$$

Therefore, a neural network $\epsilon_\theta(x_t, t)$ can be used to approximate ϵ and, consequently, the mean:

$$\widetilde{\mu}_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon_\theta(x_t, t) \right).$$

Thus, the loss function (the denoising term in the ELBO) can be expressed as:

$$L_t = \mathbb{E}_{x_0,t,\epsilon}\left[\frac{1}{2\|\Sigma_\theta(x_t,t)\|_2^2}\|\widetilde{\mu}_t - \mu_\theta(x_t,t)\|_2^2\right]$$
$$= \mathbb{E}_{x_0,t,\epsilon}\left[\frac{\beta_t^2}{2\alpha_t(1-\bar{\alpha}_t)\|\Sigma_\theta\|_2^2}\|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t)\|_2^2\right].$$

This effectively shows that instead of predicting the mean of the distribution, the model predicts the noise ϵ at each timestep t. The authors of [46] make a few simplifications to the actual loss term as they ignore a weighting term. The simplified version outperforms the full objective:

$$L_t^{simple} = \mathbb{E}_{x_0,t,\epsilon}\left[\|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t)\|_2^2\right].$$

According to [46], optimizing the above objective works better than optimizing the original ELBO. Additionally, in [46] authors keep the variance fixed and have the network learn only the mean. This approach is improved by [58], where authors let the network learn the covariance matrix (Σ) as well (by modifying L_t^{simple}), achieving better results.

Algorithm 1 Training algorithm of a DDPM ([46]).

repeat
  $x_0 \sim q(x_0)$
  $t \sim$ Uniform($\{1, ..., T\}$)
  $\epsilon \sim N(0, \mathbb{I})$
  ▷ Take gradient descent step on
  $\nabla_\theta\|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t)\|_2^2$
until converged

Algorithm 2 Sampling algorithm of a DDPM ([46]).

$x_T \sim N(0, \mathbb{I})$
for $t = T, ...,1$ **do**
  if $t > 1$ **then**
    $z \sim N(0, \mathbb{I})$
  else
    $z = 0$
  end if
  $x_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t)\right) + \sigma_t z$
end for
return x_0

2.3.5 Architecture

In order to have model's input and output of similar size, authors of [46] employ a U-Net. A U-Net is a symmetrical architecture with input and output of the same spatial size, that uses *skip connections* between encoder and decoder blocks of corresponding features dimension. Usually, the input image is first downsampled and then upsampled until reaching its initial size. In the original implementation of DDPMs, the U-Net consists of *Wide Residual Network* (WRN), *group normalization* and *self-attention* blocks. The diffusion timestep t is specified by adding a sinusoidal *position embedding* into each residual block.
Next subsections describe more in details the main features of a U-Net, starting from one of the architecture on which U-Net is based.

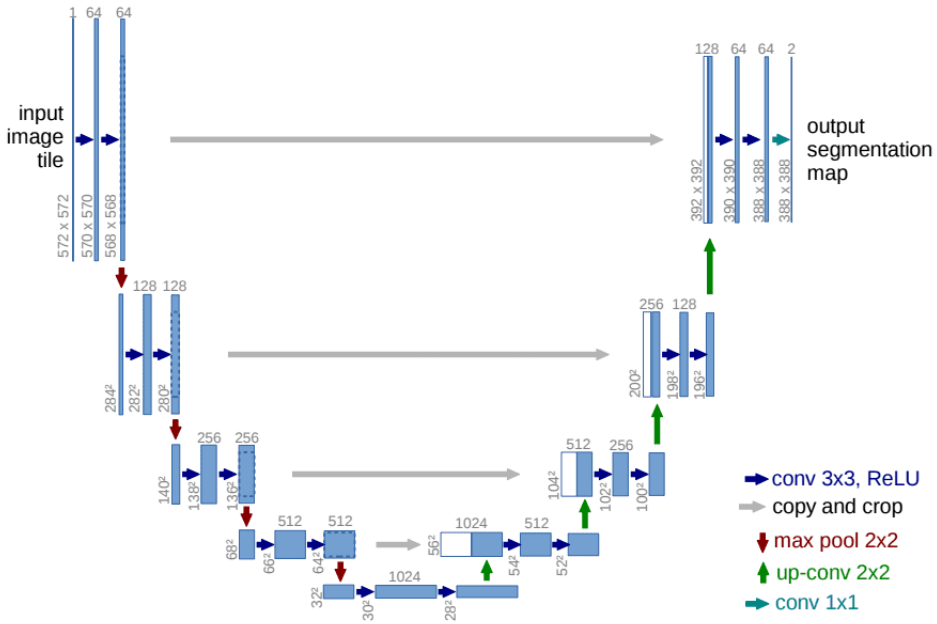

Figure 2.15: The U-Net architecture([17]).

Fully convolutional networks

A *fully convolutional network* (FCN) is one of the first architectures without fully connected layers ([16]). Apart from the fact that it can be trained end-to-end, for individual pixel prediction, it can process arbitrary-sized inputs. It is a general architecture that effectively uses *transposed convolutions* ([81]) as a trainable upsampling method.

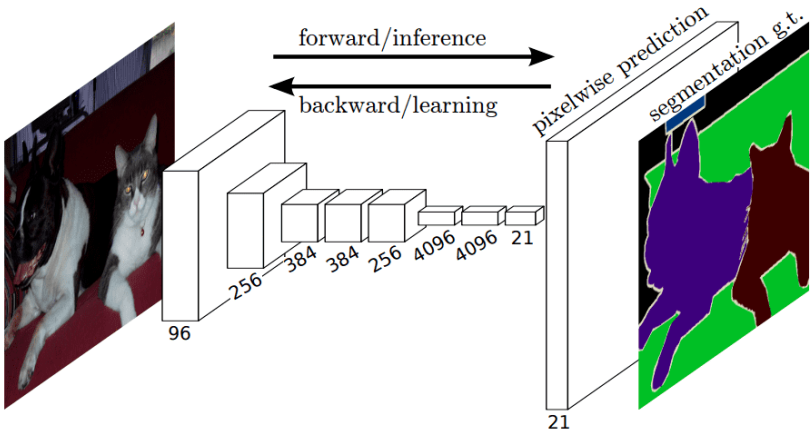

Figure 2.16: The FCN architecture ([16]).

U-Net can be considered an extension of FCN. In particular, the main idea is to make FCN maintain the high-level features in the early layer of the decoder. To this end, in a U-Net architecture *long skip-connections* ([43]) are introduced, in order to localize the segmentations. In this manner, high-resolution features (but semantically low) from the encoder path are combined and reused with the upsampled output.

Encoder and decoder structure

A U-shaped architecture consists of a specific *encoder-decoder* scheme: the encoder reduces the spatial dimensions in every layer while increasing the channels and, on the other hand, the decoder increases the spatial dimensions while reducing the channels.
Specifically, the encoder path consists of the repeated application of two 3×3 convolutions, each one followed by a ReLU and *batch normalization* ([42]). Then a 2×2 *max pooling* ([81]) operation is applied to reduce the spatial dimensions. Again, at each downsampling step, the number of feature channels is doubled, while cutting in half the spatial dimensions. In the decoder path every step consists of an upsampling of the feature map followed by a 2×2 transpose convolution, which halves the number of feature channels. There is also a concatenation with the corresponding feature map from the contracting path, and usually a 3×3 convolution (each followed by a ReLU). At the final layer, a 1×1 convolution is used to map the channels to the desired number of classes.
If not specified, a U-Net architecture processes areas (e.g., two dimensional images).

Conversely, A 3D U-Net, is capable of processing volumes[1].

```
def forward(self, x):
x1 = self.inc(x)
x2 = self.down1(x1)
x3 = self.down2(x2)
x4 = self.down3(x3)
x5 = self.down4(x4)
x = self.up1(x5, x4)
x = self.up2(x, x3)
x = self.up3(x, x2)
x = self.up4(x, x1)
x = self.outc(x)
return x
```

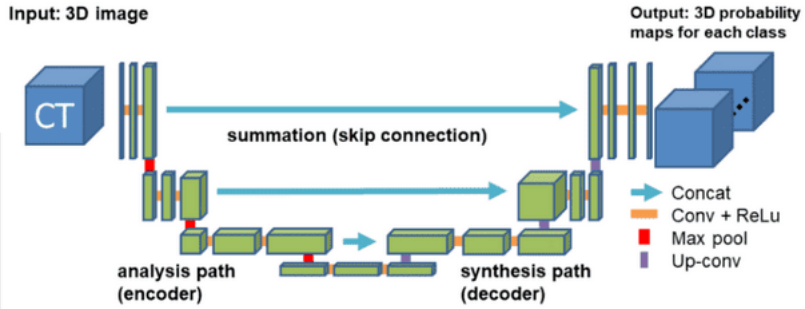

Figure 2.17: A 3D U-Net encoder-decoder structure (source: `https://theaisummer.com/unet-architectures/`).

Skip connections

Skip connections are introduced to solve a plethora of tasks (such as *semantic segmentation*, *optical flow estimation*, etc.). One of the most important task they are asked to solve is alleviating the problem of *vanishing gradient*, arising during backpropagation. Briefly, this issue is related to the fact that while computing all the necessary derivatives of the gradient, the backpropagation algorithm can come up with very small numbers, causing the arrest of the model performance improvement. The reason for this is very simple: often all these derivatives are smaller than one and so, for every layer that the algorithm navigates backward in the network, the gradient of the network gets smaller and smaller.
By using a skip connection, an alternative path for the gradient is provided (with backpropagation). It is experimentally validated that this additional paths are often beneficial for the model convergence. Skip connections in deep architectures, as the name suggests, skip some layers in the neural network and feed the output of one layer as the input to the next layers (instead of only the next one).
In general, there are two fundamental ways that one could use skip connections through different non-sequential layers:

1. *addition*, as in residual architectures, or ResNet ([81]);

2. *concatenation*, as in densely connected architectures, or DenseNet ([81]).

[1]See: `https://theaisummer.com/unet-architectures/`.

In the first case, the core idea is to backpropagate through the identity function, by just using a vector addition. Then, the gradient is simply multiplied by one and its value is maintained in the earlier layers. This is the main idea behind residual networks.

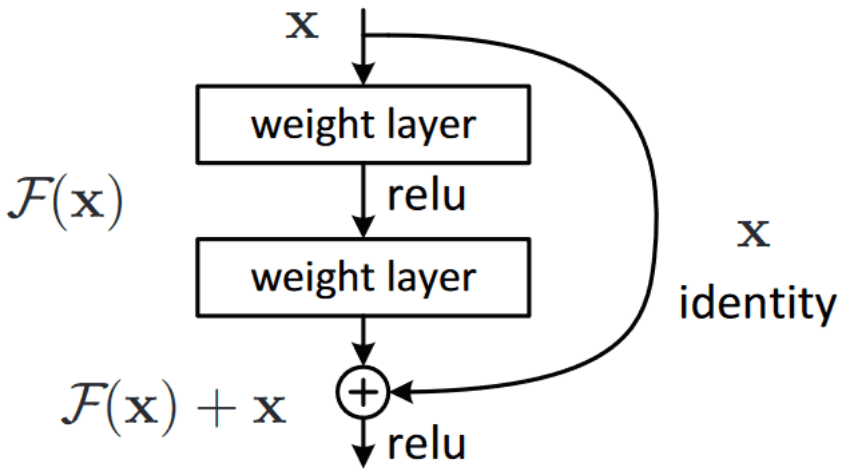

Figure 2.18: An example of skip connections via addition ([15]).

Moreover, additive connections are used in two kinds of setups:

- *short skip connections*, that are used along with consecutive convolutional layers that do not change the input dimension (see ResNet);
- *long skip connections*, that usually exist in symmetrical encoder-decoder architectures, where the spatial dimensionality is reduced in the encoder part and, gradually, increased in the decoder part via *transpose convolutional layers*[2].

Even though there is no theoretical justification, symmetrical long skip connections work incredibly effectively in dense prediction tasks.

[2] `machinecurve.com/index.php/2019/09/29/understanding-transposed-convolutions`.

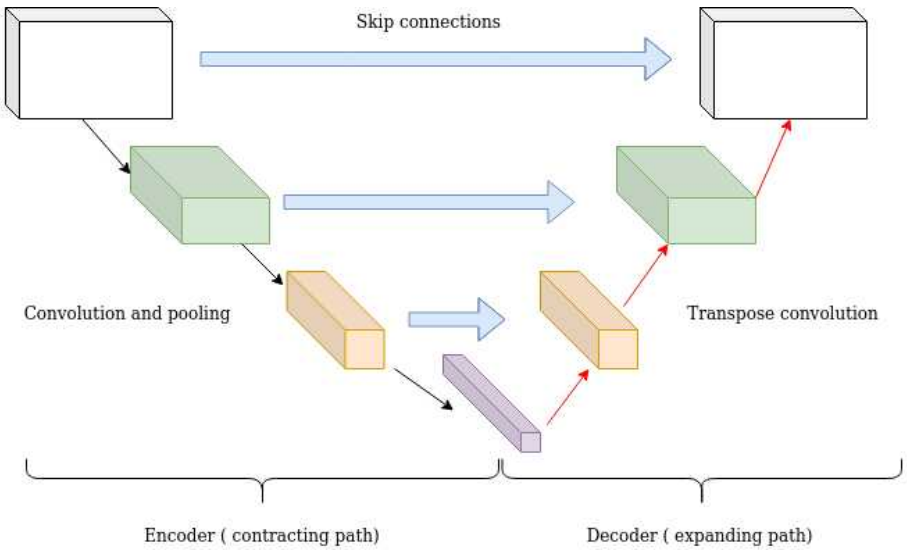

Figure 2.19: An exampe of a symmetrical encoder-decoder architecture provided with long skip connections ([43]).

In the case of concatenated skip connections, the aim is to use features concatenation so as to ensure maximum information flow between layers in the network. This is achieved by connecting via concatenation all layers directly with each other, as opposed to ResNets. Practically, this leads to a concatenation of the features channel dimensions.

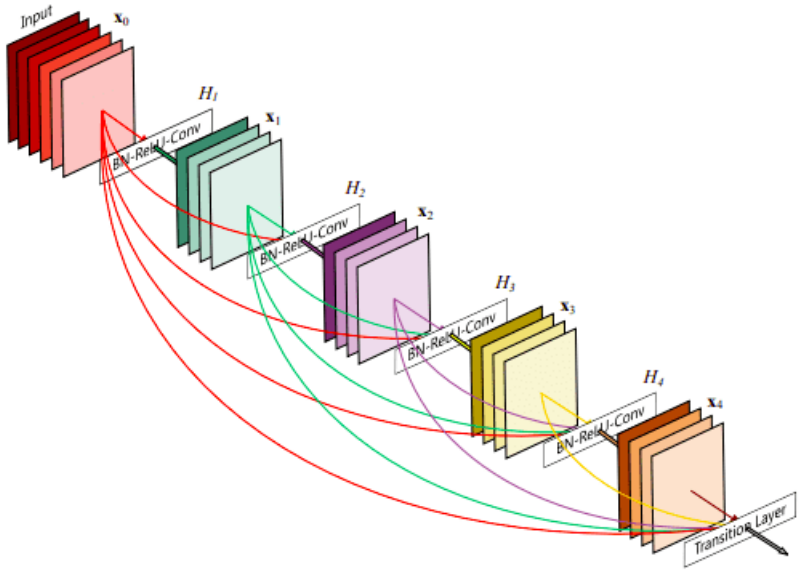

Figure 2.20: An example of skip connections via concatenation ([33]).

Batch normalization and group normalization

Batch normalization (BN) normalizes the mean and standard deviation for each individual feature channel/map.

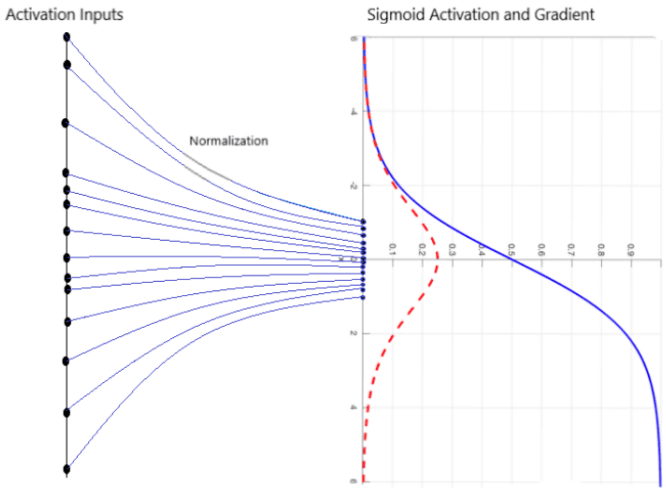

Figure 2.21: Image by MC.AI. It shows how batch normalization brings the values in a compact range.

Specifically, features are demanded to follow a Gaussian distribution with zero mean and unit variance. Mathematically, this can be expressed as

$$\mathrm{BN}(x) = \gamma \left(\frac{x - \mu(x)}{\sigma(x)} \right) + \beta$$

$$\mu_c(x) = \frac{1}{NHW} \sum_{n=1}^{N} \sum_{h=1}^{H} \sum_{w=1}^{W} x_{nchw}$$

$$\sigma_c(x) = \sqrt{\frac{1}{NHW} \sum_{n=1}^{N} \sum_{h=1}^{H} \sum_{w=1}^{W} (x_{nchw} - \mu_c(x))^2}$$

where:

- N is the batch size;
- H, W, C refer, respectively, to the height, the width and the feature channel dimensions;
- $\mu_c(x)$ is the mean of the batch for a specific feature channel;
- $\sigma_c(x)$ is the standard deviation of the batch for a specific feature channel.

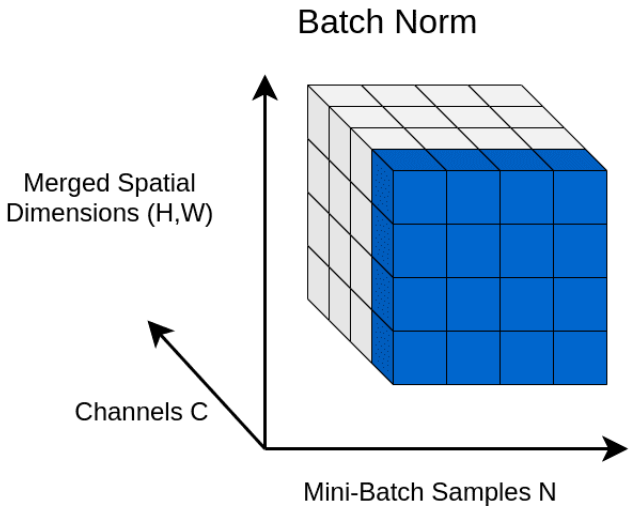

Figure 2.22: An illustration of batch normalization ([42]).

Notably, the spatial dimensions, as well as the image batch, are averaged. In this way, features live in a compact Gaussian-like space, which is usually beneficial. In fact, γ and β correspond to the trainable parameters that result in the linear/affine transformation (introduced with the BN equation above), which is different for all channels. Specifically, γ and β are vectors with the channel dimensionality. The index c denotes the per-channel mean.
BN accelerates the training of deep neural networks, introducing some sort of regularization, and reduces the dependence of gradients on the scale of the parameters. However, it can lead to inaccurate estimation of batch statistics with a small batch size or, in general, when the batch size varies.

Group normalization (GN) divides the channels into groups and computes the first-order statistics within each group. As a result, GN's computation is independent of batch sizes, and its accuracy is more stable than BN in a wide range of batch sizes.
Mathematically:

$$\mu_i = \frac{1}{m} \sum_{k \in S_i} x_k$$

$$\sigma_i = \sqrt{\frac{1}{m} \sum_{k \in S_i} (x_k - \mu_i)^2 + \epsilon}$$

$$S_i = \{k | k_N = i_N, \lfloor \frac{k_C}{C/G} \rfloor = \lfloor \frac{i_C}{C/G} \rfloor \}.$$

Note that m is the number of channels inside a group, while the hyperparameter

G is the number of groups. C/G is the number of channels per group, thus, GN computes μ and σ along the (H, W) axes and along a group of C/G channels.

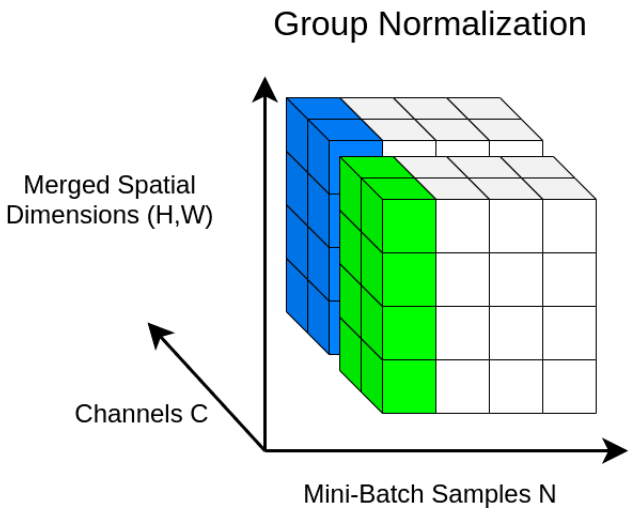

Figure 2.23: An illustration of group normalization. Here, the feature maps are divided into two groups, but the choice is completely arbitrary ([42]).

Attention

The *attention* mechanism emerges naturally from problems that deal with time-varying data, or sequences. *Sequence to sequence* (Seq2seq) learning process works pretty much like in Figure 2.24.

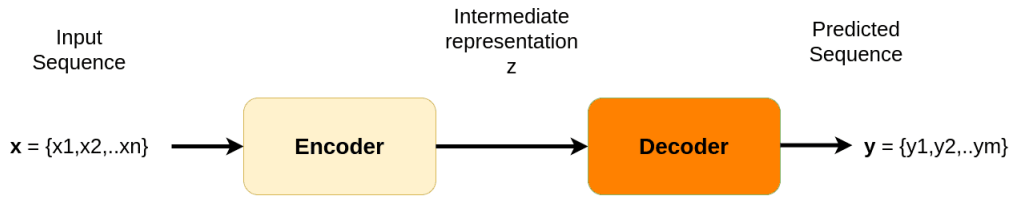

Figure 2.24: Seq2seq learning process representation ([44]).

The elements of the sequence (x1, x2,...,xn in the above figure) are usually called *tokens*, while the encoder and decoder are nothing more than stacked RNN layers. The encoder processes the input and produces one compact representation, called **z**, from all the input timesteps (see Figure 2.25). It can be regarded as a compressed format of the input.

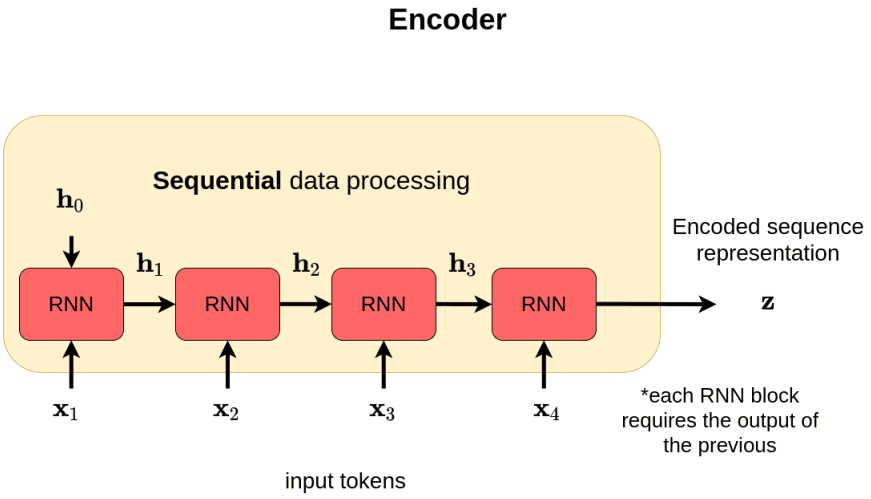

Figure 2.25: Seq2seq encoder representation ([44]).

On the other hand, the decoder receives the context vector **z** and generates the output sequence (see Figure 2.26). The most common application of Seq2seq is language translation. For instance, the input sequence can be the representation of a sentence in English and the output can be the representation of the same sentence in French.

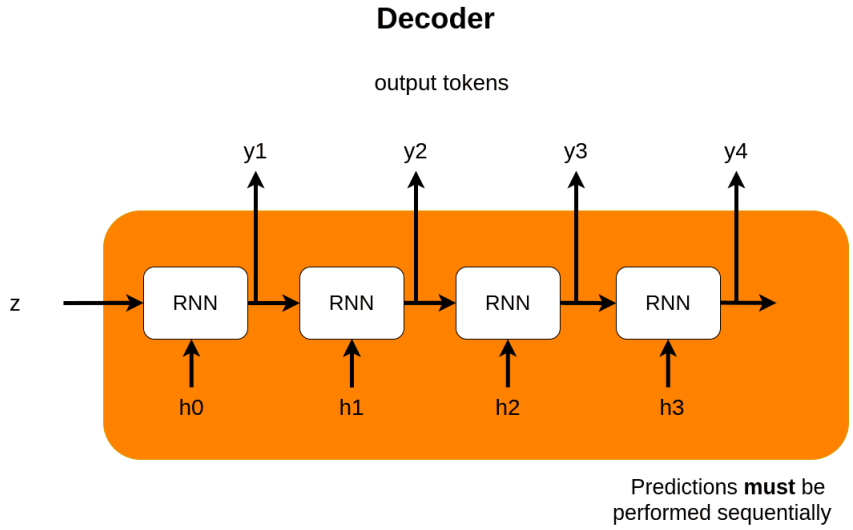

Figure 2.26: Seq2seq decoder representation ([44]).

The attention mechansim addresses two well known limitation of RNN layers in the Seq2seq learning process: early information loss and vanishing gradient. The

core idea is that the context vector **z** should have access to all parts of the input sequence instead of just the last one.
Having said so, a first way to classify attention mechanism can be the following:

- *implicit attention*, realized by all those deep neural networks systems that tend to ignore parts of the input and focus on others. For instance, when working on human pose estimation, the network will be more sensitive to the pixels of the human body. In fact, it is known that many activation units show a preference for human body parts and poses ([23]);
- *explicit attention*, realized by asking the network to weight its sensitivity to the input, based on memory from previous inputs.

Another distinction to make is between *hard* and *soft* attention:

- *soft attention* means that the function identifying the attention mechanism varies smoothly over its domain and, as a result, it is differentiable;
- *hard attention* means that the function identifying the attention mechanism is discrete and so it has many abrupt changes over its domain. Since hard attention is non-differentiable, it can be trained using *Reinforcement Learning*[3] (RL) techniques, instead of the usual gradient descent approach. Once all the sequence tokens are available, it is possible to relax the definition of hard attention, applying a smooth differentiable function to it, to be trained end to end with the usual backpropagation algorithm.

For an RNN encoder-decoder structure, attention mechanism can be placed like shown in Figure 2.27.

[3] `https://theaisummer.com/Policy-Gradients/`.

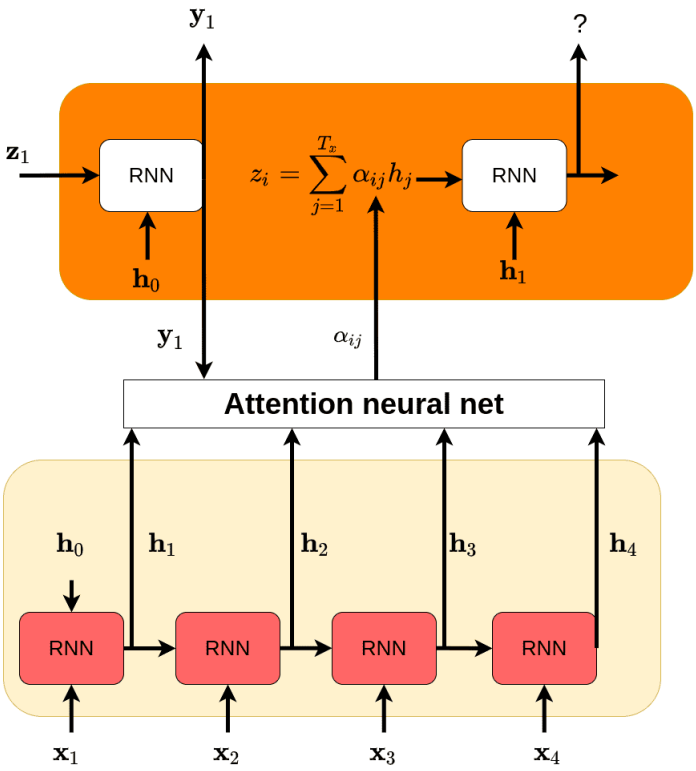

Figure 2.27: Seq2seq with attention mechanism representation ([44]).

In this case,

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}$$

is the softmax function, computed for all the T_x token of the input x. Given the previous state of the decoder, namely y_{i-1}, and an hidden state $h = h_1, h_2, ..., h_n$, the attention mechanism can be identified with

$$e_{ij} = \text{attention}_{\text{net}}(y_{i-1}, h_j) \in \mathbb{R}.$$

Finally, attention mechanism can be also divided into *global* or *local*:

- *global attention* is the result of the attention function computation over the entire input sequence;
- *local attention*, instead, considers only a subset of the input units (tokens), due to computational reasons. It can also be merely seen as hard attention since it requires taking a hard decision first, in order to exclude some input units.

An attention mechanism referred to the same sequence and not to an input-output sequence association is called *self-attention*. In this case, the mechanism looks for scores between the elements of a single sequence, as depicted in Figure 2.28.

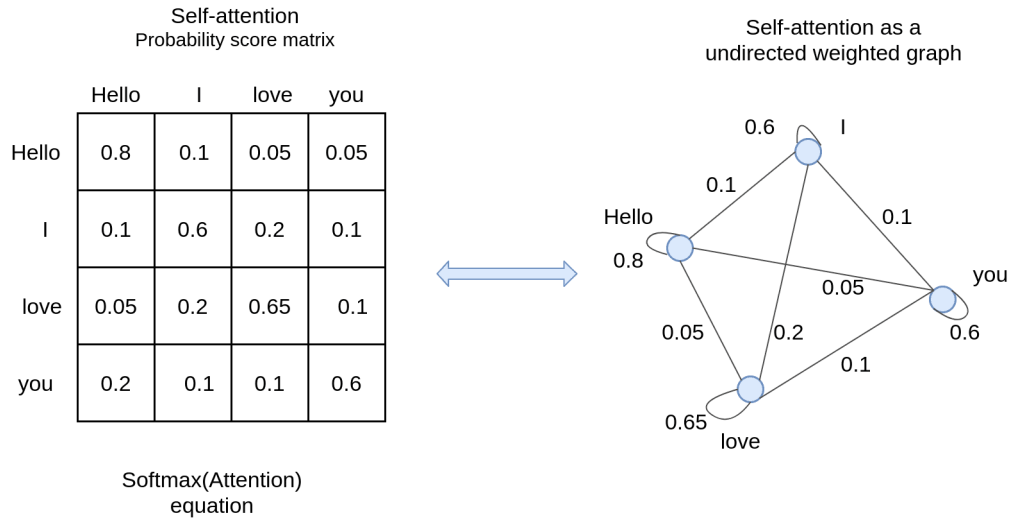

Figure 2.28: An example of a self-attention mechanism, modeled through an undirected weighted graph ([44]).

The self-attention can be computed in any trainable way. The end goal is to create a meaningful representation of the sequence before transforming it.

Positional embeddings

Position embeddings (PE) are trainable mathematical representations that capture the position of each token in the input sequence of a Seq2seq structure. For instance, when dealing with highly-structured data, as images, it is required to incorporate some strong sense of position (order) inside a multi-head self-attention (MHSA) block, whose mechanism, in principle, encodes no positional information ([49]). In order to do so, it is possible to define a map for the input token, introducing three simple objects:

1. **queries** (stored in the tensor Q), indexed with i and representing the token or a position in the sequence;

2. **keys** (stored in the tensor K), indexed with j and representing the features or characteristics of each token in the sequence;

3. **values** (stored in the tensor V), also indexed with j and representing the content or the information associated with each token in the sequence.

If the attention weight is defined as

$$e_{ij} = \frac{x_i W^Q (x_j W^K)^\intercal}{\sqrt{d}}$$

where W^Q and W^Q are, respectively, the linear transformation weight matrices of the input embeddings into queries and keys vectors of dimension d, PE are able to inject some positional information in this way:

$$e_{ij} = \frac{x_i W^Q (x_j W^K)^\intercal + x_i W^Q (p_{ij}^K)^\intercal}{\sqrt{d}}.$$

Notice that the added term represents the distance of the query element from a particular sequence position.
It is often the case that additional positional information is added to the query representation (Q) in the MHSA block. There are two main approaches here, i.e., *absolute* PE or *relative* PE:

- in the *absolute* PE setup, every input token at position i is associated with a trainable embedding vector that indicates the row of a trainable matrix R, with shape $[tokens, dim]$ and initialized in $N(0,1)$. It slightly alters the representation based on the position, resulting in the following attention function:
$$\text{attention} = \text{softmax}\left(\frac{1}{\sqrt{dim}}(QK^\intercal + QR)\right);$$
- *relative* positions represent the distance (number of tokens) between tokens. Information is, again, incorporated inside the MHSA block. The tricky part is that, for n tokens, there are $2n-1$ possible differences and so, now, R has a shape of $[2 \cdot tokens - 1, dim]$.

2.3.6 Guided diffusion

A crucial aspect of image generation is conditioning the sampling process to manipulate the generated samples. In a context of diffusion models, this is also referred to as *guided diffusion*. Mathematically, guidance refers to conditioning a prior data distribution $p(x)$ with a condition y, i.e., the class label or an image/text embedding, resulting in $p(x|y)$.
To turn a diffusion model p_θ into a conditional diffusion model, conditioning information y can be added at each diffusion step:

$$p_\theta(x_{0:T}|y) = p_\theta(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t, y).$$

In general, guided diffusion processes aim to learn $\nabla \ln p_\theta(x_t|y)$. So, using the Bayes rule, it is possible to write:

$$\begin{aligned} \nabla_{x_t} \ln p_\theta(x_t|y) &= \nabla_{x_t} \ln \left(\frac{p_\theta(y|x_t) p_\theta(x_t)}{p_\theta(y)} \right) \\ &= \nabla_{x_t} \ln p_\theta(x_t) + \nabla_{x_t} \ln p_\theta(y|x_t) \end{aligned}$$

where in the second line $p_\theta(y)$ is removed since the gradient operator, in this case, refers only to x_t.
Finally, adding a guidance scalar term s leads to:

$$\nabla \ln p_\theta(x_t|y) = \nabla \ln p_\theta(x_t) + s \nabla \ln p_\theta(y|x_t).$$

The following two families of methods aim at injecting label information.

Classifier guidance

Authors of [18] show that a classifier $f_\phi(y|x_t, t)$ can be trained on the noisy image x_t to predict its class y. This is done in order to guide the diffusion toward the target class y during the training. In this way, it is possible to build a class-conditional diffusion model with mean $\mu_\theta(x_t|y)$ and variance $\Sigma_\theta(x_t|y)$. Since $p_\theta \sim N(\mu_\theta, \Sigma_\theta)$, using the guidance formulation previousely introduced, the mean is perturbed by the gradients of $\ln f_\phi(y|x_t)$, resulting in:

$$\hat{\mu}_t(x_t|y) = \mu_\theta(x_t|y) + s \Sigma_\theta(x_t|y) \nabla_{x_t} \ln f_\phi(y|x_t, t).$$

This idea can be expanded using CLIP embeddings ([59]) to guide the diffusion.

Algorithm 3 Classifier guided diffusion sampling ([51]).

Input: class label y, gradient scale s
$x_T \leftarrow$ sample from $N(0, \mathbb{I})$
for all t from T to 1 **do**
    $\mu, \Sigma \leftarrow \mu_\theta(x_t), \Sigma_\theta(x_t)$
    $x_{t-1} \leftarrow$ sample from $N(\mu + s\Sigma\nabla_{x_t} \ln p_\phi(y|x_t), \Sigma)$
end for
return x_0

Classifier-free guidance

Using the same formulation as before it is possible to define a classifier-free guided diffusion model as:

$$\nabla \ln p_\theta(x_t|y) = s\nabla \ln p_\theta(y|x_t) + (1-s)\nabla \ln p_\theta(x_t).$$

Instead of training a second classifier, in [67] authors train a conditional diffusion model, $\epsilon_\theta(x_t|y)$, together with an unconditional model $\epsilon_\theta(x_t|0)$, using the exact same neural network. During training, they randomly set the class y to 0, so that the model can be exposed to both the conditional and unconditional setup:

$$\hat{\epsilon}_\theta(x_t|y) = s\epsilon_\theta(x_t|y) + (1-s)\epsilon_\theta(x_t|0).$$

There are two main advantages of this approach:

1. it uses only a single model to guide the diffusion;

2. it simplifies guidance when conditioning on y that is difficult to predict with a classifier (such as text embeddings).

2.3.7 Scaled diffusion models

The problem with diffusion models is that they require a lot of computations in order to scale U-Nets into high-resolution images. For this reason, two methods for scaling up diffusion models are usually adopted:

1. cascade diffusion models;

2. latent diffusion models.

Cascade diffusion models

A cascade diffusion model consists of many sequential diffusion models that generate images of increasing resolution. This is done via upsampling the image generated by each model with the addition of higher resolution details. To generate an image, it is possible to sample sequentially from each diffusion model. Data augmentation is crucially in this context because it alleviates compounding error from the previous cascaded models.

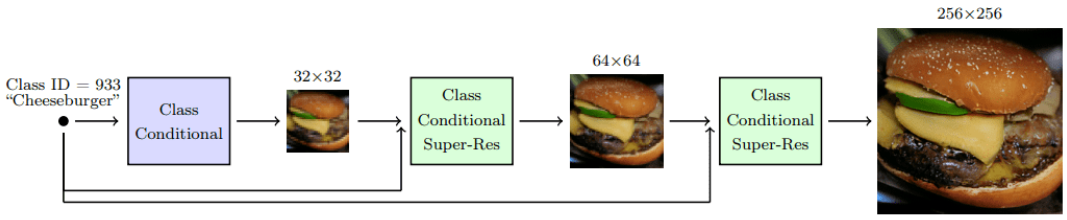

Figure 2.29: Cascade diffusion model pipeline ([55]).

Latent diffusion models

Instead of applying the diffusion process directly on a high-dimensional input, it is possible to project the input into a smaller latent space and apply the diffusion there (*stable diffusion*). In [72] authors propose to use an encoder network to encode the input into a latent representation, i.e., $z_t = g(x_t)$. Afterward, a standard diffusion model (U-Net) is applied to generate new data, which are upsampled by a decoder network.
If the loss for a typical diffusion model (DM) is formulated as

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

then, given an encoder ε and a latent representation z, the loss for a latent diffusion model (LDM) is:

$$L_{\text{LDM}} = \mathbb{E}_{\varepsilon(x),t,\epsilon}\left[\|\epsilon - \epsilon_\theta(z_t, t)\|^2\right].$$

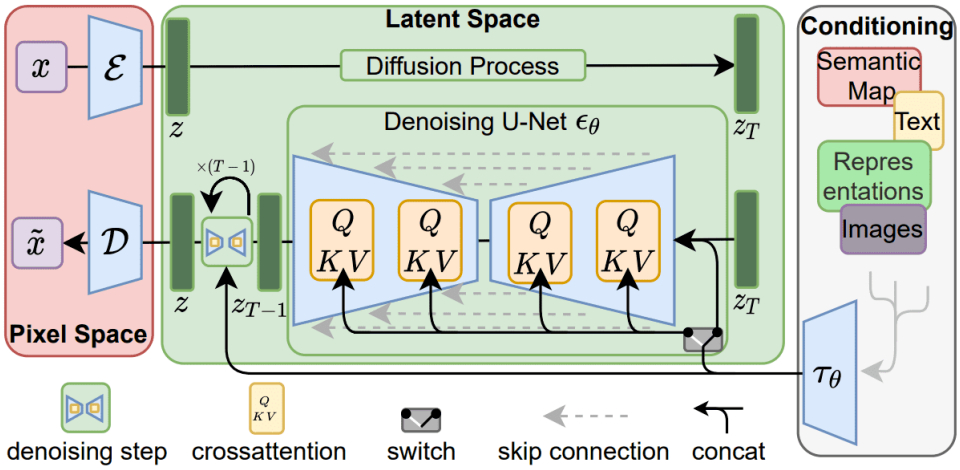

Figure 2.30: Latent diffusion models representation ([72]).

2.3.8 Score-based generative models

Score based models tackle generative learning using *score matching* and *Langevin dynamics*. Score matching refers to the process of modeling the gradient of the ln probability density function, also known as the *score* function ([6]). Langevin dynamics is an iterative process that allows to draw samples from a distribution using only its score function.
In this case, it holds:

$$x_t = x_{t-1} + \frac{\delta}{2}\nabla_x \ln p(x_{t-1}) + \sqrt{\delta}\epsilon$$

where $\epsilon \sim N(0, \mathbb{I})$ and δ is the step size.
Given a probability density $p(x)$ and a score function $\nabla_x \ln p(x)$, it is possible to train a neural network s_θ to estimate $\nabla_x \ln p(x)$ without estimating $p(x)$ first. The training objective can be formulated as follows:

$$\mathbb{E}_{p(x)}\left[\|\nabla_x \ln p(x) - s_\theta(x)\|_2^2\right] = \int p(x)\|\nabla_x \ln p(x) - s_\theta(x)\|_2^2 \, dx.$$

Then, by using Langevin dynamics, sampling directly from $p(x)$ is possible (as [70] shows) thanks to the approximated score function $s_\theta(x)$. Noticee that guided diffusion models use the formulation of score-based models as they learn directly $\nabla_x \ln p(x)$.

Noise Conditional Score Networks (NCSN)

Perturbing data points with noise and training score-based models on the noisy data can improve the accuracy of the estimated score functions in low-density

regions, where few data points are available. As a matter of fact, multiple scales of gaussian noise perturbations can be used. Thus, adding noise is the key to make both DDPMs and score-based models work.

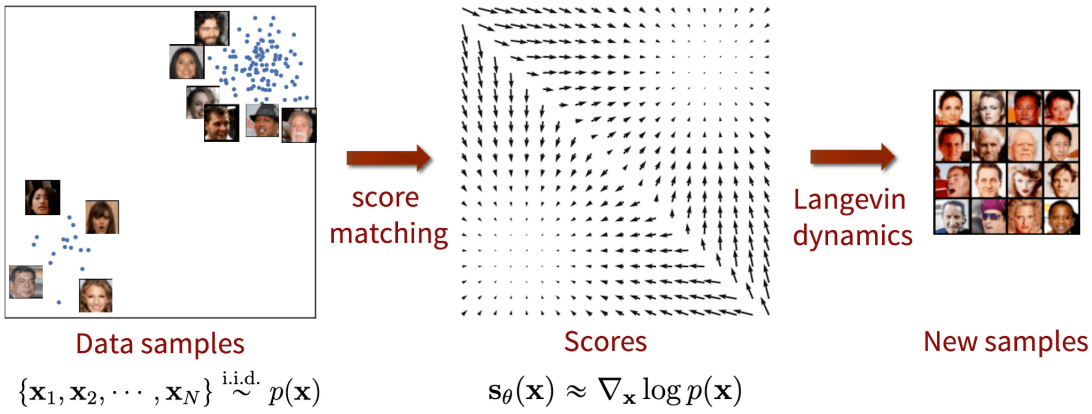

Figure 2.31: Score-based generative modeling with score matching + Langevin dynamics representation ([39]).

Mathematically, a data distribution $p(x)$ is perturbed with a Gaussian noise $N(0, \sigma_i^2 \mathbb{I})$, $i = 1,2,...,L$, in order to obtain a noise perturbed distribution:

$$p_{\sigma_i}(x) = \int p(y) N(x; y, \sigma_i^2 \mathbb{I})\, dy.$$

Then, a network $s_\theta(x, i)$, known as *Noise Conditional Score-Based Network* (NCSN), is trained to estimate the score function $\nabla_x \ln p_{\sigma_i}(x)$. The training objective is a weighted sum of Fisher divergences for all noise scales:

$$\sum_{i=1}^{L} \lambda(i) \mathbb{E}_{p_{\sigma_i}(x)} \left[\| \nabla_x \ln p_{\sigma_i}(x) - s_\theta(x, i) \|_2^2 \right]$$

where $\lambda(i)$ is a weighting factor.
In information geometry, the Fisher information metric is a particular Riemannian metric which can be defined on a smooth statistical manifold, i.e., a smooth manifold whose points are probability measures defined on a common probability space. It can be used to calculate the informational difference between measurements.

Score-based generative models through SDE

In [61] authors propose to perturbate data using a continuum of distributions that evolve over time according to a diffusion process. This process is modelled by a prescribed stochastic differential equation (SDE) that does not depend on the data

and has no trainable parameters. As always, by reversing the process it is possible to generate new samples.

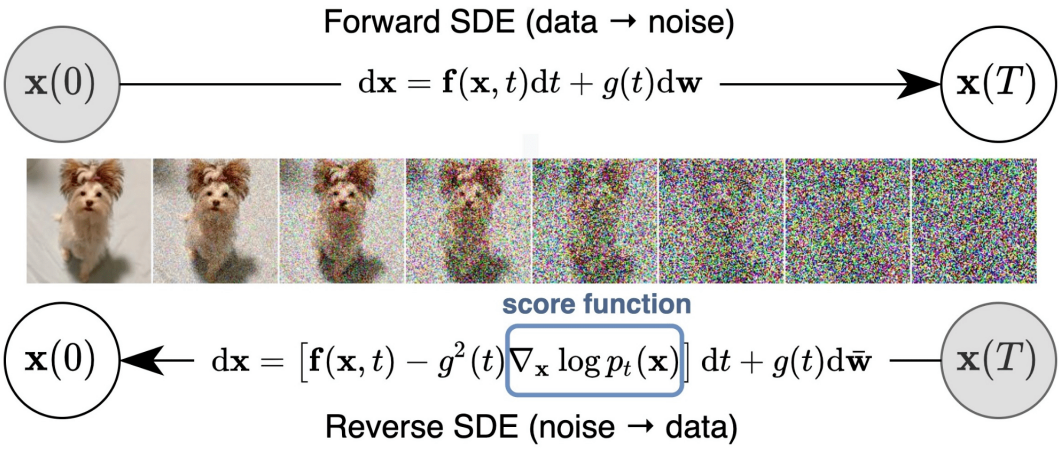

Figure 2.32: Score-based generative modelling through stochastic differential equations ([61]).

The diffusion process $\{x(t)\}_{t\in[0,T]}$ can be defined as an SDE in the following form:

$$dx = f(x,t)dt + g(t)dw$$

where:

- w is the Wiener process (described in [4]);
- $f(\cdot,t)$ is a vector valued function known as *drift coefficient* of $x(t)$;
- $g(\cdot)$ is a scalar function known as *diffusion coefficient* of $x(t)$.

After perturbing the original data distribution for a sufficiently long time, the perturbed distribution becomes close to a tractable noise distribution. To generate new samples, it is necessary to reverse the diffusion process. The SDE is chosen so that it has a corresponding reverse SDE ([1]) in closed form:

$$dx = \left[f(x,t) - g(t)^2\nabla_x \ln p_t(x)\right] dt + g(t)dw.$$

To compute the reverse SDE, it is necessary to estimate the score function $\nabla_x \ln p_t(x)$. This is done using a score-based model $s_\theta(x,t)$ and Langevin dynamics.
The training objective is a continuous combination of Fisher divergencies

$$\mathbb{E}_{t\in U(O,T)}\mathbb{E}_{p_t(x)}\left[\lambda(t)\|\nabla_x \ln p_t(x) - s_\theta(x,t)\|_2^2\right]$$

where $U(0,T)$ denotes a uniform distribution over the time interval and $\lambda(t)$ is a positive weighting function. Once obtained the score function, it is possible to solve the reverse SDE in order to sample $x(0)$ from the original data distribution $p_0(x)$.

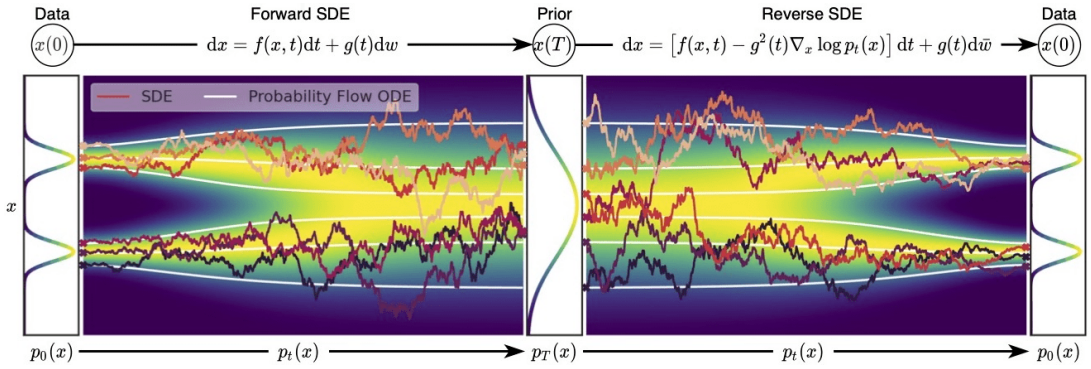

Figure 2.33: Overview of score-based generative modelling through SDEs ([61]).

2.4 Multimodalities involving music

The field of generative art includes various works that investigate models' potential on music data.
In [9], [40], [41], [57] and [19] authors focuse on the possibility of establishing different kinds of correlations between audio and visual data. In particular, authors of [40] introduce the problem of learning affective correspondences between audio (music) and visual data (images). For this task, a music clip and an image are considered similar (having true correspondence) if they have similar emotion content. In order to estimate this crossmodal, emotion-centric similarity, authors propose a deep neural network architecture that learns to project the data from the two modalities into a common representation space. Then, the network performs a binary classification task in order to predict the affective correspondence (*true* or *false*). The proposed approach achieves 61.67% accuracy for the affective correspondence prediction task on a customized database, containing more than 3500 music clips and 85000 images with three emotion classes (*positive*, *neutral* and *negative*).
In [25], [11] and [10], instead, authors explore the topic of *cross matching* between music and image. In particular, authors of [25] aim to understand whether and how music and images can be automatically matched by machines. They costruct a benchmark dataset composed of more than 45000 music-image pairs and recruit human labelers to annotate whether these pairs are well-matched or not. Results show that labelers generally agree with each other on the matching degree of music-image pairs. Secondly, a suitable semantic representation of music and image

for this cross-modal matching task is investigated: authors adopt lyrics as a middle-media to connect music and image, and design a set of lyrics-based attributes for image representation. In the end, they propose a cross-modal ranking analysis (CMRA) to learn the semantic similarity between music and image, outperforming state-of-the art cross modal methods in the music-image matching task.
Works like [66], [82], [29], [28] and [77] delve very deeply into the generative part of visual and musical data. Indeed, in [77] authors propose an audio-video generation framework that enhances engaging viewing and listening experiences simultaneously, towards high quality realistic videos. To generate joint audio-video pairs, a novel Multi-Modal Diffusion (i.e., MM-Diffusion) model is leveraged, together with two-coupled denoising autoencoders. Extensive experiments show very promising results in unconditional audio-video and zero-shot conditional tasks (e.g., video-to-audio).
Zero-shot learning (ZSL) is a problem setup in deep learning where, at test time, a learner observes samples from classes which were not observed during training, and needs to predict the class that they belong to. In this respect, another significant work dealing with zero-shot classification of music and images is the one done by Wu et al. ([74]).
Even if not totally focused on the use of music data, the research conducted by Yariv et al. in [80] propose a framework for audio-to-image generation: given an audio sample containing an arbitrary sound, authors aim to generate a high quality image representing the acoustic scene. Specifically, inspired by Gal et al. ([65]), they propose to learn a dedicated audio-token that can map the audio representation into an embedding vector. Such a vector is then forwarded into the network as a continuous representation, reflecting a new word embedding. In this way, authors can investigate the feasibility of directly encoding any audio signal into a dedicated representation that produces an additional token for a text-conditioned image generation.
Finally, among many articles inspired by the work on *Contrastive Language-Image Pre-training* (CLIP [60]), that paves the way for an entire new category of AI multimodal approaches, it is worth mentioning the one carried out by Guzhov et al. ([53]), for its contribution to the improvement of sounds classification. In particular, here, authors present an extension of the CLIP model that handles audio in addition to text and images: a model that incorporates the ESResNeXt audio-model ([54]) into the CLIP framework using the AudioSet dataset[4]. This dataset is a large-scale collection of human labeled 10-second sound clips drawn from YouTube videos. Such a combination enables the proposed model to perform both unimodal and bimodal classification as well as querying, while keeping CLIP's ability to generalize to unseen datasets in a zero-shot inference fashion.

[4] `https://research.google.com/audioset/dataset/index.html`.

Chapter 3

Dataset

The following sections describe the sources and the construction phases of the music-image dataset used for the artistic image generation task from music inputs.

3.1 Data sources

Sources of music and artistic images are 4 existing datasets (i.e., *WikiArt* [89], *A Refined WikiArt Dataset* [35], *Best Artworks of All Time* [36], and *Classical Asian Art* [84]) and 2 websites (*Pinterest* [85] and *YouTube* [90]).

3.1.1 Datasets

WikiArt

WikiArt website ([89]) contains one of the richest collection of figurative art available online. The website is structured to provide an extensive and accessible platform for exploring art across various periods, styles, and artists. Thanks to a comprehensive collection of artworks, ranging from classical to contemporary, it includes a large variety of paintings, sculptures, and other forms of visual art. Artworks are categorized according to:

- periods and movements (such as *Renaissance*, *Baroque*, *Romanticism*, *Impressionism*, *Modernism*, etc.);

- styles and genres (such as *portrait*, *landscape*, *still life*, *abstract*, *surrealism*, etc.).

Moreover, each artist featured on WikiArt has a dedicated page showcasing their biography, notable works, and contributions to the art world. Artist pages often include images of their artworks, a brief overview of their career, and links to

related articles or external resources for further exploration. The website offers robust search functionality, allowing users to search for artworks, artists, or specific keywords. This makes it easier to find particular artworks or explore themes across different artists and periods. Users can also filter search results by criteria such as medium (e.g., oil on canvas, watercolor), size, location, and more, to refine their exploration based on specific preferences. Images on WikiArt are available in JPEG or PNG format.

A Refined WikiArt Dataset

This source of artistic images adopted is a refined dataset ([35]), containing images from WikiArt website. It is organized into 29 unbalanced categories, each one identified with a label expressing an art figurative genre, from *Abstract_Expressionism* to *Ukiyo_e*. Overall, there are 25000 images in JPEG format, requiring a storage capacity of 8 GB. Authors provide 2 csv files containing, respectively, the lists of training and validation images and a text file containing the list of the dataset classes.

Best Artworks of All Time

This dataset ([36]) contains a collection of artworks from the 50 most influential figurative artists of all time. Data are organized into unbalanced folders according to artists' names, from *Albrecht_Durer* to *William_Turner*, for a total of 16800 images in JPEG format. This dataset requires a storage capacity of 2 GB and is equipped with a csv file, providing general informations for each artist, and a folder containing resized data.

Classical Asian Art

This last dataset contains 138 public images, in JPEG format, mainly related to Asian traditional art, and taken from Artvee[1].

3.1.2 Websites

Pinterest

Pinterest ([85]) is a visual discovery and bookmarking platform that allows users to find, save, and share ideas and inspiration across a wide range of interests and topics. The site is structured to provide an engaging and user-friendly experience, encouraging users to explore and curate contents. Pinterest offers robust search

[1] `https://artvee.com/`.

functionality, allowing users to search for specific topics, keywords, or hashtags. Search results can be filtered by:

- *pins*, i.e, images or videos that users save to their account;
- *boards*, i.e., collections of pins organized by themes or topics;
- *people*, i.e., users of the platform.

Images on Pinterest are available in JPEG or PNG format.

YouTube

YouTube ([90]) is a video-sharing platform where users can upload, view, and interact with videos. The site is structured to provide an engaging and user-friendly experience, encouraging users to explore, create, and connect through video material. YouTube offers a powerful search functionality, allowing users to search for specific videos, channels, or keywords. Search results can be filtered by upload date, view count, rating, and more to help users find exactly what they are looking for. The vast number of music-related videos available on YouTube makes it one of the largest platforms for easily sharing music content among users.

3.2 Data preprocessing

The fact that WikiArt ([89]) and Best Artworks of All Time ([36]) classify artworks according to specific artistic and historical criteria partially reflects in the categorization choices for the music-image dataset.

3.2.1 Images

The image collection is organized according to the following semantic criteria:

- selected images of visual art from the early Renaissance to Classicism are split into *symbolic* and *secular*;
- selected images of visual art from Romanticism to the 20th century are split into *figure* and *open view*;
- selected images of visual art from the 20th century are additionally split into *experimental* and *traditional*;
- selected images of visual art related to the traditional folklore of a specific territory are considered *folk*;

- selected images of visual art semantically related with music from 20th and 21st centuries are split into *instrumental*, *electronic* and *heavy metal*.

The entire dataset of images consists of 24216 units, all transformed in JPEG format, distributed among 18 balanced (in terms of numerosity) classes. The training set has 19366 units, while the validation and test set have, respectively, 2425 units.
Figure 3.1 shows the 18 music-inspired labels representing each class of the image dataset.

Figure 3.1: Graphic representation of music-inspired labels for the image dataset.

3.2.2 Music

Music data is obtained extracting audio files from YouTube videos and requires a storage capability of around 170 GB. This outcome is due to two main reasons:

- it is fundamental to obtain a music dataset as various as possible, in order to let the model learn several kinds of music;
- as already mentioned, following the same approach of Chen et al. ([45]), to obtain 30 seconds of audio samples in mp3 format, additional storage space is required beyond what is needed to store the entire pieces of music.

Selection and categorization of music is based on semantic criteria described by the following image-music association rules:

- images of symbolic visual art from early Renaissance to Classicism $\Longrightarrow$ religious music from early Renaissance to Classicism;
- images of secular visual art from early Renaissance to Classicism $\Longrightarrow$ secular music from early Renaissance to Classicism;
- images of visual art figures from Romanticism to the 20th century $\Longrightarrow$ mainly instrumental and chamber music from Romanticism to the 20th century;
- images of visual art depicting open views from Romanticism to the 20th century $\Longrightarrow$ orchestral and opera music from Romanticism to the 20th century;
- images of experimental visual art of the 20th century $\Longrightarrow$ experimental music of the 20th century;
- images of traditional visual art of the 20th century $\Longrightarrow$ traditional music (academic, reactionary, tonal, etc.) of the 20th century;
- images of traditional folk visual art from the past times to the present day $\Longrightarrow$ traditional folk music from the past time to the present day;
- images of visual art inspired by 20th and 21st centuries musical genres $\Longrightarrow$ music from the main 20th and 21st centuries musical genres (rock, jazz, pop, country, rap, electronic, etc.).

The music dataset contains 24216 30 seconds music samples, all transformed in wav format, and divided into the same 18 categories as the images. Every audio file name is standardized to a common encoding (*Unicode*). The training set consists of 19366 units, while the validation and test set have, respectively, 2425 units.

LABEL	MUSICAL GENRES	MAIN FEATURES OF THE CHOSEN MUSIC	DESCRITPION OF THE CHOSEN ART IMAGES	WIKI-ART CATEGORIES INVOLVED	CLASS DIMENSION
Renaissance symbolism	Pre-Renaissance (380) Flemish (362) Renaissance (508)	Symbolic Religious	A collection of paintings from pre-Renaissance, Flemish and Renaissance art movements, expressing symbolic concepts (religion, allegories, mythology, etc.).	Proto Renaissance Early Renaissance High Renaissance Flemish school Religious painting Allegorical painting Symbolic painting 14th century 15th century 16th century	1250
Renaissance secular subject	Pre-Renaissance (177) Flemish (372) Renaissance (701)	Secular	A collection of realistic paintings from pre-Renaissance, Flemish and Renaissance art movements (portraits, battles, everyday life, etc.).	Early Renaissance High Renaissance Late Renaissance Flemish school Portrait Genre painting Battle painting 14th century 15th century 16th century	1250
Baroque Classical symbolism	Baroque (694) Classic (556)	Symbolic Religious	A collection of paintings from Baroque and Classical art movements, expressing symbolic concepts (religion, allegories, mythology, etc.).	Baroque Rococo Flemish Neoclassicism Religious painting Allegorical painting Symbolic painting 17th century 18th century	1250
Baroque Classical secular subject	Baroque (854) Classic (405)	Secular	A collection of realistic paintings from Baroque and Classical art movements (portraits, battles, everyday life, etc.).	Baroque Rococo Flemish Neoclassicism Portrait Genre painting Battle painting 17th century 18th century	1259
Romanticism figure	Romantic (667) Late Romantic (876)	Chamber Instrumental	A collection of paintings from Romanticism and late-Romanticism art movements, depicting both living and inanimate subjects (portraits, battles scenes, celebrations, still lives, animals, etc.).	Romanticism Portrait Genre painting Battle painting Still life Animal painting Symbolic painting 19th century	1543

Romanticism open view	Romantic (479) Late Romantic (771)	Orchestral Opera	A collection of paintings from Romanticism and late-Romanticism art movements, depicting open views (landscapes, woods, mountain ranges, marinas, cities, wide spaces, etc.).	Romanticism Landscape Marina CityscapeImpressionism Impressionism Expressionism 19th century	1250
20th century traditional figure	Traditional 19th – 20th c. (582) Ballet 19th-20th c. (300) Traditional 20th c. (584)	Chamber Instrumental Theatre Orchestral	A collection of paintings from the late 19th and 20th centuries, primarily realistic, depicting both living and inanimate subjects (portraits, battles scenes, celebrations, still lives, animals, etc.).	Portrait Genre painting Battle painting Still life Animal painting Impressionism Expressionism Landscape Marina Cityscape 19th century 20th century	1466
20th century traditional open view	Traditional 19th – 20th c. (680) Traditional 20th c. (664)	Orchestral Opera	A collection of paintings from the late 19th and 20th centuries, primarily realistic, depicting open views (territories, woods, mountain ranges, marinas, cities, wide spaces, etc.).	Impressionism Expressionism Landscape Marina Cityscape 19th century 20th century	1344
20th century experimental figure	Experimental 19th – 20th c. (380) Experimental early 20th c. (992)	Chamber Instrumental Avant-garde	A collection of paintings from the late 19th and 20th centuries, primarily unrealistic, depicting both living and inanimate subjects (portraits, battles scenes, celebrations, still lives, animals, etc.).	Portrait Genre painting Still life Animal painting Impressionism Expressionism Surrealism 19th century 20th century	1372
20th century experimental open view	Experimental 19th – 20th c. (545) Experimental early 20th c. (712)	Orchestral Opera Avant-garde	A collection of paintings from the late 19th and 20th centuries, primarily unrealistic, depicting open views (territories, woods, mountain ranges, marinas, cities, wide spaces, etc.).	Impressionism Expressionism Landscape Marina Cityscape Surrealism Tonalism Post-Impressionism 19th century 20th century	1257

20th century abstractionism	Experimental 20th c. (1494)	Orchestral Opera Chamber Avant-garde	A collection of abstract paintings from the end of 19th and 20th centuries.	Cubism Avant-garde Dada Abstract art Metaphysical art Conceptual art Minimalism Bauhaus 20th century	1494
European traditional folklore	North-Centre Europe folk (388) Est-Europe folk (499) Mediterranean Europe folk (363)	Vocal Instrumental Ritualistic Religious	A collection of European folkloristic art pictures.	Naive art Arte povera Pop art Caricature Folk art Realism Regionalism Neo-Pop art Contemporary Sketch and study 20th century 21st century	1250
Asian traditional folklore	Middle East, South Asia folk (519) Cina, Japan, South-East Asia folk (819)	Vocal Instrumental Ritualistic Religious	A collection of folkloristic art picture, realised by Asian artists throughout history.	Bangladeshis Bengalis Chinese Emiratis Indians Indonesians Iranians Japaneses Outsider art Native art Sketch and study Folk art	1338
American traditional folklore	North America folk (373) Central, South America folk (1061)	Vocal Instrumental Ritualistic Religious	A collection of American folkloristic art pictures.	Argentineans Brazilians Chileans Colombians Cubans Dominicans Ecuadorians Guatemalans Indigenous North Americans Mexicans Peruvians Puerto Ricans Venezuelans Folk art	1434

African traditional folklore	African folk (625) Afro music (627)	Vocal Instrumental Ritualistic Religious Worldwide	A collection of African and afro-cultural folkloristic art pictures.	Algerians Angolans Cameroonians Egyptians Ethiopians Ancient Egypt Kenyans Libyans Moroccan Namibians Nigerians South Africans Outsider art Native art Sketch and study Folk art	1252
Modern instrumental music	Jazz (281) Rock (371) Western songwriters (231) Country (115) Latin American (260)	Worldwide Instrumental Popular	A collection of historical and contemporary art pictures, aimed at summarizing the main concepts expressed by instrumental musical genres primarily developed in western countries (jazz, rock, country, etc.).	Pop art Contemporary Sketch and study	1258
Electronic music	Electronic (538) Dance (344) Pop (516) Rap, Hip-hop (204)	Worldwide Informatic Ambient Digital Popular	A collection of historical and contemporary art pictures, aimed at summarizing the main concepts expressed by musical genres like electronic, dance, pop, rap and hip hop.	Illustration Digital art Street art Graffiti Pop art	1602
Heavy metal	Heavy metal (1347)	Instrumental Worldwide	A collection of historical and contemporary art pictures, aimed at summarizing the main concepts expressed by heavy metal music.	Fantasy art Poster	1347

Table 3.1: Informations about the customized image-music dataset. Images and music samples are assigned to various subcategories. Numbers between brackets correspond to the numerosity of each subcategory.

Chapter 4

Methodology

The architecture ([80]) used to generate artistic images from music inputs, as well as its adaptation to the music-image dataset previousely described, are provided in the following sections.

4.1 Description of the model

The model proposed by Yariv et al. ([80]) and adopted to generate artistic images from music inputs relies on an audio-to-image generation method that leverages a pre-trained text-to-image diffusion model ([72]), together with a pre-trained audio encoder ([64]). In particular, the learning process of the pre-trained diffusion is conducted on a latent representation of an encoder-decoder architecture, because Rombach et al. ([72]) show that this strategy can produce higher quality results than an approach based on raw inputs.
Given a data distribution $p(x)$, a reverse Markov process of length T, a timestamp $t \in [0,1]$, a denoising function $\epsilon_\theta : \mathbb{R}^d \to \mathbb{R}^d$ that learns predicting a clean version of the perturbed x_t from the training distribution $S = \{x_1, ..., x_m\}$, latent diffusion operates on top of a representation given by an encoder f, yielding the following loss function:

$$\mathcal{L}_{\text{LDM}} \triangleq \mathbb{E}_{x\sim S, t\sim U(0,1), \epsilon\sim N(0,\mathbb{I})} \left[\|\epsilon - \epsilon_\theta(f(x_t), t)\|_2^2 \right].$$

The output of the diffusion can later be forwarded through the decoder to obtain the raw result (e.g., audio, image, text).
An important component of modern generative models is conditioning. This allows the generative process to be conditioned on a given input, i.e., modeling $p(x|y)$, where y is a data entry. Usually, the conditioning component is done by an injection of a condition representation from an encoder τ to the attention mechanism of ϵ_θ.

Conditioning the diffusion process yields the following loss function:

$$\mathcal{L}_{\text{CLDM}} \triangleq \mathbb{E}_{(x,y)\sim S, t\sim U(0,1), \epsilon\sim N(0,\mathbb{I})} \left[\|\epsilon - \epsilon_\theta(f(x_t), t, \tau(y))\|_2^2 \right].$$

The input to the method is a pair (i, a), where i represents an artistic image and a represents its corresponding music sample. The final purpose is to create a generative process that is audio-conditioned, i.e., $p(i|a)$. In order to do so, given that a text-conditioned generative model is being used, the audio signal a has to be associated with a text conditioning.
Figure 4.1 shows the entire architecture adopted to fulfill the aforementioned task.

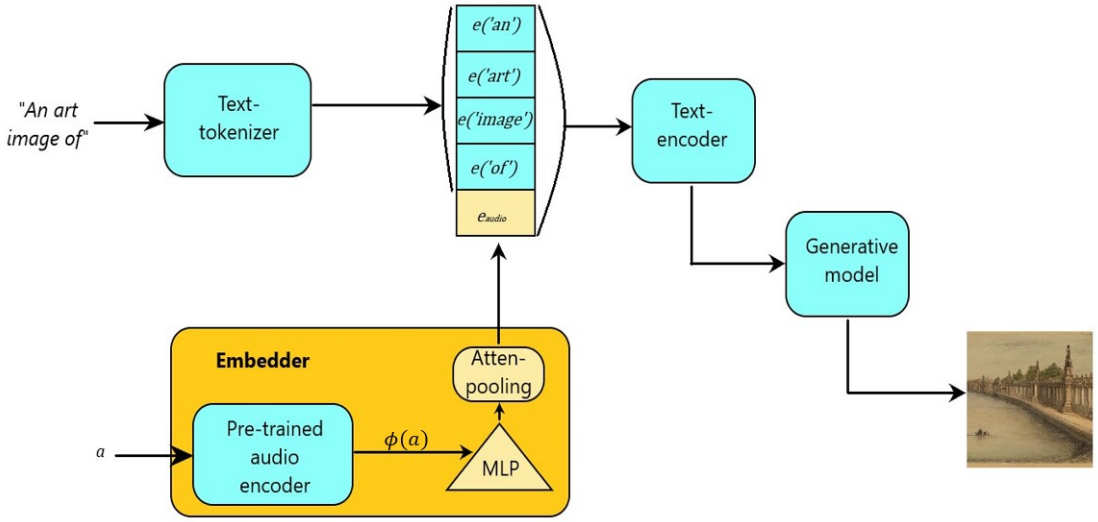

Figure 4.1: Architecture overview: a 30 seconds music sample is forwarded through a pre-trained audio encoder and then through a projection network. A pre-trained text encoder extracts tokens from vector representations of the tokenized prompt and the audio. Finally, the generative model is fed with the concatenated representations of these tokens.

The process begins with a transformer model that encodes the initial prompt "An art image of" into a representation $e_{text} \in \mathbb{R}^{4\times d_a}$, where d_a is the embedding dimension of the text input. Afterward, e_{text} is concatenated to an extra latent representation of the audio signal, denoted as $e_{audio} \in \mathbb{R}^{d_a}$.

4.1.1 Audio encoding

To obtain e_{audio}, an Embedder, composed of a pre-trained audio encoding network ([64]) and a small projection network, is adopted. This results in:

$$e_{audio} = \text{Embedder}(a).$$

The pre-trained audio encoder is based on an iterative audio pre-training framework to learn Bidirectional Encoder representation from Audio Transformer (BEATs), where an acoustic tokenizer and an audio Self Supervised Learning (SSL) model are optimized by iteration ([64]).
The Embedder leverages a pre-trained audio classification network ϕ to represent the audio. The discriminative network's last layer is typically used for classification, and thus it tends to diminish important audio informations which are irrelevant to the discriminative task. Thus, a concatenation of earlier layers and the last hidden layer (specifically selecting the 4th, 8th, and 12th layers out of a total of 12) is taken. This results in a temporal embedding of the audio $\phi(a) \in \mathbb{R}^{\hat{d} \times n_a}$, where n_a is the temporal audio dimension, and $\hat{d}$ is the new dimension obtained from the concatenation of the layers.
Then, to learn a projection into the textual embedding space, $\phi(a)$ is forwarded in two linear layers, $W_1 \in \mathbb{R}^{\hat{d} \times \hat{d}}$ and $W_2 \in \mathbb{R}^{\hat{d} \times d_{audio}}$, with a GELU non-linear function σ between them:

$$\bar{e}_{audio} = W_2\sigma(W_1\phi(a)).$$

Standing for Gaussian Error Linear Unit, the GELU function is an activation function defined as: $\text{GELU}(x) = x\Phi(x)$, where $\Phi(x)$ is the standard Gaussian cumulative distribution function.
Finally, an attentive pooling layer is applied to a sequence of 248, reducing the temporal dimension of the audio signal, i.e.,

$$e_{audio} = \text{Atten-Pooling}(\bar{e}_{audio}).$$

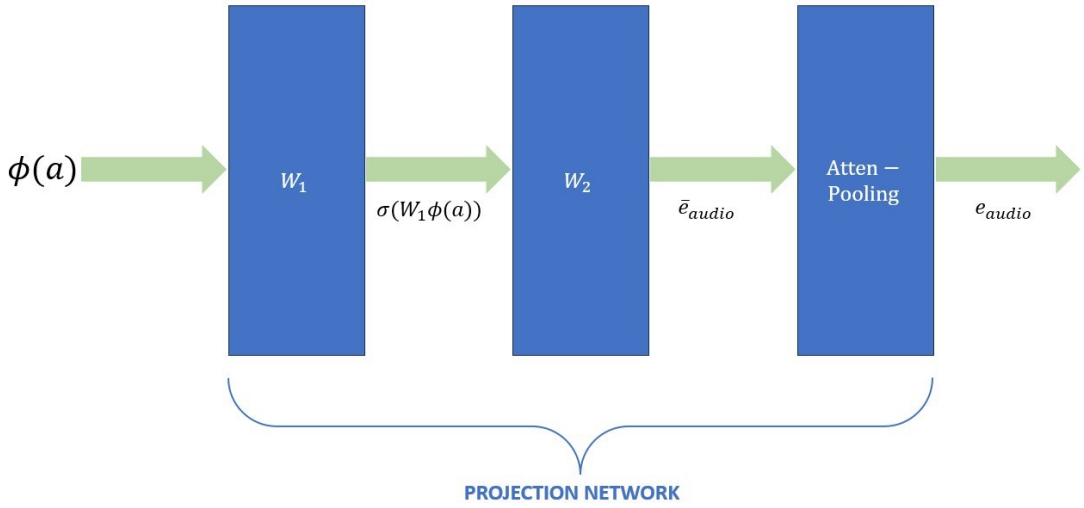

Figure 4.2: Schematic representation of the projection network inside the Embedder.

4.1.2 Optimization

During the optimization process (i.e., the training phase), the loss adopted is an additional one that complements $\mathcal{L}_{\text{LDM}}$, which involves encoding the label of the image, denoted by $l \in \mathbb{R}^{n_l \times d_a}$, where n_l represents the label's length. The label is encoded using the generative model's textual encoder, and then the spatial dimension is reduced using average pooling, i.e., $\hat{l} = \text{Avg-Pooling}(l)$. The classification loss is defined as follows:

$$\mathcal{L}_{\text{CL}} = \left(1 - \frac{\langle e_{audio}, \hat{l} \rangle}{\|e_{audio}\| \|\hat{l}\|}\right)^2.$$

Intuitively, this term ensures that the audio embedding remains close to the image's concept, facilitating faster and stabler convergence. Finally, an l_1 norm regularization term is added to the encoded audio token, in order to let it be more evenly distributed.
The overall loss that is optimized is given by

$$\mathcal{L} = \mathcal{L}_{\text{LDM}} + \lambda_{l_1} \|e_{audio}\|_1$$

where λ_{l_1} is a regularization term for the l_1 norm.
The overall loss that is optimized with classification loss is given by

$$\mathcal{L} = \mathcal{L}_{\text{LDM}} + \lambda_{l_1} \|e_{audio}\|_1 + \lambda_{\text{CL}} \mathcal{L}_{\text{CL}}$$

where λ_{CL} is a regularization term for the classification loss.

4.1.3 Evaluation functions

Evaluation functions in a context of artistic images generation from music inputs is an opened problem ([50]). Standard measures like FID (Fréchet Inception Distance) or AIC (Audio-Image Content) can be helpful to compare different performances and establish baselines. However, in an experimental context, they can reduce the possibility of exploring models' potential. For the method proposed here, the following objective evaluation functions are considered:

- **Audio-Image Similarity (AIS)**, that ideally measures the similarity between the semantic input audio and generated image features. As Yariv et al. ([80]) do for AudioToken, Wav2CLIP model ([74]) is employed. The Wav2CLIP model enables to measure the similarity between representations of an audio and image pair. This allows to quantify to which extent the generated image describes the audio. Leveraging the Wav2CLIP embedding representations for audio and images, the similarity between two embedded vectors, namely u and v, can be expressed as:

$$\cos(\theta) = \frac{\langle u, v \rangle}{\|u\| \|v\|}$$

  where θ is the angle between the two vectors. This measure is also known as *cosine similarity*. The AIS measure is, then, obtained as the mean percentage of the distances between similarities computed with generated images and their respective audio inputs, and mean similarities computed with the same generated images and all audios in the test set. Formally:

$$\text{AIS} = \left(\frac{1 - \cos^{-1}\left(\sum_{i=1}^{G} \left(\frac{\text{sim}(i_g, a_i) - \sum_{j \neq i, j=1}^{T} \left(\frac{\text{sim}(i_g, a_j)}{T} \right)}{G} \right) \right) \cdot \frac{2}{\pi} + 1}{2} \right) \cdot 100$$

  where G and T are, respectively, the number of generated images and the number of audio file in the test set, $\text{sim}(i_g, a_i)$ is the similarity computed between the generated image i_g and its corresponding music input a_i, and $\text{sim}(i_g, a_j)$ is the similarity computed between the generated image i_g and a generic music input inside the test set;

- **Image-Image Similarity (IIS)**, that measures the semantic similarity between the generated image and the ground truth one (i.e., the one associated with the music input adopted to generate the image). Again, the same reference-based method used to compute the AIS metric is employed. The only difference is that, in this case, music inputs and all audios in the test set are replaced, respectively, with the ground truth images and all images in the test set. Formally:

$$\text{IIS} = \left(\frac{1 - \cos^{-1}\left(\sum_{i=1}^{G} \left(\frac{\text{sim}(i_g, i) - \sum_{j \neq i, j=1}^{T} \left(\frac{\text{sim}(i_g, j)}{T} \right)}{G} \right) \right) \cdot \frac{2}{\pi} + 1}{2} \right) \cdot 100$$

  where T, now, is the number of images in the test set, i is the ground truth image relative to the generated image i_g, and j is a generic image inside the test set;

- **Fréchet Inception Distance (FID)**, that compares the distribution of the generated images against the original ones using an internal representation obtained from a pre-trained model (in this case, [47]). It is a standard score, introduced for the first time by Heusel et al. ([30]) that, essentially, measures the quality of the generated images. For any two probability distributions, μ and ν over $\mathbb{R}^n$, having finite mean and variance, their Fréchet distance is:

$$d_F(\mu, \nu) \coloneqq \left(\inf_{\gamma \in \Gamma(\mu, \nu)} \int_{\mathbb{R}^n \times \mathbb{R}^n} \|x - y\|^2 d\gamma(x, y) \right)^{\frac{1}{2}}$$

  where $\Gamma(\mu, \nu)$ is the set of all measures on $\mathbb{R}^n \times \mathbb{R}^n$ with marginal distributions μ and ν, respectively, on the first and second factors (in other words, the FID measure is the 2-Wasserstein distance on $\mathbb{R}^n$);

- **Generated Image-Image Label Similarity (GIILS)**, that relies on the same similarity measure used to compute AIS and IIS but, this time, applied to the generated images obtained from music inputs belonging to the same category. The internal mean, thus, is computed with respect to the generated images space. Formally:

$$\text{GIILS} = \left(\frac{1 - \cos^{-1}\left(\sum_{(i_{g_L}, j_{g_L})} \left(\frac{\text{sim}(i_{g_L}, j_{g_L}) - \sum_{j_g \neq i_{g_L}} \left(\frac{\text{sim}(i_{g_L}, j_g)}{G} \right)}{\#(i_{g_L}, j_{g_L})} \right) \right) \cdot \frac{2}{\pi} + 1}{2} \right) \cdot 100$$

  where (i_{g_L}, j_{g_L}) is a pair of generated images from music inputs having the same label L, and j_g is a generic generated image.

4.2 Adaptation of the model

In a context of artistic image generation from music input, leveraging the AudioToken architecture implies mainly working on the type of data the model needs to receive. In fact, in the case of the method described by Yariv et al. ([80]), a single datum is a 10 seconds video, taken from the VGG-Sound dataset (i.e, an audio-visual correspondent dataset consisting of short clips of audio sounds extracted from YouTube videos, as described in [45]), while, in the case of the image-music dataset, a single datum can be an image or an audio file.
Authors of [80] extract a random frame from the VGG-Sound video by selecting among those with the highest CLIP score ([60]) relative to the VGG-Sound label of the video. Then, the aforemontioned frame is coupled with the audio file of the video, in order to obtain the image-audio pair to be processed by the model. Instead, an image-audio pair from the image-music dataset is created as follows: for each audio file of the dataset, its corresponding image is sampled from those belonging to the same category of the audio file. Thus, the resulting image-audio pair is ready to be processed following the same pipeline as AudioToken.
Moreover, Yariv et al. ([80]) do not need to learn a new token for each individual class of audio or type scene, because the generative model is also conditioned by the trasformation of an initial prompt. In the case of the audio-image music dataset, the prompt is changed from “a photo of” to “an art image of”.
Altough the AudioToken code does not include a proper validation phase, the model adopted for the image-music dataset does have one.

Chapter 5

Experiments and results

Results of 2 out of a series of experiments are provided in the next section. The dataset adopted is the customized music-image dataset described in Chapter 3, which consists of 24216 samples for each type of file, grouped into 18 labeled categories. As far as the generative part is concerned, *ImageBind's unified latent* ([75]), together with *CompVis/stable-diffusion-v1-4* ([72]) are adopted, resulting in a 8853507 parameters model.

5.1 Experiments

5.1.1 Music-image conditional model 1

For this experiment only the prjoection network is optimized, i.e., the pre-trained audio encoder network and the pre-trained text-to-image generative network remain frozen. The model is trained on a Nvidia A100 for 25 epochs, with an initial learning rate of $8e-5$, a training batch size equal to 8, a validation batch size equal to 1 and a *cosine* learning rate schedule. This last condition means that the learning rate parameter of the model is forced to change during the training phase according to a periodic function, defined as:

$$\eta_t = \eta_{min} + \frac{1}{2}\left(\eta_{max} - \eta_{min}\right)\left(1 + \cos\left(\frac{t}{T}\pi\right)\right)$$

where t is a time step, η_{min} is the initial learning rate, η_{max} is the finale learning rate, and T is the total number of training steps.
Figure 5.1 shows the training and validation losses graph.

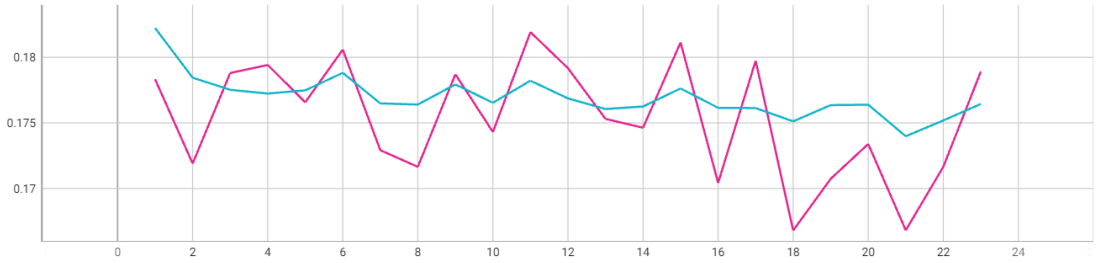

Figure 5.1: Train (blue) and validation (purple) losses performance for the first experiment.

The validation loss trend is irregular and does not decrease steadily. On the other hand, although faint, the training loss shows a decreasing profile.
Results of the objective evaluation functions are computed after generating 335 images from music inputs sampled inside the test set. The weights adopted to run this inference are those learned at epoch 21 because, as shown by Figure 5.1, in that circumstance losses are at their lowest values. Table 5.1 contains the AIS, IIS and FID values, obtained for the aforementioned generated images.

METHOD	METRIC		
	AIS ↑	IIS ↑	FID ↓
Music-image conditional model 1	50.15	49.62	179.16

Table 5.1: AIS, IIS and FID values obtained from 335 images generated during the first experiment, with respect to the distribution of the test samples (for AIS and IIS scores) and the distribution of the ground truth images (for the FID score).

Table 5.2 shows ordered results of the GIILS measure computations over 10 generated images, sampled among those obtained with a specific input category.

MUSIC INPUTS LABEL	**GIILS ↑**
Renaissance symbolism	58.70
Baroque Classical symbolism	57.91
Electronic music	56.12
Renaissance secular subject	54.77
20th century abstractionism	54.38
Romanticism open view	54.29
Heavy metal	53.90
African traditional folklore	53.75
20th century experimental open view	53.49
20th century traditional open view	53.41
20th century experimental figure	53.31
Asian traditional folklore	52.92
Romanticism figure	52.44
American traditional folklore	52.35
European traditional folklore	52.21
Modern instrumental music	51.87
Baroque Classical secular subject	51.56
20th century traditional figure	51.31

Table 5.2: Ordered mean percentage values of similarities computed between all images generated from 10 music inputs belonging to the same category, with respect to the distribution of the 335 generated images, during the first experiment.

As Table 5.2 shows, GIILS measures are significantly higher than AIS and IIS values for almost every label. This indicates that the model is able to recognize the input music context and adapt to it during the inference phase.
Next figures are selected in order to visualize the semantic relation among outputs generated from music inputs having the same label (i.e., the GIILS measure) and between these outputs and their input label. Some of these figures (as figures 5.6, 5.10, 5.11, and 5.13), can be considered definitely similar to the their label content.

Figure 5.2: 4 images generated from music inputs with label *20th century traditional figure*, during the first experiment.

Figure 5.3: 4 images generated from music inputs with label *20th century experimental figure*, during the first experiment.

Figure 5.4: 4 images generated from music inputs with label *20th century traditional open view*, during the first experiment.

Figure 5.5: 4 images generated from music inputs with label *20th century experimental open view*, during the first experiment.

Figure 5.6: 4 images generated from music inputs with label *Romanticism open view*, during the first experiment.

Figure 5.7: 4 images generated from music inputs with label *Romanticism figure*, during the first experiment.

Figure 5.8: 4 images generated from music inputs with label *Electronic music*, during the first experiment.

Figure 5.9: 4 images generated from music inputs with label *Heavy metal*, during the first experiment.

Figure 5.10: 4 images generated from music inputs with label *Renaissance secular subject*, during the first experiment.

Figure 5.11: 4 images generated from music inputs with label *Baroque Classical secular subject*, during the first experiment.

Figure 5.12: 4 images generated from music inputs with label *African traditional folklore*, during the first experiment.

Figure 5.13: 4 images generated from music inputs with label *Asian traditional folklore*, during the first experiment.

Figure 5.14: 4 images generated from music inputs with label *American traditional folklore*, during the first experiment.

5.1.2 Music-image conditional model 2

In the second experiment, the optimization phase involves the projection network, as well as the pre-trained audio encoder network and the pre-trained text-to-image generative network. The model is trained on a Nvidia A100 for 50 epochs, with an initial learning rate of $1e-5$, a training batch size equal to 8, a validation batch size equal to 8, and a *cosine* learning rate schedule.
Figure 5.15 shows the training and validation losses graph.

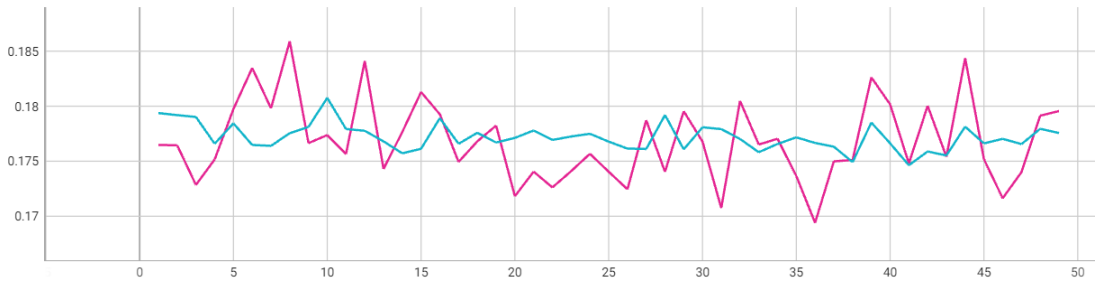

Figure 5.15: Train (blue) and validation (purple) losses performance for the second experiment.

Not taking into account the first 8 epochs, the validation loss graph decreases irregularly at least until epoch 36, where it reaches its minimum. On the other hand, the training loss, apart from the first 10 epochs, decreases irregularly until epoch 41.
Results of the objective evaluation functions are computed after generating 335 images from music inputs sampled inside the test set. The weights adopted to run this inference are those learned at epoch 36 because, as shown by Figure 5.15, in that circumstance the validation loss is at its lowest value and the training loss is almost at its lowest value. Table 5.3 contains the AIS, IIS and FID values for the aforementioned generated images.

METHOD	METRIC		
	AIS ↑	IIS ↑	FID ↓
Music-image conditional model 2	49.91	50.03	191.73

Table 5.3: AIS, IIS and FID values obtained from 335 images generated during the second experiment, with respect to the distribution of the test samples (for AIS and IIS scores) and the distribution of the ground truth images (for the FID score).

Table 5.4 shows ordered results of the GIILS measure computations over 10 generated images, sampled among those obtained with a specific input category.

MUSIC INPUTS LABEL	**GIILS ↑**
Renaissance symbolism	58.28
Baroque Classical symbolism	57.16
Renaissance secular subject	54.85
Electronic music	53.83
Heavy metal	53.80
Romanticism figure	53.72
20th century abstractionism	53.48
20th century experimental open view	53.32
20th century experimental figure	53.12
American traditional folklore	52.86
African traditional folklore	52.75
European traditional folklore	52.00
Romanticism open view	51.97
20th century traditional figure	51.76
Asian traditional folklore	51.44
Modern instrumental music	51.35
Baroque Classical secular subject	51.29
20th century traditional open view	50.96

Table 5.4: Ordered mean percentage values of similarities computed between all images generated from 10 music inputs belonging to the same category, with respect to the distribution of the 335 generated images, during the second experiment.

Also in this case, as Table 5.2 shows, GIILS measures are significantly higher than AIS and IIS values for almost every label. Again, the model is able to recognize the input music context and adapt to it during the inference phase. Anyway, by comparing the objective metrics results of the second experiment to those obtained in the first one, it is possible to notice that there are no significant changes in the values, except for the fact that the measurements in Table 5.4 are all lower than the corresponding ones (i.e., those with the same rank) in Table 5.2. Given the definition of the GIILS metric, this indicates a higher tendency to diversify outputs generated from the same context inputs by the model with the second hyperparameter configuration compared to the first one.
Next qualitative results, such as figures 5.17, 5.19, 5.22, 5.24, 5.28, 5.31, and 5.32, obtained from the second experiment, decisively validate what has just been stated.

Figure 5.16: 4 images generated from music inputs with label *20th century abstractionism*, during the second experiment.

Figure 5.17: 4 images generated from music inputs with label *20th century experimental figure*, during the second experiment.

Figure 5.18: 4 images generated from music inputs with label *20th century traditional open view*, during the second experiment.

Figure 5.19: 4 images generated from music inputs with label *20th century experimental open view*, during the second experiment.

Figure 5.20: 4 images generated from music inputs with label *Romanticism open view*, during the second experiment.

Figure 5.21: 4 images generated from music inputs with label *Romanticism figure*, during the second experiment.

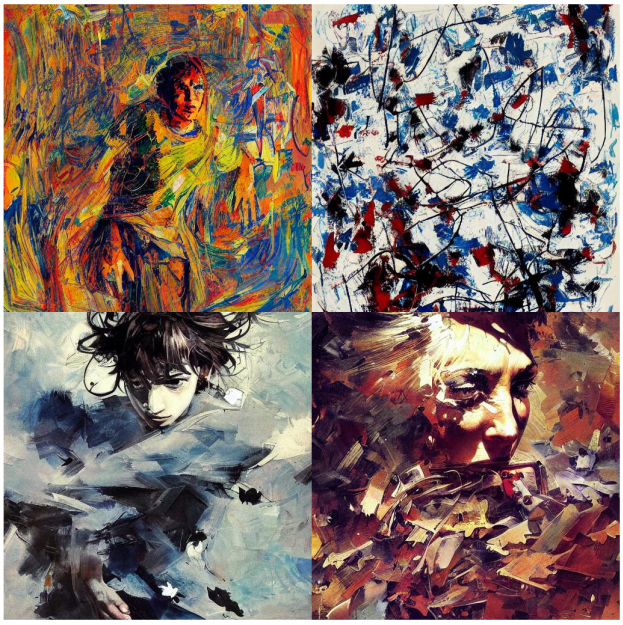

Figure 5.22: 4 images generated from music inputs with label *Electronic music*, during the second experiment.

Figure 5.23: 4 images generated from music inputs with label *Heavy metal*, during the second experiment.

Figure 5.24: 4 images generated from music inputs with label *Renaissance secular subject*, during the second experiment.

Figure 5.25: 4 images generated from music inputs with label *Baroque Classical secular subject*, during the second experiment.

Figure 5.26: 4 images generated from music inputs with label *African traditional folklore*, during the second experiment.

Figure 5.27: 4 images generated from music inputs with label *Asian traditional folklore*, during the second experiment.

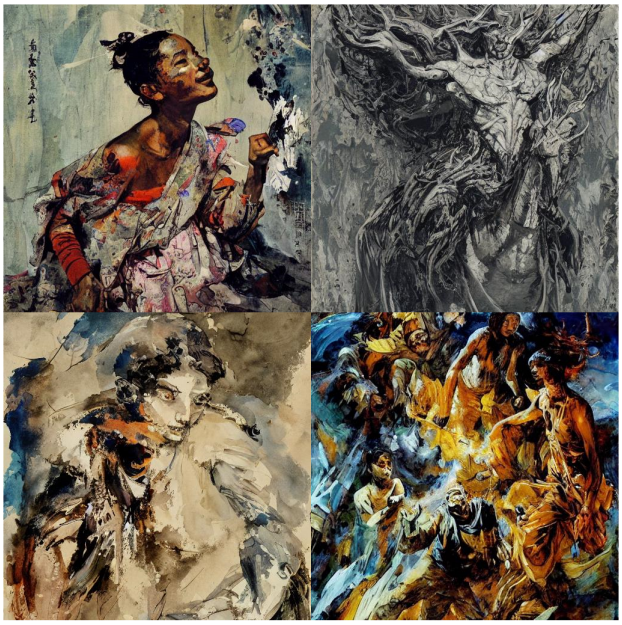

Figure 5.28: 4 images generated from music inputs with label *American traditional folklore*, during the second experiment.

Figure 5.29: 4 images generated from music inputs with label *20th century traditional figure*, during the second experiment.

Figure 5.30: 4 images generated from music inputs with label *Baroque Classical symbolism*, during the second experiment.

Figure 5.31: 4 images generated from music inputs with label *European traditional folklore*, during the second experiment.

Figure 5.32: 4 images generated from music inputs with label *Modern instrumental music*, during the second experiment.

Figure 5.33: 4 images generated from music inputs with label *Renaissance symbolism*, during the second experiment.

Chapter 6

Conclusions

This thesis introduces a method for leveraging text-conditioned generative models based on music conditioning. The developed method produces artistic images that are historically and semantically related to the music inputs used to generate them, thanks to a structured categorization of a customized dataset. The benefits of the proposed methodology are evaluated through a comprehensive framework that takes into account objective metrics, following a typical semantic knowledge extraction approach. The model proposed is only a first step toward music-conditioned images generation and the introduction of new evaluation frameworks. From this perspective, the research project described in this thesis shows many interesting aspects that warrant further investigations.
First of all, given the complexity of the model, it is crucial to define a proper validation setup, capable of efficiently handling hyperparameters changes during the model training phase. Moreover, even though dealing with pre-trained models provides clear advantages during the testing phase of the experiments, it can negatively affects attempts to reduce losses, especially when the trainable architectures are significantly smaller than the pre-trained ones. Unfortunately, traditional machine learning models rely on some well known validation techniques (e.g.,*cross-validation*, *bootstrap*, etc.) that, when dealing with large datasets (as in the case of generative modeling), can not be adopted due to computational constraints. That is why a more realistic approach to validate the training results in a generative modeling framework could start from the application of dimensionality reduction methodologies (e.g., *feature selection*, *random search*, *Bayesian optimization*, etc.).

AI tools investigating creativity can be very useful to our community for a pletora of reasons (research, entertainment, education, etc.). Nevertheless, due to the depth and diversity of the semantic content related to a piece of art, especially when dealing with multimodal generative learning, structuring a priori the connections between inputs and outputs that a model needs to understand is fundamental.

From this point of view, music-image association rules introduced in this work represent only one of many possible strategies, with respect to a specific purpose, that can be adopted to establish basic relations among data. Then, evaluation metrics (both subjective and objective) for the generated outputs could be selected according to the aforementioned connections. A very important work that sheds light on this issue is that of Theis et al. ([24]), concerning the significant independence of three of the most commonly used criteria (i.e, *average log-likelihood*, *Parzen window estimates*, and *visual fidelity of samples*) to evaluate and interpret results of generative models. Benny et al. ([50]), moreover, show that it is also possible to introduce new metrics for evaluating generative model performance in the class-conditional image generation setting. In this respect, the GILLS metric defined in Chapter 4 could be considered a valid starting point for introducing a creativity measure in a generative model framework like the one depicted in this thesis. Specifically, the similarity among generated images from inputs having the same label could serve as a penalization factor for the model's ability to diversify outputs for a specific task, with respect to predefined thresholds.
Finally, considering that open problems such as those involving AI and creativity require numerous interdisciplinary studies, it is essential that collaboration and sharing become the guiding principles for the entire community working in this field.

Appendix

Below are some images generated with the music-image multimodal generative model, using music inputs taken out of the dataset. A brief description of the input is provided for every image shown. Additionally, except in the case of the 2 experiments depicted in Chapter 5, the training hyperparameter configuration adopted to achieve that specific inference is also provided.

Preliminary model 1

The model is trained on a Nvidia GeForce RTX 3060 for 8 epochs, with an initial learning rate of $8e-5$, a training batch size equal to 1, gradient accumulation step of 4, a validation batch size equal to 1, and a *constant* learning rate schedule. Weights used for the generation are taken at epoch 8.

Figure 1: An electronic music sample. The music is essentially experimental, obtained with a mixture of pre-sampled traditional and concrete sounds.

Figure 2: Sound of horses passing through a wood.

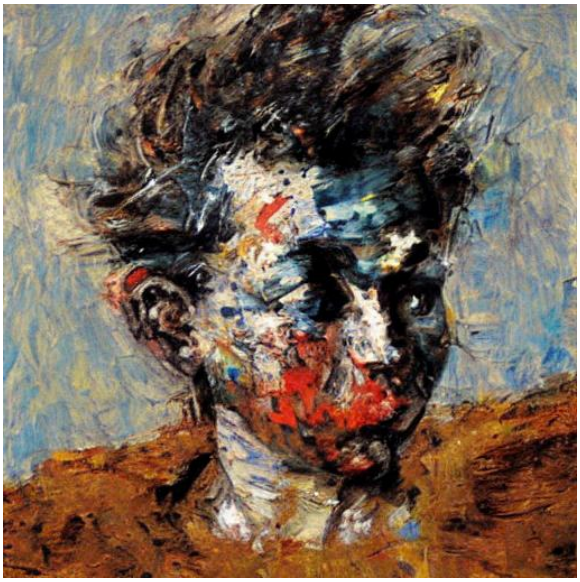

Figure 3: A melancholic piano improvisation recording.

Figure 4: A melodic fragment of a suite for piano and alto saxophone, named *Suite hellenique* and composed by Pedro Iturrlade (Falces, July 13 1929 – Madrid, November 1 2020).

Preliminary model 2

The model is trained on a Nvidia GeForce RTX 3060 for 13 epochs, with an initial learning rate of $8e-5$, a training batch size equal to 1, gradient accumulation step of 4, a validation batch size equal to 1, and a *cosine* learning rate schedule. An l_2 norm regularization term equal to 0.01 for the encoded audio is added to the loss function, that becomes:

$$\mathcal{L} = \mathcal{L}_{\text{LDM}} + \lambda_{l_1} \|e_{audio}\|_1 + \lambda_{l_2} \|e_{audio}\|_2 + \lambda_{\text{CL}} \mathcal{L}_{\text{CL}}.$$

Weights used for the generation are taken at epoch 13.

Figure 5: An electronic music sample. Sounds are not traditional, but they are obtained with a digital synthesizer.

Figure 6: Another electronic music sample. The music is a combination of concrete (recorded voices) and digital sounds.

Figure 7: A fragment of a digital symphony.

Figure 8: A digital noise/industrial music sample.

Music-image conditional model 1

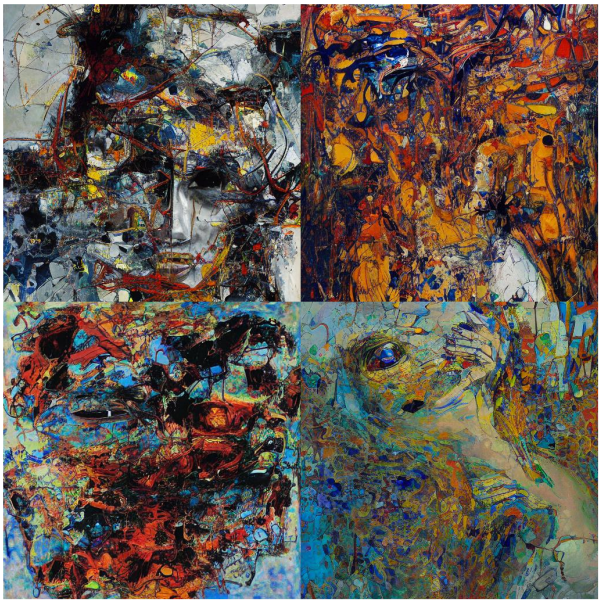

Figure 9: 4 images generated from a 90's dance music sample.

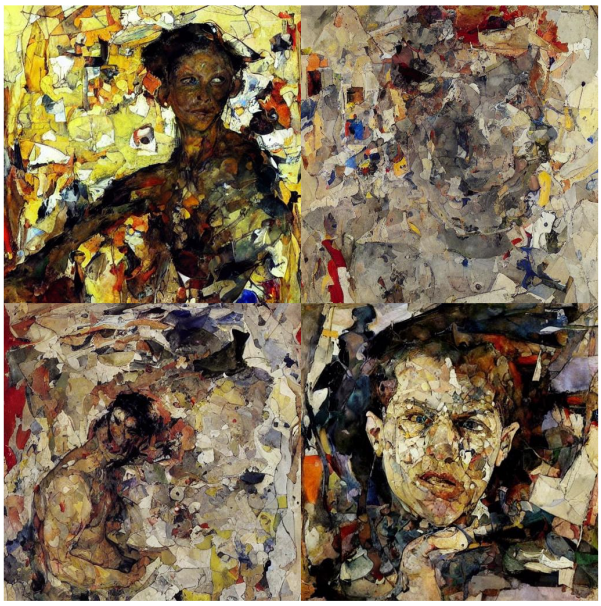

Figure 10: 4 images generated from an experimental music fragment, realized with an electric piano, a classical guitar and a bass guitar.

Figure 11: 4 images generated from a fragment of a nostalgic piano improvisation in jazz style.

Figure 12: 4 images generated from a two violins canon sample. The music style is mainly classical, and the imitation is realized in unison.

Figure 13: 4 images generated from a melodic fragment of a tonal composition for flute and harp.

Music-image conditional model 2

Figure 14: 4 images generated from an experimental electronic piece of music titled *The Witchfinder*, composed by a duo called Amorphous Androgynous.

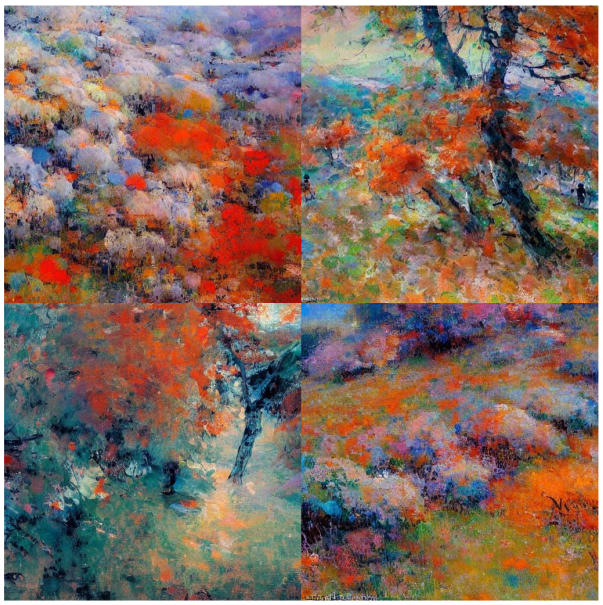

Figure 15: 4 images generated from a recording of birds chirping in the early morning.

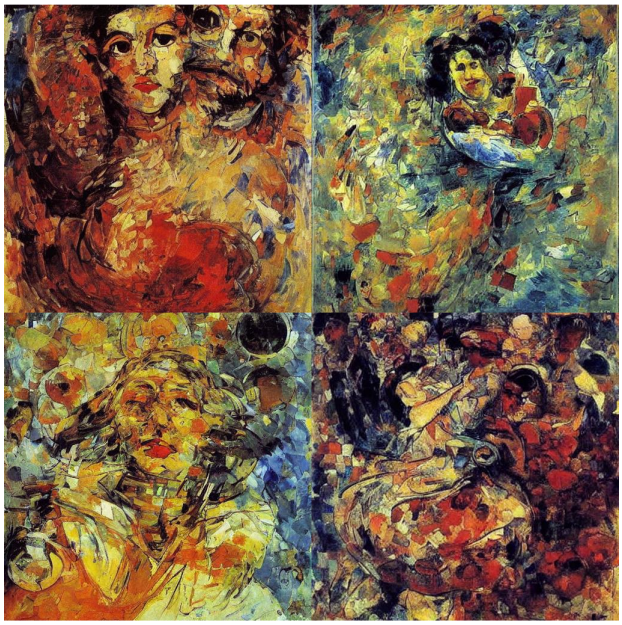

Figure 16: 4 images generated from a fragment of a dynamic piano improvisation in G-flat major.

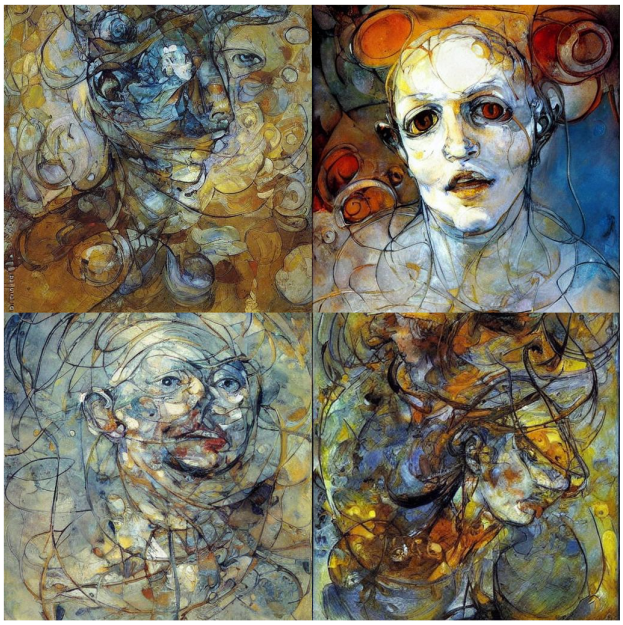

Figure 17: 4 images generated from a fragment of a piano prelude, influenced primarily by the early piano pieces from *Mikrokosmos*, a work composed by Béla Bartók (Nagyszentmiklós, March 25, 1881 – New York, September 26, 1945).

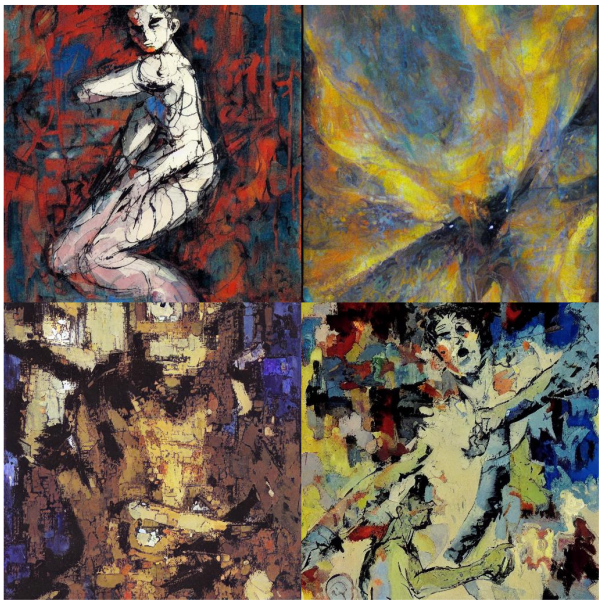

Figure 18: 4 images generated from an elaborately manipulated (i.e., distorting sounds) improvisation for electric piano and alto saxophone.

Figure 19: 4 images generated from a fragment of *The Caterpillar Song*, featured in the famous Disney movie *Alice in Wonderland*.

Bibliography

[1] William Feller. «On the Theory of Stochastic Processes, with Particular Reference to Applications». In: 1949. URL: https://api.semanticscholar.org/CorpusID:121027442 (cit. on p. 41).

[2] Morris W. Hirsch. *Differential Topology*. Springer-Verlag, 1994 (cit. on p. 13).

[3] Luc Devroye. «Random variate generation in one line of code». In: *Proceedings Winter Simulation Conference* (1996), pp. 265–272. URL: https://api.semanticscholar.org/CorpusID:197651 (cit. on p. 15).

[4] Sheldon M. Ross. *Stochastic Processes*. John Wiley & Sons, 1996 (cit. on pp. 18, 41).

[5] Michael Jordan, Zoubin Ghahramani, Tommi Jaakkola, and Lawrence Saul. «An Introduction to Variational Methods for Graphical Models». In: *Machine Learning* 37 (Jan. 1999), pp. 183–233. DOI: 10.1023/A:1007665907178 (cit. on p. 13).

[6] Aapo Hyvärinen. «Estimation of Non-Normalized Statistical Models by Score Matching». In: *J. Mach. Learn. Res.* 6 (2005), pp. 695–709. URL: https://api.semanticscholar.org/CorpusID:1152227 (cit. on p. 39).

[7] Xuelong Li, Dacheng Tao, Stephen Maybank, and Yuan Yuan. «Visual music and musical vision». In: *Neurocomputing* 71 (June 2008), pp. 2023–2028. DOI: 10.1016/j.neucom.2008.01.025 (cit. on p. 1).

[8] Sheldon M. Ross. *Probability and Statistics for Engineers and Scientist*. Elsevier, 2009 (cit. on pp. 13, 14).

[9] Pasi Saari, Tuomas Eerola, and Olivier Lartillot. «Generalizability and Simplicity as Criteria in Feature Selection: Application to Mood Classification in Music». In: *IEEE Transactions on Audio, Speech, and Language Processing* 19.6 (2011), pp. 1802–1812. DOI: 10.1109/TASL.2010.2101596 (cit. on pp. 3, 42).

[10] Xixuan Wu, Yu Qiao, Xiaogang Wang, and Xiaoou Tang. «Cross matching of music and image». In: Oct. 2012, pp. 837–840. DOI: 10.1145/2393347.2396325 (cit. on pp. 1, 42).

[11] Xixuan Wu, Bing Xu, Yu Qiao, and Xiaoou Tang. «Automatic music video generation: cross matching of music and image». In: *Proceedings of the 20th ACM International Conference on Multimedia*. MM '12. Nara, Japan: Association for Computing Machinery, 2012, pp. 1381–1382. ISBN: 9781450310895. DOI: 10.1145/2393347.2396495. URL: https://doi.org/10.1145/2393347.2396495 (cit. on pp. 1, 42).

[12] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Y. Bengio. «Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling». In: (Dec. 2014) (cit. on p. 7).

[13] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. *Generative Adversarial Networks*. 2014. arXiv: 1406.2661 [stat.ML] (cit. on p. 16).

[14] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. *A Convolutional Neural Network for Modelling Sentences*. 2014. arXiv: 1404.2188 [cs.CL] (cit. on p. 8).

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV] (cit. on p. 26).

[16] Jonathan Long, Evan Shelhamer, and Trevor Darrell. *Fully Convolutional Networks for Semantic Segmentation*. 2015. arXiv: 1411.4038 [cs.CV] (cit. on pp. 23, 24).

[17] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. «U-Net: Convolutional Networks for Biomedical Image Segmentation». In: *CoRR* abs/1505.04597 (2015). arXiv: 1505.04597. URL: http://arxiv.org/abs/1505.04597 (cit. on pp. 3, 23).

[18] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. *Deep Unsupervised Learning using Nonequilibrium Thermodynamics*. 2015. arXiv: 1503.03585 [cs.LG] (cit. on pp. 2, 17, 36).

[19] Baixi Xing, Kejun Zhang, Shouqian Sun, Lk Zhang, Zenggui Gao, Jiaxi Wang, and Shi Chen. «Emotion-driven Chinese folk music-image retrieval based on DE-SVM». In: *Neurocomputing* 148 (Jan. 2015), pp. 619–627. DOI: 10.1016/j.neucom.2014.08.007 (cit. on pp. 1, 42).

[20] Quanzeng You, Jiebo Luo, Hailin Jin, and Jianchao Yang. «Robust Image Sentiment Analysis Using Progressively Trained and Domain Transferred Deep Networks». In: *Proceedings of the AAAI Conference on Artificial Intelligence* 29 (Sept. 2015). DOI: 10.1609/aaai.v29i1.9179 (cit. on p. 1).

[21] Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Jozefowicz, and Samy Bengio. *Generating Sentences from a Continuous Space*. 2016. arXiv: 1511.06349 [cs.LG] (cit. on p. 3).

[22] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. «Density estimation using Real NVP». In: *CoRR* abs/1605.08803 (2016). arXiv: 1605.08803. URL: http://arxiv.org/abs/1605.08803 (cit. on p. 11).

[23] Ishan Misra, C. Lawrence Zitnick, and Martial Hebert. *Shuffle and Learn: Unsupervised Learning using Temporal Order Verification*. 2016. arXiv: 1603.08561 [cs.CV] (cit. on p. 32).

[24] Lucas Theis, Aäron van den Oord, and Matthias Bethge. *A note on the evaluation of generative models*. 2016. arXiv: 1511.01844 [stat.ML] (cit. on pp. 12, 82).

[25] Xixuan Wu, Yu Qiao, Xiaogang Wang, and Xiaoou Tang. «Bridging Music and Image via Cross-Modal Ranking Analysis». In: *IEEE Transactions on Multimedia* 18 (July 2016), pp. 1–1. DOI: 10.1109/TMM.2016.2557722 (cit. on pp. 1, 42).

[26] Quanzeng You, Jiebo Luo, Hailin Jin, and Jianchao Yang. «Building a Large Scale Dataset for Image Emotion Recognition: The Fine Print and The Benchmark». In: *Proceedings of the AAAI Conference on Artificial Intelligence* 30 (May 2016). DOI: 10.1609/aaai.v30i1.9987 (cit. on p. 1).

[27] Anton Becker, Magda Marcon, Soleen Stocker-Ghafoor, Moritz Wurnig, Thomas Frauenfelder, and Andreas Boss. «Deep Learning in Mammography: Diagnostic Accuracy of a Multipurpose Image Analysis Software in the Detection of Breast Cancer». In: *Investigative Radiology* 52 (Feb. 2017), p. 1. DOI: 10.1097/RLI.0000000000000358 (cit. on p. 3).

[28] Lele Chen, Sudhanshu Srivastava, Zhiyao Duan, and Chenliang Xu. *Deep Cross-Modal Audio-Visual Generation*. 2017. arXiv: 1704.08292 [cs.CV] (cit. on pp. 1, 43).

[29] Wangli Hao, Zhaoxiang Zhang, and He Guan. *CMCGAN: A Uniform Framework for Cross-Modal Visual-Audio Mutual Generation*. 2017. arXiv: 1711.08102 [cs.CV] (cit. on pp. 1, 43).

[30] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, Günter Klambauer, and Sepp Hochreiter. «GANs Trained by a Two Time-Scale Update Rule Converge to a Nash Equilibrium». In: *CoRR* abs/1706.08500 (2017). arXiv: 1706.08500. URL: http://arxiv.org/abs/1706.08500 (cit. on p. 58).

[31] Pierre Schaeffer. *Treatise on Musical Objects, An Essay across Disciplines*. Oakland, California: Univerity of California Press, 2017 (cit. on p. ii).

[32] David Ha and Jürgen Schmidhuber. «World Models». In: (2018). DOI: 10.5281/ZENODO.1207631. URL: https://zenodo.org/record/1207631 (cit. on p. 3).

[33] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. *Densely Connected Convolutional Networks*. 2018. arXiv: 1608.06993 [cs.CV] (cit. on p. 27).

[34] Martin Simonovsky and Nikos Komodakis. *GraphVAE: Towards Generation of Small Graphs Using Variational Autoencoders*. 2018. arXiv: 1802.03480 [cs.LG] (cit. on p. 3).

[35] *WikiArt Dataset (Refined)*. A Refined WikiArt dataset. 2018. URL: https://www.kaggle.com/datasets/trungit/wikiart25k?rvi=1 (cit. on pp. 44, 45).

[36] *Best Artworks of All Time*. Collection of Paintings of the 50 Most Influential Artists of All Time, from artchallenge.ru. 2019. URL: https://www.kaggle.com/datasets/ikarus777/best-artworks-of-all-time (cit. on pp. 44–46).

[37] Aurora Linh Cramer, Ho-Hsiang Wu, Justin Salamon, and Juan Pablo Bello. «Look, Listen, and Learn More: Design Choices for Deep Audio Embeddings». In: *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2019, pp. 3852–3856. DOI: 10.1109/ICASSP.2019.8682475.

[38] Samarth Sinha, Sayna Ebrahimi, and Trevor Darrell. *Variational Adversarial Active Learning*. 2019. arXiv: 1904.00370 [cs.LG] (cit. on p. 3).

[39] Yang Song and Stefano Ermon. «Generative Modeling by Estimating Gradients of the Data Distribution». In: *CoRR* abs/1907.05600 (2019). arXiv: 1907.05600. URL: http://arxiv.org/abs/1907.05600 (cit. on pp. 17, 40).

[40] Gaurav Verma, Eeshan Gunesh Dhekane, and Tanaya Guha. «Learning Affective Correspondence between Music and Image». In: *CoRR* abs/1904.00150 (2019). arXiv: 1904.00150. URL: http://arxiv.org/abs/1904.00150 (cit. on pp. 1, 42).

[41] Baixi Xing, Kejun Zhang, Lekai Zhang, Xinda Wu, Jian Dou, and Shouqian Sun. «Image–Music Synesthesia-Aware Learning Based on Emotional Similarity Recognition». In: *IEEE Access* 7 (2019), pp. 136378–136390. DOI: 10.1109/ACCESS.2019.2942073 (cit. on pp. 1, 42).

[42] Nikolas Adaloglou. «In-layer normalization techniques for training very deep neural networks». In: *https://theaisummer.com/* (2020). URL: https://theaisummer.com/normalization/ (cit. on pp. 24, 29, 30).

[43] Nikolas Adaloglou. «Intuitive Explanation of Skip Connections in Deep Learning». In: (2020). URL: https://theaisummer.com/skip-connections/ (cit. on pp. 24, 27).

[44] Nikolas Adaloglou and Sergios Karagiannakos. «How attention works in deep learning: understanding the attention mechanism in sequence models». In: *https://theaisummer.com/* (2020). URL: https://theaisummer.com/attention/ (cit. on pp. 30, 31, 33, 34).

[45] Honglie Chen, Weidi Xie, Andrea Vedaldi, and Andrew Zisserman. *VGGSound: A Large-scale Audio-Visual Dataset*. 2020. arXiv: 2004.14368 [cs.CV] (cit. on pp. 2, 48, 59).

[46] Jonathan Ho, Ajay Jain, and Pieter Abbeel. *Denoising Diffusion Probabilistic Models*. 2020. arXiv: 2006.11239 [cs.LG] (cit. on pp. 2, 17, 19–23).

[47] Maximilian Seitzer. *pytorch-fid: FID Score for PyTorch*. https://github.com/mseitzer/pytorch-fid. Version 0.3.0. Aug. 2020 (cit. on p. 58).

[48] Yang Song and Stefano Ermon. «Improved Techniques for Training Score-Based Generative Models». In: *CoRR* abs/2006.09011 (2020). arXiv: 2006.09011. URL: https://arxiv.org/abs/2006.09011.

[49] Nikolas Adaloglou. «How Transformers work in deep learning and NLP: an intuitive introduction». In: *https://theaisummer.com/* (2021). URL: https://github.com/The-AI-Summer/self-attention-cv (cit. on p. 34).

[50] Yaniv Benny, Tomer Galanti, Sagie Benaim, and Lior Wolf. «Evaluation Metrics for Conditional Image Generation». In: *International Journal of Computer Vision* 129.5 (Mar. 2021), pp. 1712–1731. ISSN: 1573-1405. DOI: 10.1007/s11263-020-01424-w. URL: http://dx.doi.org/10.1007/s11263-020-01424-w (cit. on pp. 3, 57, 82).

[51] Prafulla Dhariwal and Alex Nichol. «Diffusion Models Beat GANs on Image Synthesis». In: *CoRR* abs/2105.05233 (2021). arXiv: 2105.05233. URL: https://arxiv.org/abs/2105.05233 (cit. on p. 37).

[52] Shuyang Gu, Dong Chen, Jianmin Bao, Fang Wen, Bo Zhang, Dongdong Chen, Lu Yuan, and Baining Guo. «Vector Quantized Diffusion Model for Text-to-Image Synthesis». In: *arXiv preprint arXiv:2111.14822* (2021) (cit. on p. 3).

[53] Andrey Guzhov, Federico Raue, Jörn Hees, and Andreas Dengel. *AudioCLIP: Extending CLIP to Image, Text and Audio*. 2021. arXiv: 2106.13043 [cs.SD] (cit. on p. 43).

[54] Andrey Guzhov, Federico Raue, Jörn Hees, and Andreas Dengel. *ESResNe(X)t-fbsp: Learning Robust Time-Frequency Transformation of Audio*. 2021. arXiv: 2104.11587 [cs.SD] (cit. on p. 43).

[55] Jonathan Ho, Chitwan Saharia, William Chan, David J. Fleet, Mohammad Norouzi, and Tim Salimans. «Cascaded Diffusion Models for High Fidelity Image Generation». In: *CoRR* abs/2106.15282 (2021). arXiv: 2106.15282. URL: https://arxiv.org/abs/2106.15282 (cit. on pp. 2, 38).

[56] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning, with Applications in R*. 2021 (cit. on pp. 5, 13, 16).

[57] Seung Hyun Lee, Wonseok Roh, Wonmin Byeon, Sang Ho Yoon, Chan Young Kim, Jinkyu Kim, and Sangpil Kim. *Sound-Guided Semantic Image Manipulation*. 2021. arXiv: 2112.00007 [cs.GR] (cit. on pp. 1, 42).

[58] Alex Nichol and Prafulla Dhariwal. *Improved Denoising Diffusion Probabilistic Models*. 2021. arXiv: 2102.09672 [cs.LG] (cit. on pp. 2, 20, 22).

[59] Alex Nichol, Prafulla Dhariwal, Aditya Ramesh andPranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. «GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models». In: *CoRR* abs/2112.10741 (2021). arXiv: 2112.10741. URL: https://arxiv.org/abs/2112.10741 (cit. on pp. 3, 36).

[60] Alec Radford et al. *Learning Transferable Visual Models From Natural Language Supervision*. 2021. arXiv: 2103.00020 [cs.CV] (cit. on pp. 43, 59).

[61] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. *Score-Based Generative Modeling through Stochastic Differential Equations*. 2021. arXiv: 2011.13456 [cs.LG] (cit. on pp. 40–42).

[62] Daniel Voigt Godoy. *Deep Learning with PyTorh Step-by-Step, A Beginner's Guide*. Voigt Godoy D., 2021 (cit. on p. 7).

[63] Lilian Weng. «What are diffusion models?» In: *lilianweng.github.io* (July 2021). URL: https://lilianweng.github.io/posts/2021-07-11-diffusion-models/ (cit. on p. 2).

[64] Sanyuan Chen, Yu Wu, Chengyi Wang, Shujie Liu, Daniel Tompkins, Zhuo Chen, and Furu Wei. *BEATs: Audio Pre-Training with Acoustic Tokenizers*. 2022. arXiv: 2212.09058 [eess.AS] (cit. on pp. 2, 3, 53, 55).

[65] Rinon Gal, Yuval Alaluf, Yuval Atzmon, Or Patashnik, Amit H. Bermano, Gal Chechik, and Daniel Cohen-Or. *An Image is Worth One Word: Personalizing Text-to-Image Generation using Textual Inversion*. 2022. arXiv: 2208.01618 [cs.CV] (cit. on p. 43).

[66] Songwei Ge, Thomas Hayes, Harry Yang, Xi Yin, Guan Pang, David Jacobs, Jia-Bin Huang, and Devi Parikh. *Long Video Generation with Time-Agnostic VQGAN and Time-Sensitive Transformer*. 2022. arXiv: 2204.03638 [cs.CV] (cit. on p. 43).

[67] Jonathan Ho and Tim Salimans. *Classifier-Free Diffusion Guidance*. 2022. arXiv: 2207.12598 [cs.LG] (cit. on pp. 2, 37).

[68] Sergios Karagiannakos. «Vision Language models: towards multi-modal deep learning». In: (2022). URL: https://theaisummer.com/vision-language-models/.

[69] Sergios Karagiannakos and Nikolaos Adaloglou. «Diffusion models: toward state-of-the-art image generation». In: (2022). URL: https://theaisummer.com/diffusion-models/ (cit. on pp. 2, 17).

[70] Calvin Luo. *Understanding Diffusion Models: A Unified Perspective*. 2022. arXiv: 2208.11970 [cs.LG] (cit. on pp. 2, 19, 21, 39).

[71] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. *Hierarchical Text-Conditional Image Generation with CLIP Latents*. 2022. arXiv: 2204.06125 [cs.CV] (cit. on p. 3).

[72] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. *High-Resolution Image Synthesis with Latent Diffusion Models*. 2022. arXiv: 2112.10752 [cs.CV] (cit. on pp. 2, 38, 39, 53, 60).

[73] Chitwan Saharia et al. *Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding*. 2022. arXiv: 2205.11487 [cs.CV] (cit. on p. 2).

[74] Ho-Hsiang Wu, Prem Seetharaman, Kundan Kumar, and Juan Pablo Bello. *Wav2CLIP: Learning Robust Audio Representations From CLIP*. 2022. arXiv: 2110.11499 [cs.SD] (cit. on pp. 3, 43, 57).

[75] Rohit Girdhar, Alaaeldin El-Nouby, Zhuang Liu, Mannat Singh, Kalyan Vasudev Alwala, Armand Joulin, and Ishan Misra. *ImageBind: One Embedding Space To Bind Them All*. 2023. arXiv: 2305.05665 [cs.CV] (cit. on p. 60).

[76] Soumik Mukhopadhyay, Saksham Suri, Ravi Teja Gadde, and Abhinav Shrivastava. *Diff2Lip: Audio Conditioned Diffusion Models for Lip-Synchronization*. 2023. arXiv: 2308.09716 [cs.CV].

[77] Ludan Ruan, Yiyang Ma, Huan Yang, Huiguo He, Bei Liu, Jianlong Fu, Nicholas Jing Yuan, Qin Jin, and Baining Guo. *MM-Diffusion: Learning Multi-Modal Diffusion Models for Joint Audio and Video Generation*. 2023. arXiv: 2212.09478 [cs.CV] (cit. on pp. 1, 43).

[78] A. J. Solberg. «Musical Source Separation using Diffusion Models». Master's thesis. Norwegian University of Science and Technolgy, 2023.

[79] Kim Sung-Bin, Arda Senocak, Hyunwoo Ha, Andrew Owens, and Tae-Hyun Oh. *Sound to Visual Scene Generation by Audio-to-Visual Latent Alignment*. 2023. arXiv: 2303.17490 [cs.CV] (cit. on p. 1).

[80] Guy Yariv, Itai Gat, Lior Wolf, Yossi Adi, and Idan Schwartz. *AudioToken: Adaptation of Text-Conditioned Diffusion Models for Audio-to-Image Generation*. 2023. arXiv: 2305.13050 [cs.SD] (cit. on pp. 2, 43, 53, 57, 59).

[81] Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. *Dive into Deep Learning*. 2023. arXiv: 2106.11342 [cs.LG] (cit. on pp. 3, 7, 17, 23–25).

[82] Ye Zhu, Yu Wu, Kyle Olszewski, Jian Ren, Sergey Tulyakov, and Yan Yan. *Discrete Contrastive Diffusion for Cross-Modal Music and Image Generation*. 2023. arXiv: 2206.07771 [cs.CV] (cit. on pp. 1, 43).

[83] Aritra Roy Gosthipaty and Ritwik Raha. «Getting Started with Diffusers for Text-to-Image». In: *PyImageSearch*. Ed. by Puneet Chugh, Susan Huot, and Kseniia Kidriavsteva. 2024. URL: https://pyimg.co/4ukb0.

[84] *Classical Asian Art*. Public domain images taken from Artvee and the Smithsonian Museum. URL: https://www.kaggle.com/datasets/ajrhee/classical-asian-art/data (cit. on p. 44).

[85] *Pinterest*. A website to find ideas. URL: https://www.pinterest.it/ (cit. on pp. 2, 44, 45).

[86] Stuart Russell and Peter Norvig. *Intelligenza artificiale, un approccio moderno, volume 1, quarta edizione*. Milano - Torino: Pearson Italia (cit. on p. 4).

[87] Stuart Russell and Peter Norvig. *Intelligenza artificiale, un approccio moderno, volume 2, quarta edizione*. Milano - Torino: Pearson Italia (cit. on p. 4).

[88] Jakub M. Tomczak. *Jakub M. Tomczak*. URL: https://jmtomczak.github.io/ (cit. on pp. 3–5, 7, 8, 10–12, 15, 17).

[89] *WikiArt*. The online home for visual arts from all around the world. URL: https://www.wikiart.org/ (cit. on pp. 2, 44, 46).

[90] *YouTube*. A web platform to share and visualize multimedia content. URL: https://www.youtube.com/ (cit. on pp. 2, 44, 46).