POLITECNICO DI TORINO

Master of Science in ICT for Smart Societies

Master Thesis

Generative Adversarial Networks for Emotion-based Music Generation

Supervisors
prof. Cristina Rottondi
tutor Mirko Zaffaroni

Candidato
Emanuele Aiello

October 2021

Compiled with LaTeX on October 10, 2021.

Abstract

Emotions are an important aspect of music, and the ability to regulate this characteristic might find a variety of applications in generating soundtracks or melodies appropriate to different types of domains. Various algorithms have been created to generate music with a specific emotion, however the exploitation of GAN in this field is still at an early stage. To investigate the applicability of GAN to the problem, the already existing GAN for symbolic music generation (*MuseGAN*) has been identified, and starting from it a modified architecture has been created. However, in this network, there is no control over the characteristics of the generated pieces, in particular since emotion is a very complex feature, achieving such control is very challenging. Three experiments are performed to control the generation obtaining melodies that elicit the target feeling in the listeners. The first one is based on a transfer learning approach, the second investigate the latent space exploration and the third exploits a mood classifier with an unconditioned GAN model.

Acknowledgements

I would first like to express my deepest appreciation to my thesis industrial supervisor Dr. Mirko Zaffaroni and my academic supervisor Dr. Cristina Rottondi. The completion of my work would not have been possible without their support and nurturing, in fact, they have been present since the beginning of the job, offering insightful suggestions.

Then I would like to thank all of my colleagues for offering comments on my work, discussing their views on this research with me, and providing constructive advice.

Finally, I am grateful to my family and friends who have supported me during these years and have always had a profound belief in my abilities. Their presence has been fundamental in allowing me to complete my academic path. A special thanks go to my girlfriend Giorgia for her patience and encouragement.

Emanuele Aiello.

Table of Contents

List of Tables

List of Figures

Acronyms

MIDI
Musical Instrument Digital Interface

MER
Music Emotion Recognition

MIR
Music Information Retrieval

LSTM
Long Short-Term Memory

CNN
Convolutional Neural Network

GAN
Generative Adversarial Network

DCGAN
Deep Convolutional Generative Adversarial Network

WGAN
Wasserstein Generative Adversarial Network

GP
Gradient Penalty

EMD
Earth Mover's Distance

FID

Fréchet Inception Distance

TTRU

Two Time scale Update Rule

LR

Learning Rate

SGA

Stochastic Gradient Ascent

UMAP

Uniform Manifold Approximation and Projection

Chapter 1

Introduction

Nowadays, generative adversarial networks (GANs) have proven themselves to be capable of creating hyper-realistic faces, animating paintings, colorizing sketches, and so on. However, these models can handle not only images but also text and audio.

The context of this thesis is the research and exploration of Generative Adversarial Networks applied to music generation. The objective outcome is to create melodies that elicit a specific emotion in the listener. This is known as affective music composition. Emotions are an important aspect of music, and the ability to regulate this characteristic might find a variety of applications in generating soundtracks or melodies appropriate to different types of domains.

Various algorithms have been created to generate music with a specific emotion. One of these systems uses genetic algorithms and advanced evolutionary approaches to produce real-time music capable of expressing various emotional states. Others make use of Long Short-Term Memory (LSTM) networks or ad hoc architecture based on Recurrent Convolutional Neural Networks (R-CNN). However, to the best of my knowledge, there are no published studies in the literature that use GANs in the field of emotional music creation.

To accomplish this goal, the state of the art in symbolic music generation for multitrack polyphonic melodies, which is called *MuseGAN*, has been identified, and starting from it a modified architecture has been created. However, in this network, there is no control over the characteristics of the generated pieces, in particular since emotion is a very complex feature, achieving such control is very challenging.

Symbolic music generation consists in generating a representation of notes and sounds to be played by an instrument or synthesized by a computer. The most prevalent symbolic format is MIDI; nevertheless, there is only one dataset that contains MIDI files tagged with emotions, it is called *VG MIDI* and contains piano pieces made by video games soundtracks, 200 of which labelled with an associated emotion and 3000 unlabelled. This dataset was utilized to carry out

the experiments in the thesis project. During the preprocessing, the MIDI files were cleaned, divided into pieces based on the relative emotion of each chunk, and transformed to pianoroll.

The major difficulty in this project lies in the availability of data because GANs require a large number of samples to be successfully trained. Three different approaches have been tested to achieve the desired goal.

The first one is based on transfer learning: a GAN model is trained in an unsupervised manner, after this training, the architecture is changed to that of a Conditional GAN and the process of training continues with the labeled dataset.

The second experiment consists in exploring the latent space to control the generation. An emotion classifier, based on convolutional layers, is trained with the labeled dataset. The classifier is used with "stochastic gradient ascent" to obtain the latent direction that maximizes the amount of a wanted feature, in this case, the requested emotion.

Finally, the last experiment uses the unconditioned GAN model to generate random melodies and the classifier to detect the target mood, in this approach melodies are generated until the target mood is found.

The models have been evaluated by calculating the statistical difference over several musical metrics between the generated and real samples. Furthermore, because emotions are very subjective, a survey has been constructed to qualitatively evaluate the generation process.

Chapter 2

Theoretical Background

This chapter reports the theoretical background upon which the entire project is built. The listed topics are necessary to understand completely the remainder of the work.

2.1 Music Generation

Music generation with neural networks is a very interesting and challenging task, indeed the results obtained in this domain are poor compared to those obtained in the image domain, for example. Music is an art of time, with higher-level building blocks containing smaller recurrent patterns [1], the hierarchical structure of a music piece is shown in Figure 2.1.

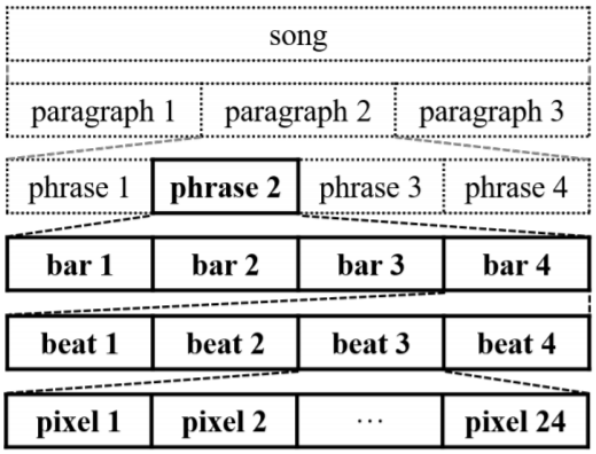


Figure 2.1: Structure of a music piece adopted from [1]

One of the difficulties in music generation is to evaluate the quality of the generated samples, indeed this cannot be done straightforwardly by visually inspecting the generated samples, as in the case of images. To this purpose, different metrics have been proposed and adopted in the literature, both quantitative and qualitative with different approaches.
The authors of [2] claim that one of the biggest challenges in designing a good model is given by the necessity to take into account the long-term structure. The long-term structure is obtained when there is coherence in the entire composition which usually contains repeated patterns distant in time and takes into account the global structure of a piece. Opposed to that, in music generation, there is also the need for a short-term structure, which is usually easier to learn for neural models. The short-term structure refers to the local relationships between a note and its preceding and following notes, and it is related to a short time frame.

2.2 Music representations

The representation of music is basically how a composition is "stored" to be read and played by a musician or a computer. It is crucial to find the proper representation which is best suited to algorithmic music generation. The two most used representations for generative tasks are divided into the symbolic and audio domains.

2.2.1 Symbolic

The symbolic representation consists of writing music using symbols. This type of representation encodes only the sequence of notes to be played along with other instructions to play the composition without storing information about the actual sound. For this reason, this kind of representation is lighter and easier to handle for generative models, on the other hand, it leaves space for the interpretation of the notes that can be played with different instruments. Thus, to have the listenable audio, the generated symbols need to be postprocessed with audio-synthesis tools.

MIDI

The MIDI format is the most used symbolic representation. Musical Instrument Digital Interface is a technical standard that describes a protocol, an interface, and electrical connectors that connect different electronic musical instruments [3]. A MIDI file is composed of events, each event is made up by two parts: time and message, its structure is reported in Figure 2.2.

time message time message time message time message
time message time message time message time message
time message time message time message time message
time message time message time message time message
time message time message time message time message ....

Figure 2.2: Structure of a MIDI file adopted from [4]

The time value refers to the amount of time to wait before playing the following message. There are two possible types of messages: command or data is distinguished by looking at the most significant bit of a byte. If the bit is 0, the associated byte is a data byte, otherwise, it is a command byte.

The most used commands in MIDI files are note-off and note-on which allow to stop or to start playing a note. The MIDI format has the drawback that in multitrack composition it does not preserve the notion of multiple notes played at once [5]. The reader who wants to investigate more the MIDI protocol and its functioning is referred to [6].

Pianoroll

The pianoroll representation is a symbolic representation that represents music in an image-like format, by looking at Figure 2.3 is it possible to notice that the x-axis represents the time while the y-axis represents the pitch. The time can be absolute or symbolic, the first reports the actual time related to the note occurrence, while in the second the tempo information is removed, keeping the same length for each beat. In this work, the symbolic timing is adopted and each beat has a temporal resolution of 4. [1] The note events are binarized, so each "note on" corresponds to one while "note off" corresponds to zero.

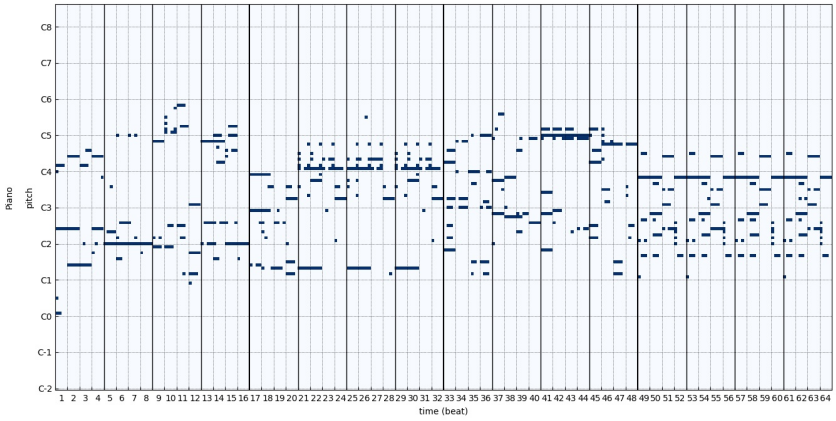


Figure 2.3: Example of a pianoroll

In this project, the pianoroll are created by using the python library "Pypianoroll" [7]. This library lets easily manage pianoroll representation with Python and it is created and was first introduced in MuseGAN [1]. With respect to the MIDI format, the pianoroll has the main limitation that there is no information about note-off and so it' impossible to discriminate between a rapidly repeated short note and a long note. A further and complete comparison for symbolic music data representations can be found in [8].

2.2.2 Acoustic

The acoustic representation of music encodes the audio signal generated when the music piece is played.

Waveform

The "waveform" is the most intuitive acoustic representation, the audio is basically a signal with time on the x-axis and amplitude on the y-axis. A waveform representation is shown in Figure 2.4. The advantage of using the waveform is that this representation holds the highest level of details since it is the "raw" signal, however this comes at the cost of increased computational load. [3]

Figure 2.4: Waveform Representation

Spectrogram

The spectrogram is a transformed representation that is obtained from the raw audio applying a Short Time Fourier Transform. Since it is a transformation, some information is lost in the process. The spectrogram is a visual representation of the spectrum of the audio signal, an example can be found in Figure 2.5.

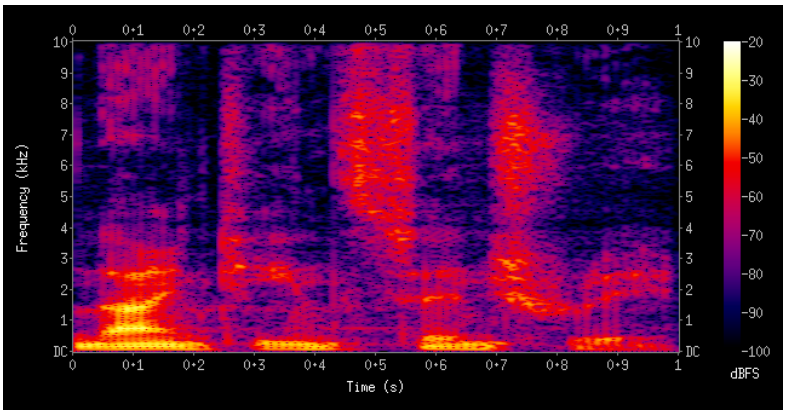


Figure 2.5: Spectrogram Representation

On the x-axis is reported the time (in seconds) while on the y-axis the frequency (in KHz). The color represents the intensity of the audio in dB.

2.3 Emotions in Music

Emotions are a strong subjective feature of music. This concept was first investigated in the literature for the task of Music Emotion Recognition (MER), in [9] the authors performed a state-of-the-art review of this field. MER is a subfield of Music Information Retrieval (MIR) that deals with emotions.

In the years researchers tried several approaches with the purpose of classifying emotions from music, using different kinds of techniques such as linear regression [10] or more recently using Convolutional Neural Networks (CNN) [11] [12]. Moreover, different audio features for this task have been identified and a comprehensive survey is present in [13].

In this thesis the emotions in music are considered for the purpose of generating melodies that represent a target emotion, this has been done recently with several attempts. The related field is called Affective Music Composition and deals with music composition with the intention to arouse a specific emotion in the listener and is analyzed in [14].

Different kinds of systems have been developed to generate music with a given sentiment. MetaCompose [15] exploits genetic algorithms and sophisticated evolutionary techniques to generate real-time music that can express different mood-states.

In [16] the authors generate music looking at the facial emotion of a subject employing a Long Short-Term Memory network (LSTM). The LSTM has been used a lot in music generation tasks since it works sequentially predicting one note after the other, which is basically similar to the way music composers write music [17]. Also in [18] the authors designed a single layer multiplicative LSTM-based architecture to generate music with sentiment. For this purpose, they created a dataset called VG MIDI annotated with emotions, that has been used in this work.

2.3.1 Models of Emotions

There are two big groups of models of emotions: categorical and dimensional. These psychological models are useful to reduce the complex emotion space into a practical set of categories [19].

The categorical models use categories and labels under which similar emotions are grouped together, the study conducted in [20] by Henver identifies 8 principal groups of emotions that are reported in Figure 2.6.

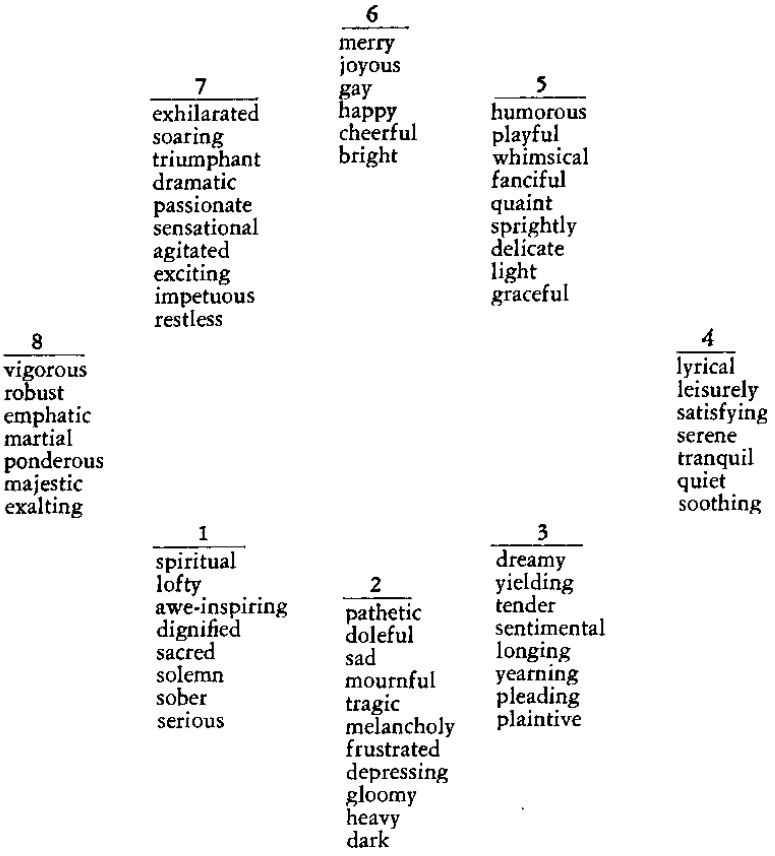


Figure 2.6: Categorical Model of Henver adopted from [20]

On the other hand, dimensional models describe a given emotion using real valued numbers, typically in 2 or 3 dimensions. The most famous dimensional model is the valence-arousal model by Russell [21]. A representation of this 2-dimensional model is reported in Figure 2.7 in which the x-axis represents valence while the y-axis arousal and different emotions are associated with different valence-arousal pairs.

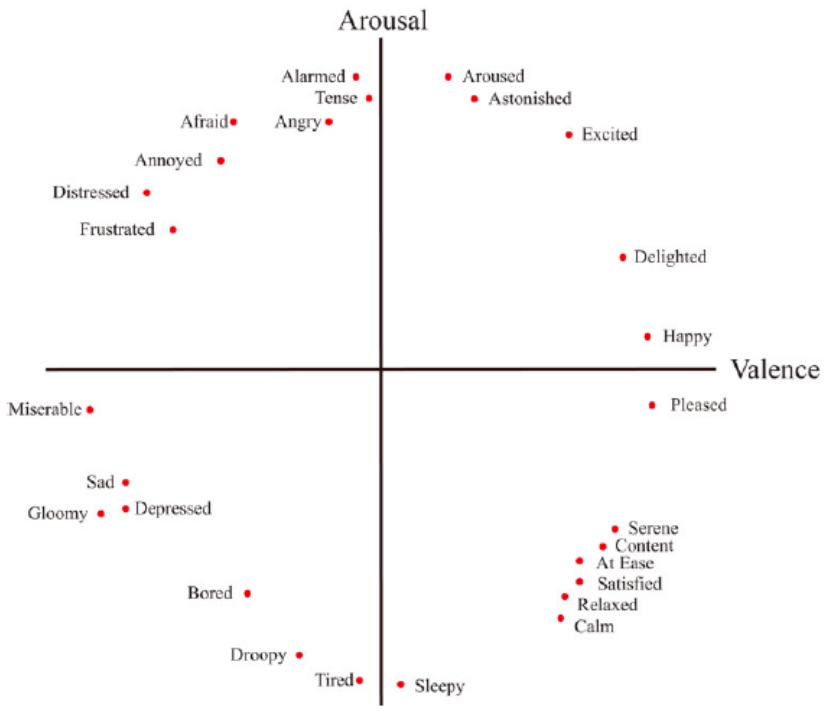


Figure 2.7: Dimensional Model of Russell adopted from [21]

These models can be used when working with music and different datasets containing music emotions can be annotated following a categorical or a dimensional model. The dataset used in this thesis, called VG MIDI, has been annotated with a valence-arousal model [18]. The advantage of a dimensional model is that it allows continuous annotation while listening to a target music piece and then if it is needed for a multiclass task the annotation can be grouped and converted into categorical.

2.4 Generative Adversarial Networks

Generative Adversarial Networks (GANs) have been introduced by Ian Goodfellow [22] in 2014. Since then, they have had a rapid development, becoming the state-of-the-art in image and video generation tasks ([23], [24], [25]). In this section, the literature about GANs and their evolution in the years is explained starting from the first GAN model. Finally, some architectures developed with GAN to specifically work with music are shown.

2.4.1 GAN models

In this section, some of the most used and popular GANs models are examined.

Vanilla GAN

The first version of GAN was introduced in [22]. Generative Adversarial Networks are a particular type of generative models. A generative model is able to learn the distribution of a dataset (p_{data}) and represent an estimation of this distribution (p_{model})[26]. The GAN framework, shown in Figure 2.8, is basically composed by two parts:

- Generator: learns to generate fake images looking as real as possible.
- Discriminator: learns to distinguish between real and fake samples.

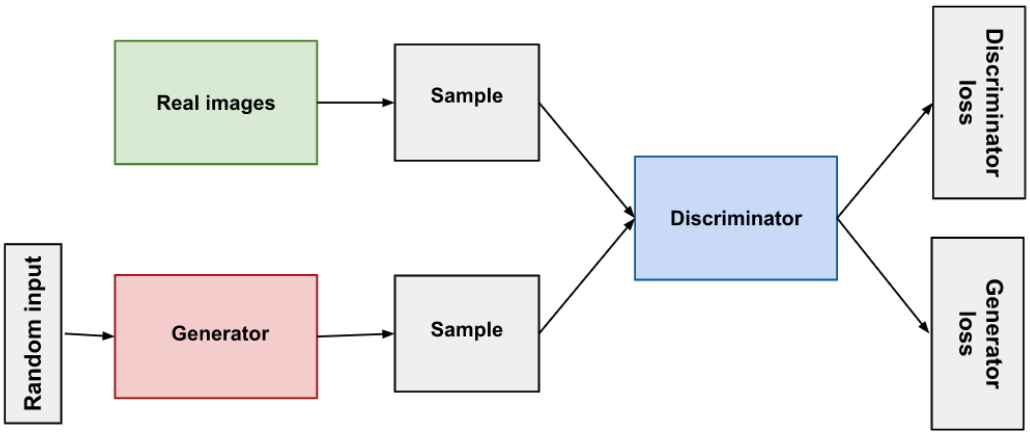


Figure 2.8: Structure of GAN adopted from [27]

The generator and the discriminator play an adversarial min-max game which is given by the loss function in eq. 2.1:

$$
\begin{aligned}
J^{(D)} &= \mathbb{E}_{x \in p_{data}}[logD(x)] - \mathbb{E}_z[\frac{1}{2}log(1 - D(G(z)))] \\
J^{(G)} &= -J^{(D)}
\end{aligned}
\tag{2.1}
$$

Where:

- x represents a real data sampled from p_{data} distribution
- z represents a random vector sampled from the latent space distribution p_z
- G(.) and D(.) represent the generator and discriminator network, respectively.

The Discriminator is basically a classifier that is trained to distinguish real samples from fake samples. When the output of the classification is 1, the sample is real, while when it is 0 the sample is fake. The discriminator loss is evaluated by comparing how well real samples D(x) (which should be as close as possible to 1) and fake samples D(G(z)) are classified (which should be as close as possible to 0).

On the other hand, the generator is trained to fool the discriminator, indeed its cost function is the opposite of the one of the discriminator.

The training of a GAN consists in finding Nash equilibrium of this two-players non-cooperative game [28]. Intuitively, Nash equilibrium is reached when each player has the minimum possible cost. However, finding this equilibrium point is really difficult, especially for the GAN case in which the cost functions are non-convex and the space of parameters has high dimensionality.

In the years, researchers have introduced different heuristic-based methods to improve stability and convergence in GANs training process, some of these will be discussed in the following sections ([28], [29], [30]).

DCGAN

The first significant improvement in GAN was introduced in [31], where it is demonstrated the feasibility to utilize deep convolutional layers inside the architectures of the generator and discriminator, and this lead to overall increased performance.

DCGAN stands for Deep Convolutional Generative Adversarial Networks, the authors who first introduced this architecture in [31] identified general guidelines for developing stable deep convolutional GAN architectures, that are reported here:

- Replace any pooling layer with strided convolutions for the discriminator and fractional-strided convolution for the generator.
- Use the Batch Normalization ([32]) in both generator and discriminator.
- Remove fully connected hidden layers for deeper architectures.
- Use ReLU activation ([33]) for all layers in the generator except for the last which uses Tanh.
- Use LeakyReLU activation ([34]) for all layers in the discriminator.

Figure 2.9 illustrates the architecture of the generator, which receives as input a 100-dimensional random vector sampled from a uniform distribution and thanks to a concatenation of fractional-strided convolutions outputs a 64x64 image.

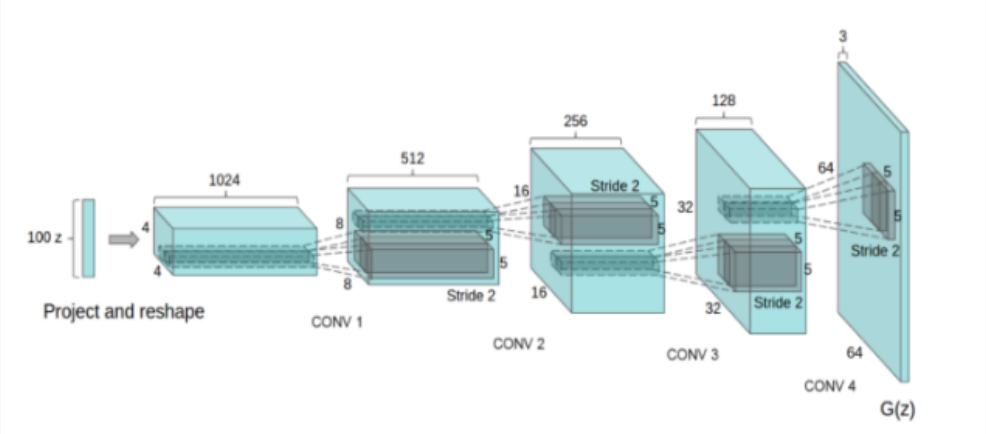


Figure 2.9: DCGAN generator adopted from [31]

The architecture of the discriminator mirrors the generator's one: it receives as input a 64x64 image that passes through a series of convolutional layers and it is classified with a scalar number between zero and one.

WGAN-GP

In this section, the Wasserstein GAN with Gradient Penalty is analyzed. This model derives from the need to overcome the GAN training difficulties, indeed with the vanilla GAN the training is very unstable, and there are several problems such as vanishing gradient and mode collapse. In [35] the authors introduce an algorithm called WGAN that is an alternative to the original GAN training, thanks to their contribution the problems peculiar to the training process are reduced. Indeed, training WGAN is less sensitive to the balance between generator and discriminator in training and to the architecture design. In addition, the mode collapse phenomenon is partially overcome.

The WGAN is based on the modification of the loss function: the original Jensen-Shannon divergence used in the loss function in eq. 2.1 is substituted with a new loss function called the Earth Mover's Distance (EMD) or Wasserstein-1. Intuitively, considering two distributions and thinking about the first as a mass of earth properly spread in space and the second as a collection of holes in that same space, the EMD measures the least amount of work needed to fill the holes with earth. An extensive explanation of EMD is present in [36].

After an appropriate mathematical adaptation of the EMD, (the evolution process from GAN to WGAN is carefully explained in [37]) the new value function proposed in [35] is :

$$\min_{G} \max_{D \in \tilde{D}} \mathbb{E}_{x \in p_{data}}[D(x)] - \mathbb{E}_{z}[D(G(z)] \tag{2.2}$$

Where $\tilde{D}$ is the set of 1-Lipschitz continuous functions, briefly this condition is met when the norm of its gradient is lower than 1 at any point. The others terms assume the same meaning they had in equation 2.1.

In simpler words, in this case, the generator wants to minimize the distance between the true and the generated distributions, while the discriminator (now called critic) wants to maximize this distance by keeping apart the two distributions. The WGAN critic now has in output a linear layer instead of the sigmoid layer usually used in the discriminator, so the output of the critic is no more comprised between 0 and 1 but could be any real number. Thus, the critic now outputs a value that is no more a discrimination between real and fake samples, but an indication about how real an image is considered by the critic.

Since the EMD is continuous and differentiable almost everywhere the discriminator can be trained 'till optimality. Indeed, during training, the gradient becomes more reliable. On the opposite, for the Jensen-Shannon divergence of the original formulation, as the discriminator improves the gradient becomes 0, thus leading to the vanishing gradient problem.

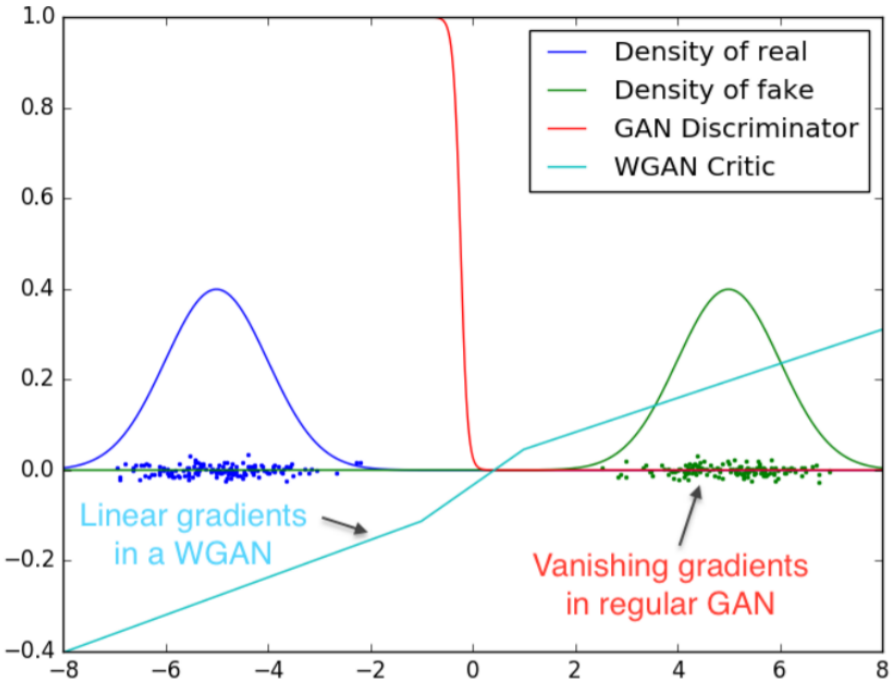


Figure 2.10: Gradient Comparison adopted from [35]

Figure 2.10 shows the optimal discriminator and critic when learning to differentiate two Gaussians, it is clearly visible that the discriminator of the regular GAN saturates.

To summarize, using the Wasserstein distance produces a value function that has better theoretical properties than the original formulation, thus obtaining better performance during the training.

One limitation of the W-Loss is that the critic should belong to the set of 1-Lipschitz continuous functions, in [30] it is introduced a method to enforce this 1-L continuity which has better performance than the weight clipping firstly introduced in W-GAN. This method consists of introducing an auxiliary regularization term to the loss function which encourages the norm of the gradient to be smaller than one, this is a soft constraint that has proved to be really effective. To get the GP first it is needed to sample some points from the real and generated data distributions and interpolate between them. This random interpolation is where the gradient penalty is evaluated on and the associated distribution will be called $p_{\hat{x}}$.

The new loss function for the critic becomes:

$$L = \mathbb{E}_{x \in p_{data}}[D(x)] - \mathbb{E}_z[D(G(z)] + \lambda \, \mathbb{E}_{\hat{x} \in p_{\hat{x}}}[(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2]$$

(2.3)

Where:

- λ is the penalty coefficient ($\lambda = 10$ is the value suggested in [30])
- $p_{\hat{x}}$ is the sampling distribution, defined as sampling uniformly along straight lines between pairs of points sampled from p_{data} and p_z
- All the other terms have the same meaning they have in eq. 3.1

Controlling the generation of GANs

The GAN models we have seen until this point have been used to perform unsupervised representation learning, so they did not require any label in the training dataset. They basically have been used to learn $P(X)$ where X is a set of features, however, conditional models can learn $P(X|Y)$ where Y represents the class label or another additional information. Learning to generate specific class samples is desirable to have a higher control on what is generated. In [38] researchers proved that it is possible to condition the generation of the GAN with additional information (that will be called y) and this may lead to good performance in terms of sample quality and variety.
The conditional architecture has been applied to the Vanilla GAN case but can be applied to other variants. The proposed method consists of feeding y as an additional input layer into both generator and discriminator. Regarding the generator, the additional information y is combined with the input noise z and there is flexibility in the way of performing this combination.

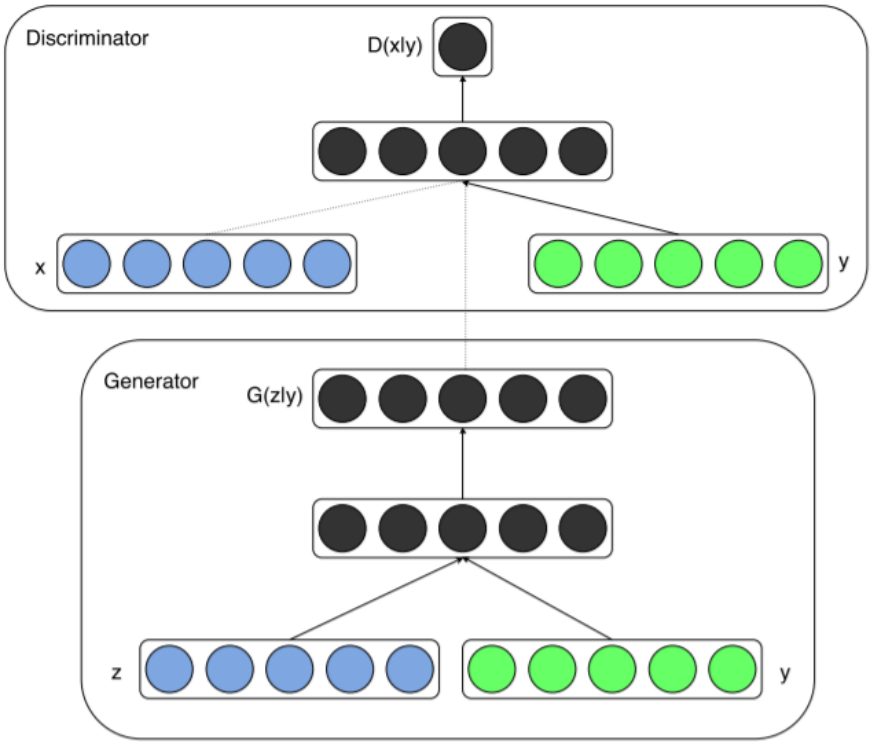


Figure 2.11: Conditional GAN adopted from [38]

Figure 2.11 shows the proposed structure for a simple Conditional GAN, for the discriminator x and y are both inputs for the discriminative function.

The value function of the two-players minimax game of eq. 2.1 will become:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x \in p_{data}}[logD(x|\mathbf{y})] - \mathbb{E}_z[log(1 - D(G(z|\mathbf{y})))] \tag{2.4}$$

This is the first and basic way that has proven to be successful in conditioning the generation of a GAN, however, in the years researchers have proposed several other methodologies such as Spatial Bilinear Pooling, Information Retrieving GANs [39], Auxiliary Classifier GANs [40], and many others. Conditional GANs have success in generating samples with desired major features, however, they fail in controlling the generation of lower-level details that distinguish one sample from another. With these premises, the authors of [41] introduced a new approach to control the generation. Other methods leverage the exploration of the latent space to control the generated feature, walking in the latent space, given that the desired features are disentangled, it is possible to control the generation process, these

walking techniques are explored in [42]. In [43] and [44] the authors applied the latent space exploration for semantic face editing, obtaining very good results.

2.4.2 GANs applied to Music

In this section two state-of-the-art GANs applied to music are shown. The first one called GANSynth is applied to audio synthesis, while the second MuseGAN to symbolic music composition.

GANSynth

Researchers of Google Magenta Team recently developed GANSynth [45] to perform efficient audio synthesis, starting from symbolic music (MIDI) to audio. This is the first model that synthesizes audio using GAN and achieves better quality than autoregressive models, as WaveNET [46]. Moreover, since WaveNET and all the regressive models work sequentially, GANSynth is significantly faster ($\sim$54,000 times faster) in generating high fidelity audio that is locally coherent and has a global structure.

The model uses log-magnitude spectrograms which are preprocessed carefully to obtain high-quality results. Moreover, global latent control is obtained by appending the conditioning latent vector to all the stacked transposed convolutional layers.

The model is a DC-GAN with Wasserstein loss and Gradient Penalty. In addition, it is trained using a method called Progressive Training [29], that increases the quality and stability. The method basically consists in training the first layers and then progressively adding the other layers one by one as the training goes on.

In order to achieve independent control on pitch and timbre, a one-hot representation of pitch is concatenated to the latent vector as in [38]. Moreover, to encourage the generator to use the pitch information, the authors add a classification loss to the discriminator.

By conducting an empirical evaluation of the performances of GANSynth the authors empirically demonstrated that their work outperforms the previous baseline.

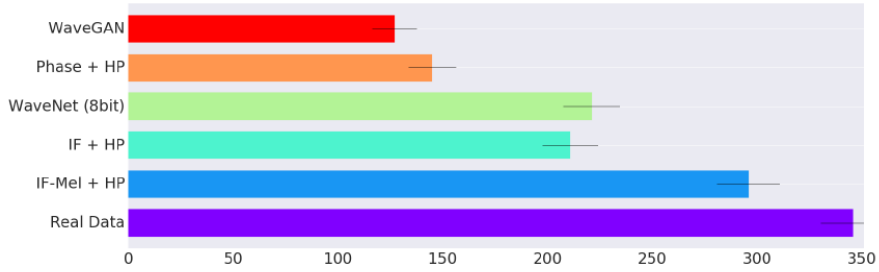


Figure 2.12: Empirical Evaluation reported from [45]

Figure 2.12 shows the number of wins on a pairwise comparison between different models with different representations: the Intermediate Frequency Mel transformation, with High-resolution sampling frequency achieves the best results.

MuseGAN

MuseGAN [1] is a network for multitrack sequence generation that can generate polyphonic music either from scratch or conditioned on a given track.

The proposed architecture is a classic WGAN-GP with a complex architecture, in which a temporal model is employed to generate music with coherence among the bars. To obtain the temporal structure, a temporal Generator G_{temp} is used which maps the noise vector z to a sequence of some latent vectors. This sequence of latent vectors will be then used by the bar generator to generate each bar sequentially.

In order to model the multi-track interdependency, the authors proposed three models that work in different ways (Figure 2.13):

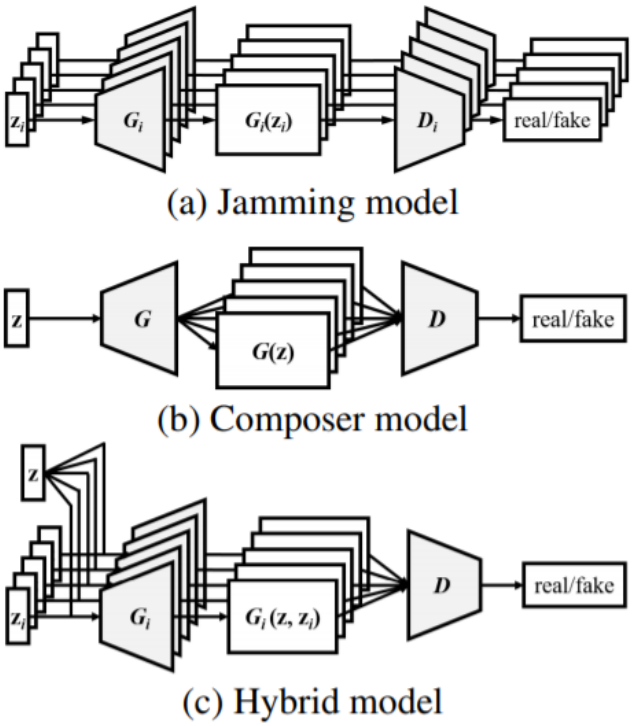


Figure 2.13: Different GAN models for music generation adopted in [1]

- **Jamming Model**: Each generator is independent and generates its own track from a private random vector, with different discriminators for each track. To generate music composed by M tracks, M generators and M discriminators are needed

- **Composer Model**: One generator creates a multi-channel pianoroll, where each track is represented by a channel. The models need only one generator and one discriminator, regardless of the number of tracks.

- **Hybrid Model**: Each of the private generators takes an inter-track random vector, which can coordinate the generation (z), and an intra-track random vector (z_i). The model to generate M tracks needs M generators and one discriminator.

MuseGAN has been evaluated with different metrics, with both a quantitative and qualitative evaluation, and the results showed good performance, although the generated music is not close to the level of that created by human composer.

2.5 Model Evaluation

The evaluation of a music generation system is a challenging task, indeed it is not straightforward to assign a quantitative score to benchmark the performance of a model. In this section is presented the standard quantitative metrics to evaluate GANs and some music-related metrics.

2.5.1 Fréchet Inception Distance

The Fréchet Inception Distance (FID) is the most used metric to quantitative evaluate the performance of GANs. It was introduced in [47] for the evaluation of images. The authors proved that the FID captures the similarity between generated and real images better than the previous standard: the Inception Score (IS). In essence, the FID measures the distance between feature vectors evaluated for both generated and real samples.

The features are extracted taking the activations of the final layer of an inception model, which is basically a neural network trained to classify images.

The authors in the original paper ([47]) define the FID as:

$$d^2((\mu, \Sigma), (\mu_w, \Sigma_w)) = \|\mu - \mu_w\|_2^2 + Tr(\Sigma + \Sigma_w - 2(\Sigma\Sigma_w)^{\frac{1}{2}}) \tag{2.5}$$

Where:

- (μ, Σ) are the feature-wise mean and covariance matrix for the real samples
- (μ_w, Σ_w) are the feature-wise mean and covariance matrix for the generated samples

The FID, also called the Wasserstein-2 distance, is used to measure the difference between two Gaussians.

2.5.2 Musical Metrics

To evaluate music generative models, the authors of the library *Muspy* [48] proposed a set of musical metrics. These metrics can be used to evaluate the "goodness" of the model by comparing the statistical difference between real and generated samples. The number of metrics is large, but they can be grouped in two categories:

- Pitch related metrics
- Rhythm related metrics

In this section are listed and explained the metrics used in the remainder of the work to evaluate the proposed generative model which are:

- **Pitch Range**: defined as the range of the used pitches
- **Pitch Classes Used**: defined as the number of unique pitch classes used.
- **Polyphony**: defined as the average number of pitches being played at the same time. It is evaluated at time when at least one pitch is on.
- **Polyphony Rate**: defined as the ratio between the number of timesteps when multiple pitches are played concurrently to the total number of timesteps (used in [1]).
- **Scale Consistency**: defined as the largest pitch-in-scale rate over all major and minor scales (used in [49]).
- **Empty Beat Rate**: defined as the ratio of the number of empty beats to the total number of beats.

Chapter 3

Methodology

In this chapter, the methodology used to reach the objective of conditioning symbolic music generation is carefully reviewed. First, the description of the used dataset and the developed model is presented, then the training process is explained. Successively, different attempts to control the MIDI generation are presented. Finally, the approach for performing the model evaluation is explained in section 3.5.

3.1 Dataset

The used dataset is called VG MIDI, it has been developed in [18] and already introduced in section 2.3.1.

This dataset is the only publicly available dataset containing MIDI files annotated with the label of emotions. It was developed with the purpose of generating music with sentiment, the authors of [18] used it to generate emotionally "positive" and "negative" pieces.

VG MIDI is composed of 3735 pieces extracted from video game soundtracks in MIDI format. The choice of video game soundtracks is appropriate since they are used to keep the player in a particular affective state. The pieces are piano arrangements and their length spans from 30 seconds to 3 minutes. Out of the 823 pieces 95 are annotated with a valence-arousal dimensional model of emotion, explained in section 2.3.1. This model allows greater flexibility with respect to a categorical model since the different a-v pairs can be directly mapped to discrete classes. Thus, it can be used for both classification and regression problems.

Each piece was annotated by 30 subjects using an online tool, the detailed procedure is explained in [18]. This annotation consists of an average time series of valence-arousal pairs for all the subjects. Then, the results were preprocessed splitting each piece into phrases of the same sentiment, obtaining 205 distinct

labelled phrases.

The 205 distinct labelled phrases contain values of valence-arousal that are discretized by tresholding: for each of the two dimensions the possible values can be $+1, -1$. Then, the phrases are grouped in 4 distinct categories based on their position on the valence-arousal plane. It is straightforward to associate those 4 resulting classes to the classes proposed in [19].

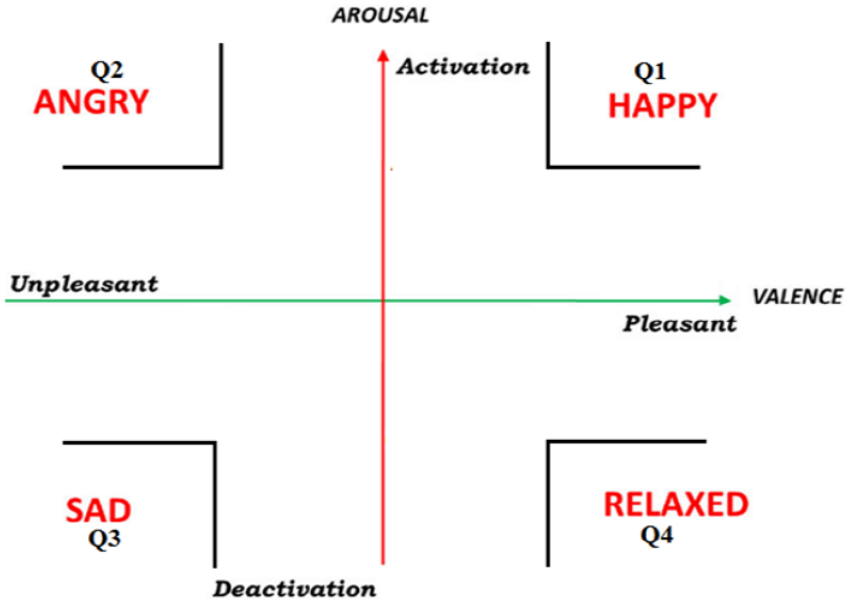


Figure 3.1: Mood classes in model of Russell adopted from [19]

Therefore, the mapping is performed according to the classes shown in figure 3.1, that are *Angry*, *Happy*, *Sad* and *Relaxed*. After this mapping, the number of samples per each class is shown as an histogram in Figure 3.2.

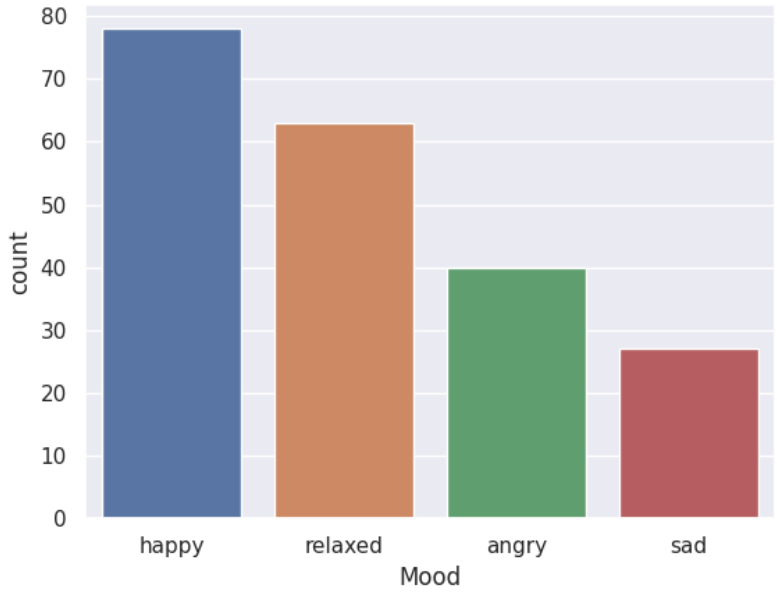


Figure 3.2: Histogram of number of samples per class.

The dataset is unbalanced: the majority class corresponds to the happy label, while the minority to sad pieces. This could lead to biases and problems when these data are used.

In this work, the MIDI files are preprocessed using the library "Pypianoroll" [7], that allows to easily convert from the MIDI format and perform manipulations on the pianorolls.

The preprocessing pipeline for a single sample, a phrase of a piece, is listed below, and is valid for both the labelled and unlabelled cases.

- The MIDI file is converted into pianoroll representation.
- The pianoroll is binarized, and the resolution of the beat is set.
- From the pianoroll a random number of different subsamples is taken, the number depends on the piece's length.
- The subsamples that contain too few notes are skipped.

Regarding the labelled dataset, for the classes that contain too few samples, the number of subsamples taken is increased to obtain a balanced dataset. The results of the preprocessing phase are reported in section 4.1.

3.2 Models

In this section, the different models developed to perform the conditioning experiments are explained.

3.2.1 Unconditioned GAN

The Unconditioned GAN model has been trained with the unlabelled dataset to generate musical pieces that should be similar to the real data.

The selected model is a WGAN-GP, it is inspired by MuseGAN [1], with an architecture modified to handle single track pianorolls.

Generator

The generator receives as input a random sample of the latent space and outputs a pianoroll with dimensions [1x64x72], where the entries stand for [#tracks, #timesteps, #pitches].

The architecture of the generator is based on 3d transposed convolutional layers, each convolution is followed by a 3d batch normalization layer. The basic block of the generator is composed by:

- 3d Transposed Convolutional Layer
- 3d Batch Normalization

The diagram representation of this basic block is reported in figure 3.3. The activation function for each block is the *ReLU*, except for the final one which has *tanh* activation, as suggested in [31]. The number of trainable parameters in the discriminator is 304787.

The detailed description for each layer, with the kernel sizes and strides used, is printed with the help of *Pytorch* and shown in Figure 3.4.

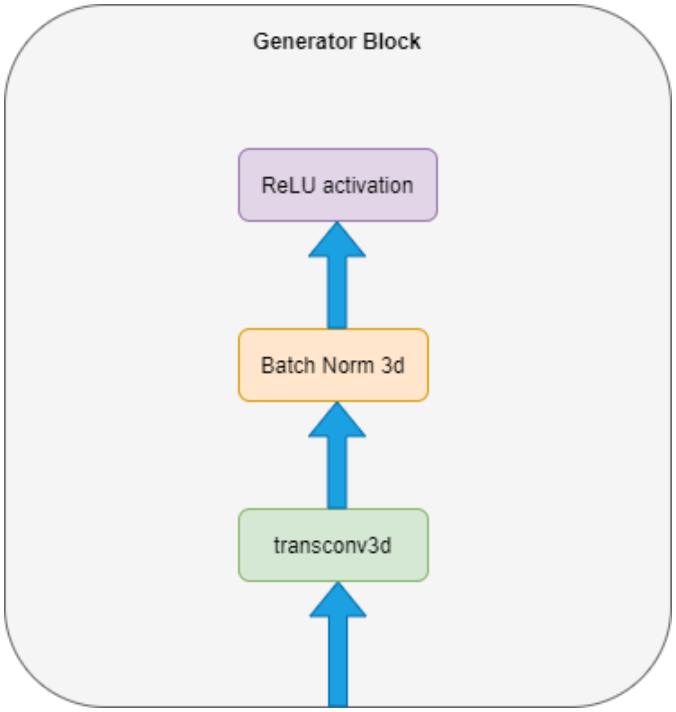


Figure 3.3: Diagram of the basic Generator Block

```
----------------------------------------------GENERATOR-----------------------------------------------

(transconv0): GeneratorBlock(
  (transconv): ConvTranspose3d(132, 256, kernel_size=(4, 1, 1), stride=(4, 1, 1))
  (batchnorm): BatchNorm3d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
)
(transconv1): GeneratorBlock(
  (transconv): ConvTranspose3d(256, 128, kernel_size=(1, 4, 1), stride=(1, 4, 1))
  (batchnorm): BatchNorm3d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
)
(transconv2): GeneratorBlock(
  (transconv): ConvTranspose3d(128, 64, kernel_size=(1, 1, 4), stride=(1, 1, 4))
  (batchnorm): BatchNorm3d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
)
(transconv3): GeneratorBlock(
  (transconv): ConvTranspose3d(64, 32, kernel_size=(1, 1, 3), stride=(1, 1, 1))
  (batchnorm): BatchNorm3d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
)
(transconv4): GeneratorBlock(
  (transconv): ConvTranspose3d(32, 16, kernel_size=(1, 4, 1), stride=(1, 4, 1))
  (batchnorm): BatchNorm3d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
)

(transconv5): GeneratorBlock(
  (transconv): ConvTranspose3d(16, 1, kernel_size=(1, 1, 12), stride=(1, 1, 12))
  (batchnorm): BatchNorm3d(1, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

)
```

Figure 3.4: Description of the Generator architecture

Discriminator

The discriminator is opposed to the generator, it takes as input a pianoroll with dimensions [1x64x72] and outputs a scalar indicating the confidence that the input is real as explained in sect 2.4.

The architecture of the discriminator is based on 3d convolutional layers, moreover, each convolution is followed by a layer normalization operation. The basic building block of the discriminator is composed by :

- 3d Convolutional Layer
- Layer Normalization

The diagram representation of this basic block can be seen in figure 3.5. The activation function for each block is the *Leaky ReLU*, with the negative slope set equal to 0.2. The number of trainable parameters in the discriminator is 186785.

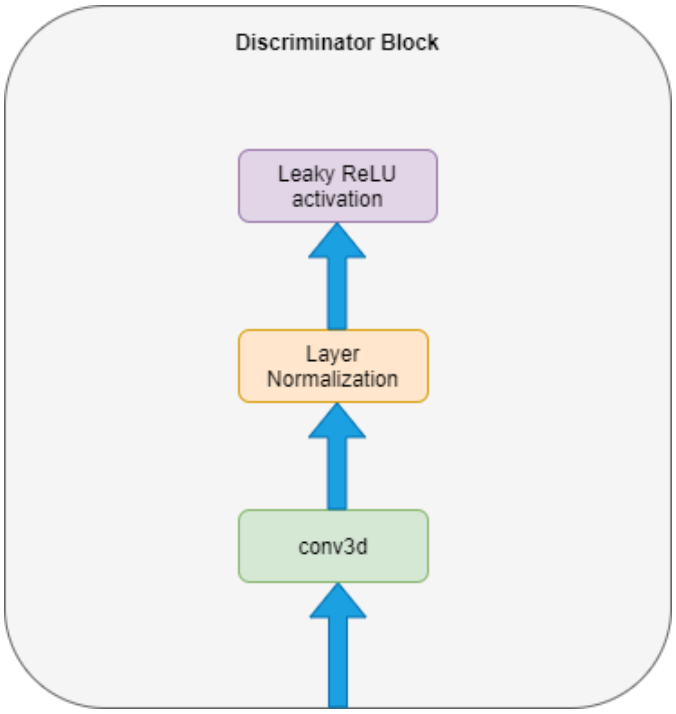


Figure 3.5: Diagram of the basic Discriminator Block

As for the generator, the detailed description for each layer, with the kernel sizes and strides used, is printed with the help of *Pytorch* and shown in Figure 3.6.

```
----------------------------DISCRIMINATOR-----------------------------------

(conv0): DiscriminatorBlock(
  (conv): Conv3d(1, 16, kernel_size=(1, 1, 12), stride=(1, 1, 12))
  (layernorm): LayerNorm()
)
(conv1): DiscriminatorBlock(
  (conv): Conv3d(16, 16, kernel_size=(1, 4, 1), stride=(1, 4, 1))
  (layernorm): LayerNorm()
)
(conv2): DiscriminatorBlock(
  (conv): Conv3d(16, 64, kernel_size=(1, 1, 3), stride=(1, 1, 1))
  (layernorm): LayerNorm()
)
(conv3): DiscriminatorBlock(
  (transconv): Conv3d(64, 64, kernel_size=(1, 1, 4), stride=(1, 1, 4))
  (layernorm): LayerNorm()
)
(conv4): DiscriminatorBlock(
  (transconv): Conv3d(64, 128, kernel_size=(1, 4, 1), stride=(1, 4, 1))
  (layernorm): LayerNorm()
)
(conv5): DiscriminatorBlock(
  (transconv): Conv3d(128, 128, kernel_size=(2, 1, 1), stride=(1, 1, 1))
  (layernorm): LayerNorm()
)
(conv6): DiscriminatorBlock(
  (transconv): Conv3d(128, 256, kernel_size=(3, 1, 1), stride=(3, 1, 1))
  (layernorm): LayerNorm()
)
(dense): Linear(in_features=256, out_features=1, bias=True)
```

Figure 3.6: Description of the Discriminator architecture

Different settings of parameters have been tried and the best performing model, considering the evaluation proposed in sect. 3.5 , has been selected.

3.2.2 Mood Classifier

The Mood Classifier has been created using an architecture similar to that of the discriminator of the GAN, which is supposed to extract relevant features starting from the pianoroll representation.

```
----------------------------------------CLASSIFIER----------------------------------------

  (conv0): Sequential(
    (0): Conv3d(1, 16, kernel_size=(1, 1, 12), stride=(1, 1, 12))
    (1): BatchNorm3d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): LeakyReLU(negative_slope=0.2, inplace=True)
  )
  (conv1): Sequential(
    (0): Conv3d(16, 16, kernel_size=(1, 4, 1), stride=(1, 4, 1))
    (1): BatchNorm3d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): LeakyReLU(negative_slope=0.2, inplace=True)
  )
  (conv2): Sequential(
    (0): Conv3d(16, 64, kernel_size=(1, 1, 3), stride=(1, 1, 1))
    (1): BatchNorm3d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): LeakyReLU(negative_slope=0.2, inplace=True)
  )
  (conv3): Sequential(
    (0): Conv3d(64, 64, kernel_size=(1, 1, 4), stride=(1, 1, 4))
    (1): BatchNorm3d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): LeakyReLU(negative_slope=0.2, inplace=True)
  )
  (conv4): Sequential(
    (0): Conv3d(64, 128, kernel_size=(1, 4, 1), stride=(1, 4, 1))
    (1): BatchNorm3d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): LeakyReLU(negative_slope=0.2, inplace=True)
  )
  (conv5): Sequential(
    (0): Conv3d(128, 256, kernel_size=(3, 1, 1), stride=(3, 1, 1))
    (1): BatchNorm3d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): LeakyReLU(negative_slope=0.2, inplace=True)
  )
  (dense): Sequential(
    (0): Linear(in_features=256, out_features=4, bias=True)
  )
```

Figure 3.7: Description of the Mood Classifier architecture

The main differences with respect to the discriminator are:

- The number of stacked convolutional layers is reduced from 6 to 5.
- The layer normalization is substituted by a 3-d batch normalization as in the generator.
- The last layer is a dense layer with 256 inputs and 4 outputs corresponding to the 4 mood categories and its activation function is a *Softmax*.

The kernel size and stride used are the same used in the discriminator and proposed in the official implementation of MuseGAN [1]. These hyperparameters have been searched to optimally extract information from the pianoroll representation.

The description of the Mood Classifier architecture can be found in figure 3.7, while the architecture of the classifier visualized with *Tensorboard* is reported in figure 3.8.

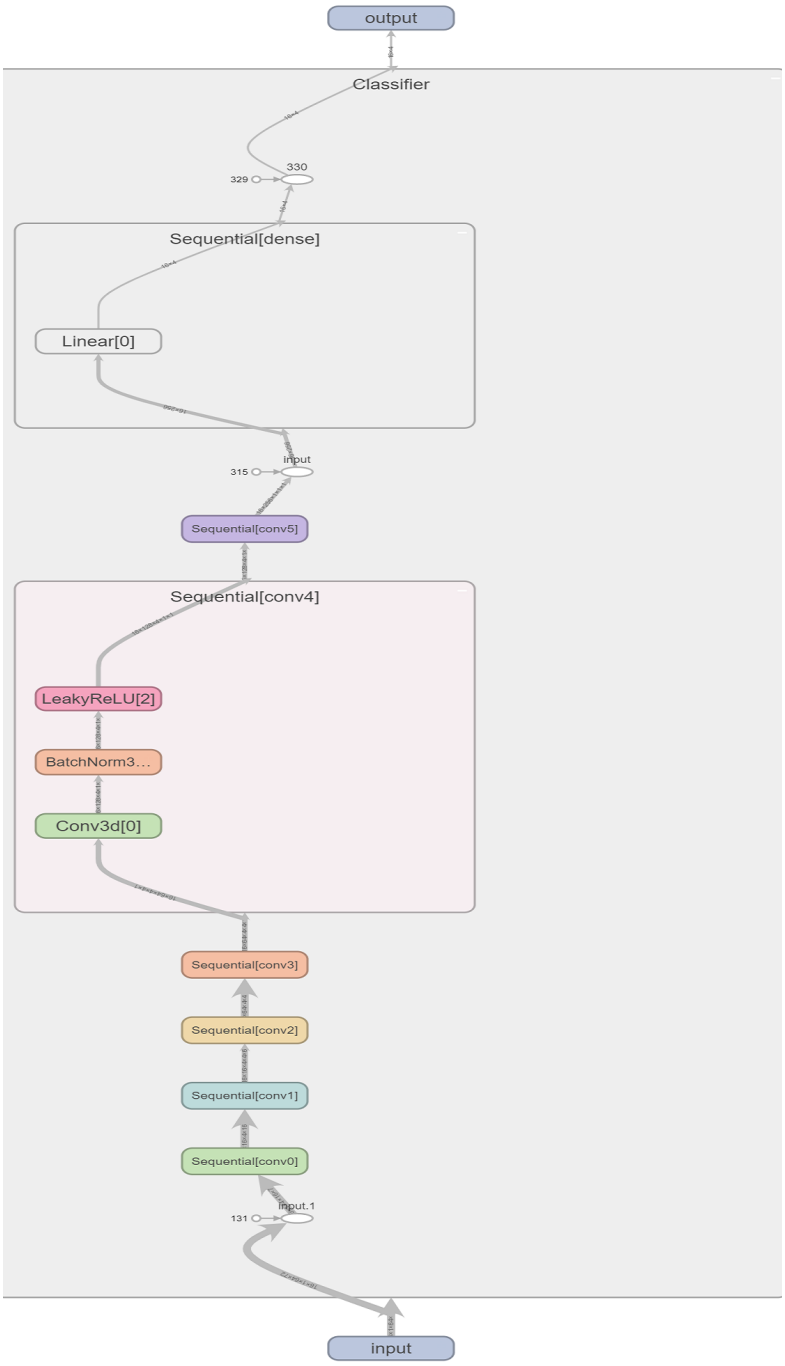


Figure 3.8: Mood Classifier architecture visualized in *Tensorboard*

3.3 Training

In this section, the training processes are explored for both the GAN and the classifier.

3.3.1 GAN Training

The GAN training process was built from the conventional algorithm for training the WGAN-GP model, which can be found in [30]. The training is unsupervised, and the unlabeled version of VG MIDI that was examined in sect. 3.1 was utilized.

First, the discriminator is fed with real samples and the loss related to real samples is evaluated, the same is done with fake samples. Then, the gradient penalty is evaluated. The discriminator loss is composed by the summation of the W-loss related to the fake samples, the W-loss related to the real samples, and the gradient penalty. The gradient of this loss is backpropagated and the weights of the discriminator are updated. The generator is updated once every five updates of the discriminator. A generator update consists in evaluating the prediction of fake samples with the discriminator, evaluating the W-loss, backpropagating the gradient and updating the weights.

In addition, further changes have been made to increase the training stability and convergence:

- Using Gaussian weight initialization: this can lead to better performances during training. Before the training process starts the weights are initialized with a Gaussian. The mean of the Gaussian is 0 with a standard deviation of 0.02 for the convolutional or transposed convolutional layers as suggested in [31]. While for the batch normalization layers the mean of the Gaussian is 1 and the standard deviation is 0.02.

- Two Time Scale Update Rule (TTUR): with this approach, the generator and the discriminator have different learning rates. TTUR was proposed in [47], where the authors show that this method can bring benefits to the training process. Indeed, the convergence of the training process is faster and the quality of the generated samples outperforms the conventional GANs performance. The used learning rates (LR) for the Adam optimizer are 0.0004 for the discriminator and 0.0001 for the generator.

- Adding Gaussian Noise to the input of the discriminator: a practice that has been proven successful in stabilizing the training and preventing the discriminator to overpower the generator is to add Gaussian noise to the real and fake samples before they are fed as input to the discriminator. The noise has mean 0 and a variance that starts from 0.1 and decays as the training goes on. This technique is proposed in [50].

The training hyperparameters are reported in Table 3.1:

Number of Epochs	10000
Batch size	256
Latent dimensions	128
Adam β_1	0.5
Adam β_2	0.9
LR Generator	0.0001
LR Discriminator	0.0004
λ for the GP	10

Table 3.1: GAN Training hyperparameters

3.3.2 Classifier Training

The Classifier has been trained using Binary Cross Entropy with Logits loss function from Pytorch, the details on this loss function can be found in [51]. The training is the usual training process for a classifier. For each batch, the network makes a prediction, relative to the mood, then the loss between the predicted and real classes is evaluated, the gradients are backpropagated, and the weights are updated.

The hyperparameters for the training process are reported in 3.2.

Number of Epochs	5000
Batch size	16
Adam β_1	0.5
Adam β_2	0.999
LR	0.0001

Table 3.2: Mood Classifier Training hyperparameters

The dataset used to train the classifier is the labelled VG_MIDI dataset described in sect. 3.1, after the preprocessing and balancing phases.

3.4 Controlling the Generation

The objective of controlling the generation according to a particular mood is very challenging. Three experiments are carried out in this part to achieve the intended goal, adopting various techniques.

3.4.1 Transfer Learning

A transfer learning technique is used in the first experiment to control the generation. Initially, the GAN is trained in an unsupervised manner with the unlabeled dataset. Following this initial training, the pretrained model is converted to a conditional architecture, and the C-GAN is trained to produce pieces from the dataset's four mood classes. The primary changes to the conditional architecture are as follows:

- Appending a one-hot representation of musical mood to the latent vector given as input to the generator

- Add an auxiliary classification loss to the discriminator so that it tries to predict the mood, in an effort to encourage the generator to exploit the mood information (as the authors of [45] did for the pitch).

In the unconditioned model, the last layer of the discriminator outputs a scalar. To add a classification term, the final layer's inputs are copied and delivered as input to a "classification" layer with a *Softmax* activation as output. The final classification layer outputs the probability of the sample belonging to a given mood class, the Negative Log Likelihood evaluated between this prediction and the target mood represents the previously mentioned auxiliary classification loss. This loss is multiplied by an extra hyperparameter (m), which is set equal to 10 because it is a commonly used number in the literature.

The conditioned GAN model was trained for 5000 epochs beginning from the unconditioned model, with the same hyperparameters as the unconditioned model. Furthermore, alternative class embedding sizes were explored to see if the dimensions of the attached class-related vector influenced the generation. The following mood information representations have been tested: one-hot representation, five-hot representation, and ten-hot representation. Working with four mood classes, the one-hot added vector has a size of four, the five-hot vector has a dimension of twenty, and the ten-hot vector has a dimension of forty. In the one-hot representation the dimension corresponding to the encoded class is set to 1 and the other to zero, analogously, in the ten-hot representation, ten dimensions corresponding to one class are set equal to one and the others to 0.

3.4.2 Latent Space Exploration

This experiment is based on the exploration of the Latent Space to control the generation. While conditional generation learns features during training, controllable generation consists in controlling specific features through the appropriate choice of the latent space starting point. Since some features are continuous, this approach is more flexible than conditional generation, indeed it allows to obtain a certain amount of a desired feature. In [52] the proposed technique is outlined and thoroughly examined.

With conditional generation, the model learns to represent the class that is requested, on the other hand, controllable generation achieves control on individual features. This control is obtained by tweaking the input noise vector in some direction that produces the desired features.

One possible approach for determining how to move in z-space in order to control certain characteristics is to utilize the gradient of a pretrained classifier. The trained GAN generator and the mood classifier are utilized to get a melody with the desired mood. First, a set of noise vectors is fed into the generator, which generates various pianorolls. Following that, the melodies are classified according to their mood, and the beginning input noise is shifted in the direction of the gradient. This gradient penalizes the generator for each pianoroll that does not match the specified mood.

Stochastic gradient ascent is used to update the input noise; this approach is typically used to discover the local maxima, as opposed to stochastic gradient descent, which finds the local minima. Thus, stochastic gradient ascent (SGA) is used to maximize the amount of the desired feature, in this case a certain mood; a mathematical definition of SGA is shown in eq. 3.1.

$$z_{i+1} = z_i + (w \nabla z_i) \tag{3.1}$$

Where:

- z is the latent vector
- i is the step in the iteration of SGA
- w is the weight by which the gradient of the latent vector is multiplied

The experiment is performed with the 4 moods by inspecting the difference and the results obtained are commented in sect 4.3.2.

3.4.3 Generate and Filter

This experiment is focused on randomly generating melodies with the unconditioned GAN model and filtering those that have the required mood. The filtering procedure is accomplished through the use of the Mood Classifier, which discards melodies categorized with a mood other than the desired one. To put it in another way, the generator generates random melodies in a loop that ends when the desired emotion is detected. Figure 3.9 depicts a schematic of this pipeline.

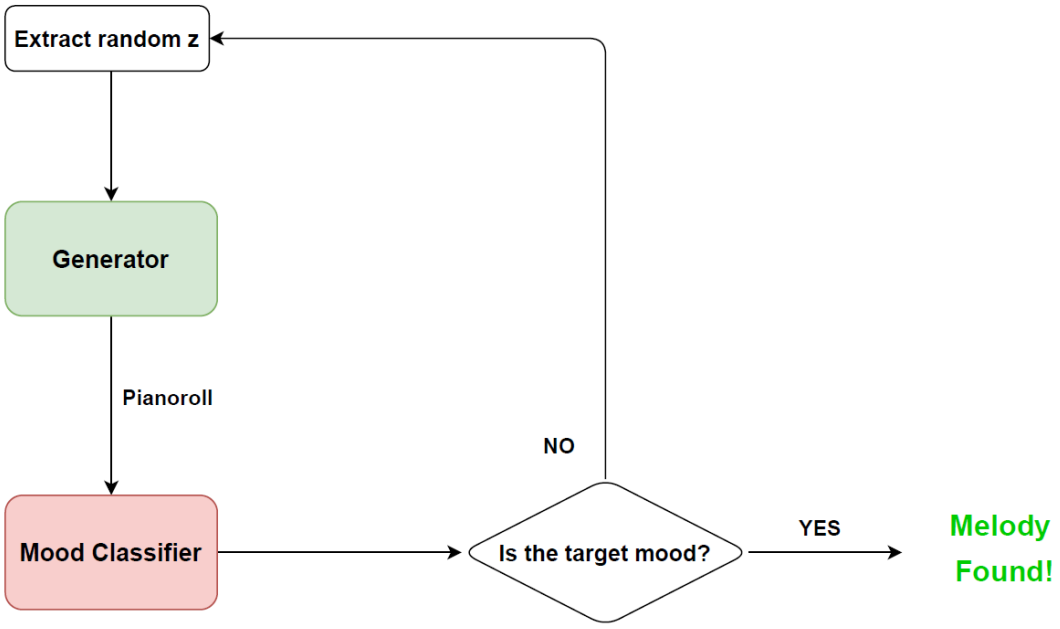


Figure 3.9: Generate and Filter flowchart

This approach is simpler than the others and more expensive in terms of computations required, but the relatively quick creation and classification time makes it practicable. In fact finding the required melodies takes less than a minute. Furthermore, one of the benefits of this technique is that various filters may be used throughout the process, for example, some musical metrics can be computed for each created piece and multiple thresholds can be specified, so that melodies with scores lower than the given threshold are rejected.

3.5 Performance Evaluation

This section provides an overview of the techniques used to select and evaluate models. To choose the optimal model for the purpose of mood conditioned generation, quantitative measures are first assessed on the unconditioned GAN model. Then, the user study conducted to assess the quality of mood-conditioned samples is discussed.

3.5.1 Quantitative Metrics

The quantitative musical metrics explained in sect. 2.5.2, are evaluated on both the training data and the generated data. The statistical difference between these metrics is used as a reference value to assess the model quality.

In order to compute the difference between the distribution of the metrics on the generated and real samples, the FID, explained in section 2.5.1, is used. The FID evaluates the distance between the distributions for each feature and expresses it with a scalar.

The FID is evaluated every 200 epochs to pick the best model for the Unconditioned GAN that has been trained for 10000 epochs. A lower FID value implies that the produced samples' characteristics are more comparable to those of the real samples, and hence the model performs better.

3.5.2 Survey - A Qualitative Evaluation

Because emotions are very subjective, a quantitative evaluation of the suggested technique may not accurately reflect reality. A survey has been designed to determine if the produced samples arouse the intended emotion. Participants are asked to listen to eight samples with a duration of around 20 seconds and associate a mood with each sample, selecting from one of the four mood categories with the option of adding additional alternatives. The survey has two randomly generated items for each mood category, ensuring that the evaluation is balanced across classes. It was created with the aid of *"Google Forms"*, and one sample question (in Italian) is shown in Figure 3.10.

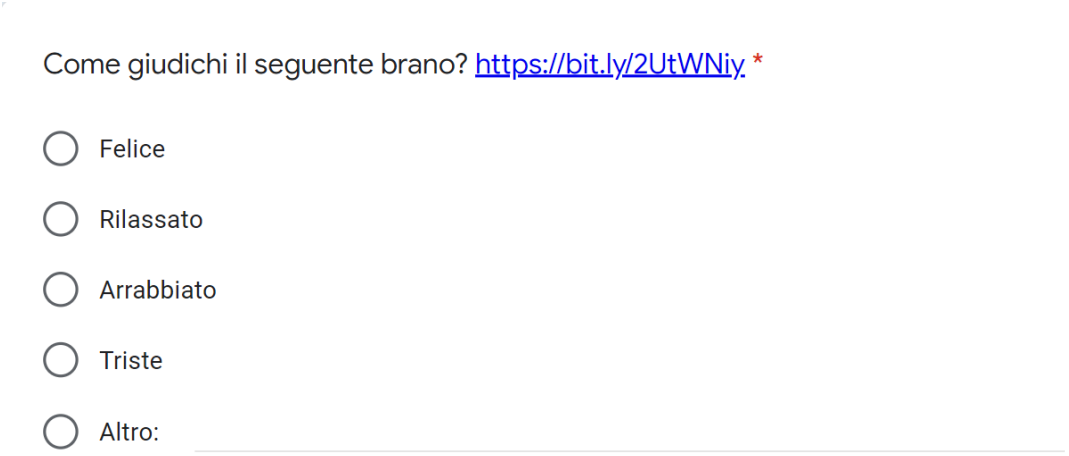


Figure 3.10: Example of a question from the survey.

Chapter 4

Results

This chapter reports the results obtained following the methodology illustrated in Chapter 3. The structure of the section is very similar to that of the methodology, reporting the results for each of the different work phases.

4.1 Data Preprocessing

The results of the preprocessing explained in section 3.1 are reported here.

From the unlabelled dataset, 15834 samples were extracted, while for the labelled one 1369. Since the dataset was originally unbalanced, the balancing process, which basically consists in oversampling the pieces that belong to a minority class, successfully compensated the difference in number between the classes.

The balanced distribution among classes is shown as an histogram in figure 4.1: with respect to the distribution shown in figure 3.2, now the classes contain more or less the same amount of samples. The new sample count in the balanced dataset is 2168. These datasets, both tagged and unlabeled, were utilized to explore mood-conditioned music creation throughout the course of this research.

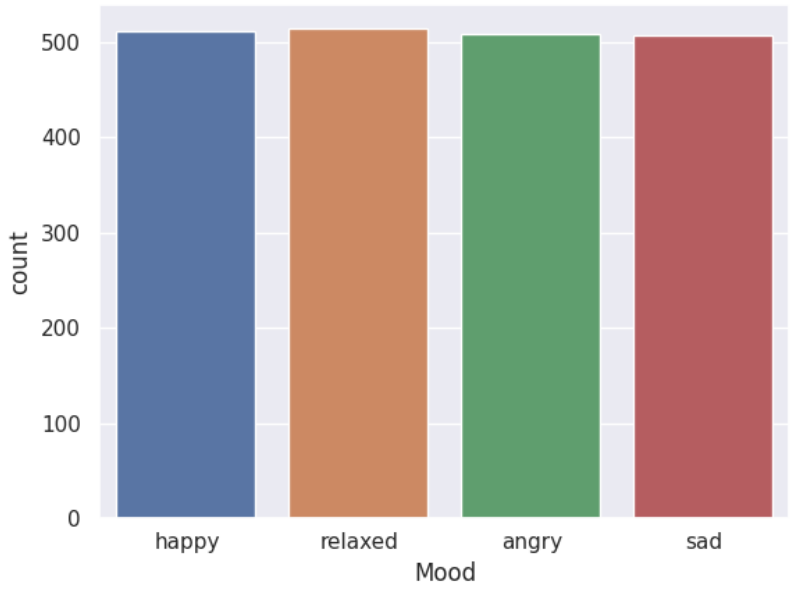


Figure 4.1: Histogram of number of samples per class - balanced.

4.2 Training Statistics

This section reports the results of the training process for both the Unconditioned GAN model and the Mood Classifier.

4.2.1 Unconditioned GAN

The GAN has been trained for 10000 epochs on the unlabelled samples in an unconditional way. The training process is discussed in section 3.3.1. The results of the training are reported for the best performing model.

Training a stable model is considered a very challenging task, since the training process is most of the times unstable. Moreover, there are no good objective metrics for evaluating whether a GAN is performing well during training. Instead, visual inspection of the created samples and subjective judgment are the best approaches. Working with music, on the other hand, is quite time consuming in terms of listening to and judging the created pieces for each model.

The loss of the generator over the epochs is reported in figure 4.2. Unfortunately the generator loss is not very meaningful in WGAN training and its value often does not correlate with the sample quality.

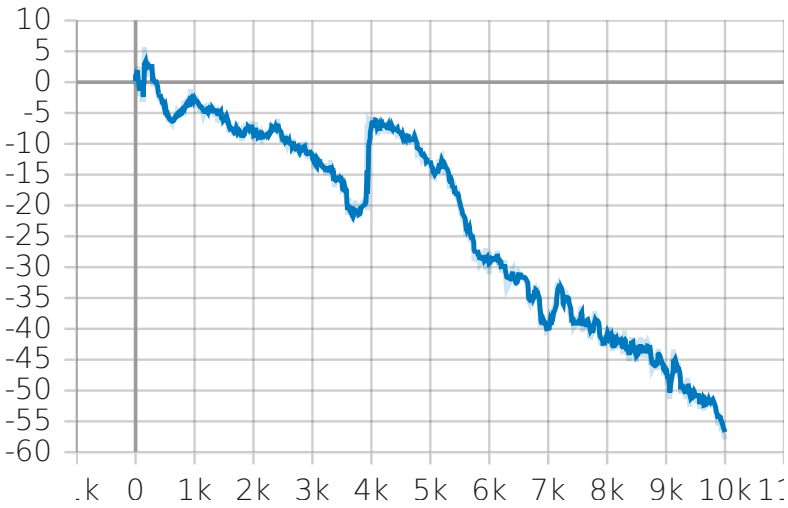


Figure 4.2: Generator Loss during training

From figure 4.2 it is possible to see that the loss is negative and decreases as the training goes on.

On the other hand, the discriminator loss in WGAN is an approximation of the negative Wasserstein distance between the generator distribution and the data distribution. Therefore, it is interpretable: indeed, higher values of the discriminator loss indicate worse performance.

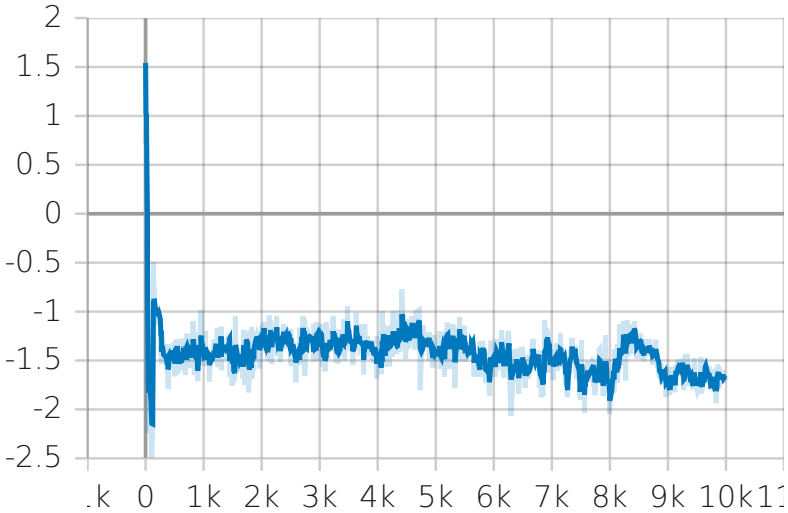
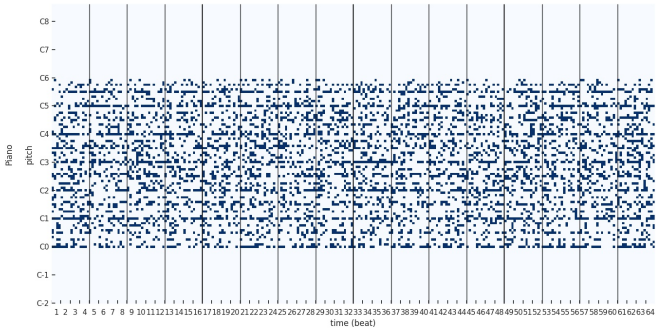
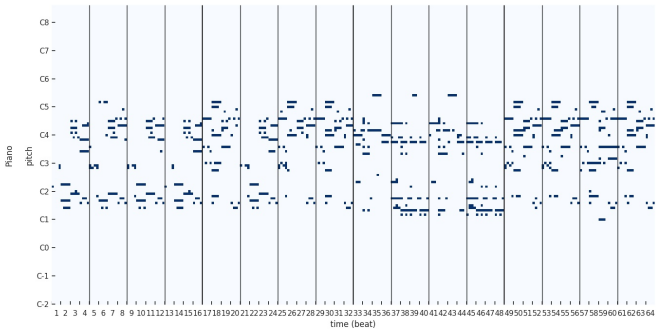


Figure 4.3: Discriminator Loss during training

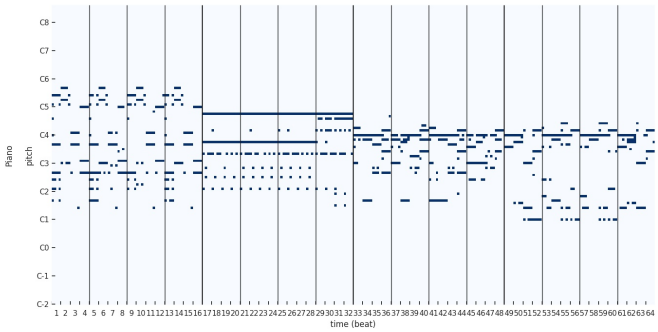
Figure 4.3 shows that the discriminator first oscillates between positive and negative values and then remains more or less constant for the rest of the training.



(a) Epochs 200



(b) Epochs 2000



(c) Epochs 6000

Figure 4.4: Evolution of generated samples at different epochs

Since the values of the losses are not really informative on the behaviour of the model, the improvements in the samples at different timesteps are visualized in figure 4.4. The first sample is clearly very noisy, and during the training the quality

of generated samples generally improves, showing more consistent patterns of notes in the pianorolls at epoch 2000 and epoch 6000.

To conclude, it is possible to state that even if the discriminator loss remains constant during the training, the quality of the generated samples has an overall improvement as the training process goes on.

4.2.2 Mood Classifier

The training process of the mood classifier has been carried out for 200 epochs. The loss function over the epochs is depicted in figure 4.5.

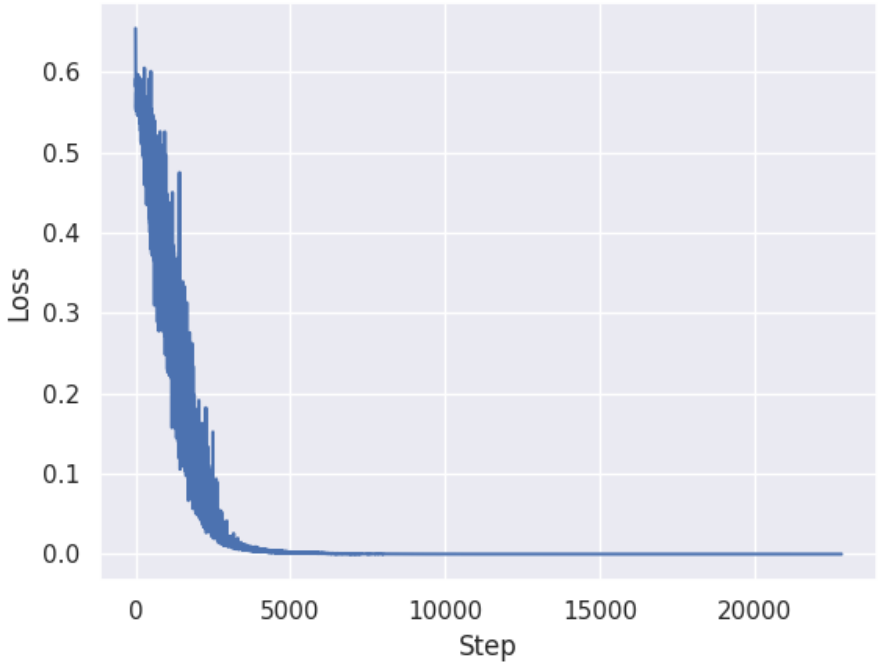


Figure 4.5: Loss of the classifier over training steps

The loss is reported versus the training steps. One training step consists in an iteration of the training process over a batch, i.e in one gradient update. The total number of steps is 22800 and since the training is run over 200 epochs, each epoch is composed by 114 steps. It is possible to observe that the loss begins to closely approach zero at step 4000, which corresponds to epoch 35.

The model is assessed based on its accuracy over the test dataset, which accounts for 10% of the total dataset, whereas the training dataset accounts for 90% of the total.

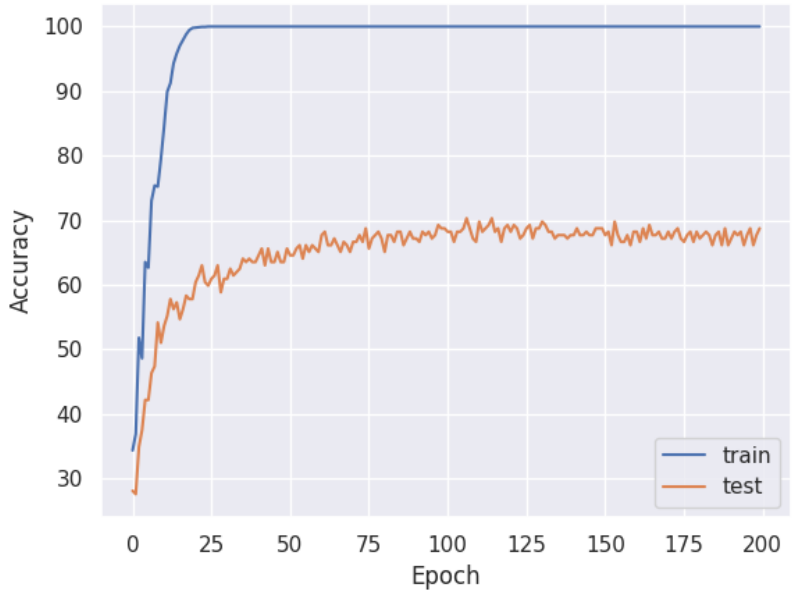


Figure 4.6: Accuracy of the classifier on training and test sets over the epochs.

Figure 4.6 depicts the accuracy of the training and test sets evaluated during the training phase. The model overfits the training dataset due to the small amount of data supplied and properly classifies all of the samples in the training dataset within 20 epochs. On the other hand, accuracy across the test dataset grows slower and finally stabilizes, peaking around epoch 100, when the greatest accuracy is 71%.

In order to inspect how well the single classes are classified and which are the classes that confuse the classification, the confusion matrix evaluated on the test set is reported (figure 4.7).

Figure 4.7: Confusion matrix of the classifier on the test set

By analyzing the matrix, it is clear that the sad pieces are identified better than the others, even if they are mistaken with the relaxed ones in 16% of the cases. The happy pieces are confused with the relaxed and angry ones. Furthermore, the angry pieces are confused with the happy and relaxed one with a percentage of 12%.

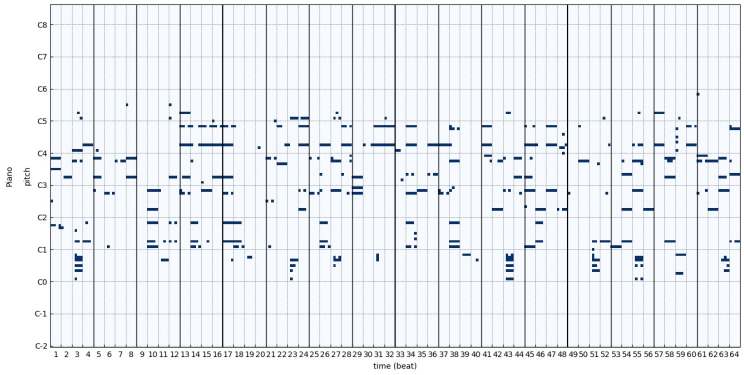
4.3 Controlling the Generation

In this section, the results related to the experiments described in sect. 3.4 are reported.

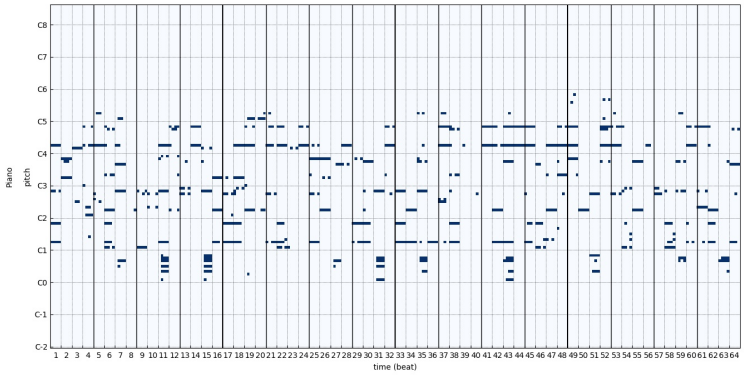
4.3.1 Transfer Learning

The approach of transfer learning, described in section 3.4.1, failed in successfully generating mood conditioned pieces.

Even if the experiment relies on a solid theoretical background, the big limitation is given by the amount of labelled data. Indeed, the labelled data contains to few samples and the model is not able to learn a consistent representation for each of the mood classes.



(a) Sample generated to be happy



(b) Sample generated to be relaxed

Figure 4.8: Comparison of conditioned samples

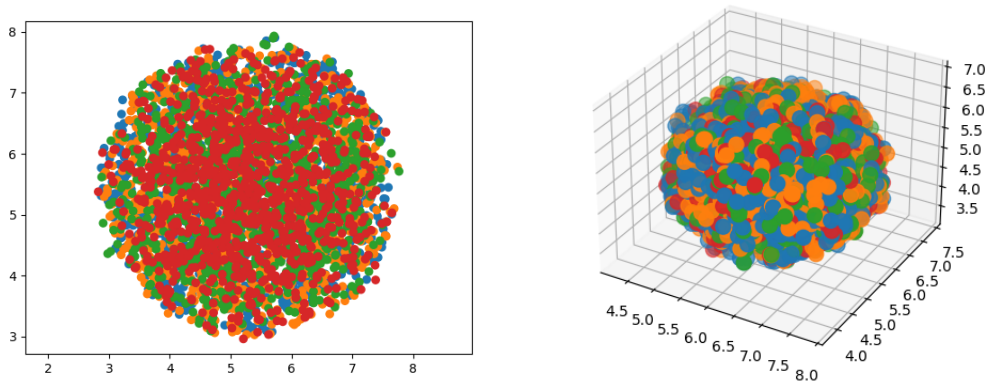
These conclusions are derived by visually inspecting the obtained results and by listening to the generated samples. Changing the class on which the generation is conditioned, it appears that the generated samples undergo a random variation and that the class given as input does not correlate with the obtained output. Moreover, the quality of the samples is affected negatively by this approach and decreases. Indeed, the generated pianorolls appear very noisy, with a lot of random notes, thus not preserving the musicality.

Figure 4.8 represents two pianorolls generated starting from the same latent vector except for the conditioning part that is appended, which is related to different classes (happy and relaxed). It is straightforward to notice that the variations from one sample to the other does not correspond to the class information. Thus, this method cannot accomplish the desired goal mainly due to the peculiarity of the dataset.

4.3.2 Latent Space Exploration

In this section, the results related to the second experiment (sect. 3.4.2) are discussed.

This method does not succeed in generating emotional conditioned pieces. Controllable generation works well only if the features we want to control are disentangled between each other. This means that each dimension in the latent space is correlated with only one feature, so that if the movement is performed in that direction, that single feature is controlled. Unfortunately, this is not the case since the features related to the emotion of a piece are really entangled between each other. The entanglement of the mood classes in the latent space has been demonstrated by classifying 4000 generated pieces and plotting the low-dimensional representation of the related input in the latent space.



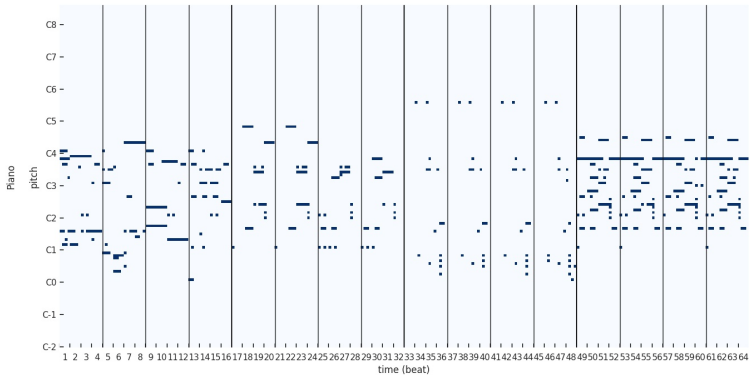
(a) 2 dimensions **(b)** 3 dimensions

Figure 4.9: UMAP projection of mood categories

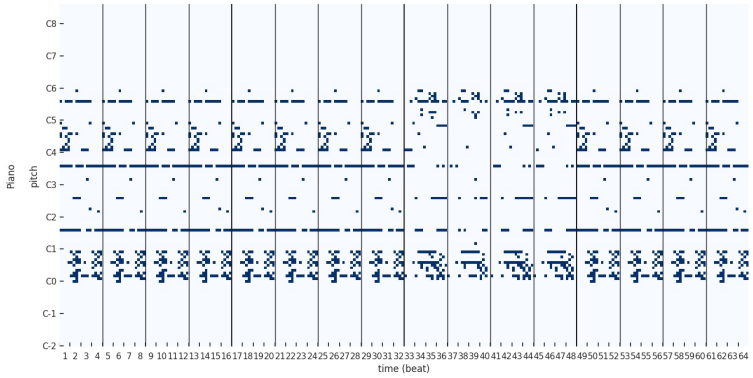
Figure 4.9 has been obtained using the Uniform Manifold Approximation and Projection (UMAP) for dimension reduction that has been proposed in [53]. This method is a novel manifold learning technique for dimension reduction that preserves the global structure. The plots have been obtained by reducing the 128-dimensional input points of the latent space in two or three dimensions. Each color represents one class of emotion.

The different starting points for each mood are scattered around the latent space, and regions of this space that correspond to a specific emotion are not identifiable.

The fact that the features are entangled leads to unsuccessful results. Indeed, the SGA's search for local maxima fails to locate particular areas associated with the desired emotion. The movement in the latent space does not correlate with the features related to mood, and as the number of SGA iterations increases, weird artifacts arise in the produced pianorolls.



(a) SGA iterations = 5



(b) SGA iterations = 30

Figure 4.10: Generated pianoroll moving with SGA maximizing happy feature

Figure 4.10 shows the pianorolls generated starting from a random point in the latent space and using the gradient of the mood classifier with SGA to maximize the amount of happy mood. The changes for each SGA iteration are random and when the number of iterations increases, strange patterns appear as in 4.21b.

4.3.3 Generate and Filter

The third experiment consists in generating unconditioned pieces and filtering the one corresponding to the target mood with the "Mood Classifier".

This method is the one that gives better results in terms of quality of the generated samples and correlation with the wanted mood. These considerations can be draw by visually inspecting the generated samples and listening to the

synthesized audio.

The unconditioned GAN generates melodies and the mood classifier is able to correctly identify the mood related to the generated pieces. In this way the output piece should reflect well the target emotion. However, a quantitative evaluation of this approach is difficult due to the subjective nature of the characteristic that is controlled.

A qualitative evaluation of this method is carried out in section 3.5.2, indeed the results of the survey are fundamental to investigate whether the proposed method effectively reaches the aim of the project.

4.4 Model Evaluation

This section reports the results of the model evaluation performed first by a quantitative evaluation and then through a survey.

4.4.1 Quantitative Metrics

The quantitative metrics explained in section 3.5.1 have been evaluated to select the best unconditioned GAN model. First, the FID evaluated at different epochs is discussed. Then the distribution for each metric of section 2.5.2 evaluated on the best model is shown and commented.

The methodology of section 3.5.1 is applied to select the best performing unconditioned GAN model. The FID scores evaluated every 200 epochs are visualized in Figure 4.11.

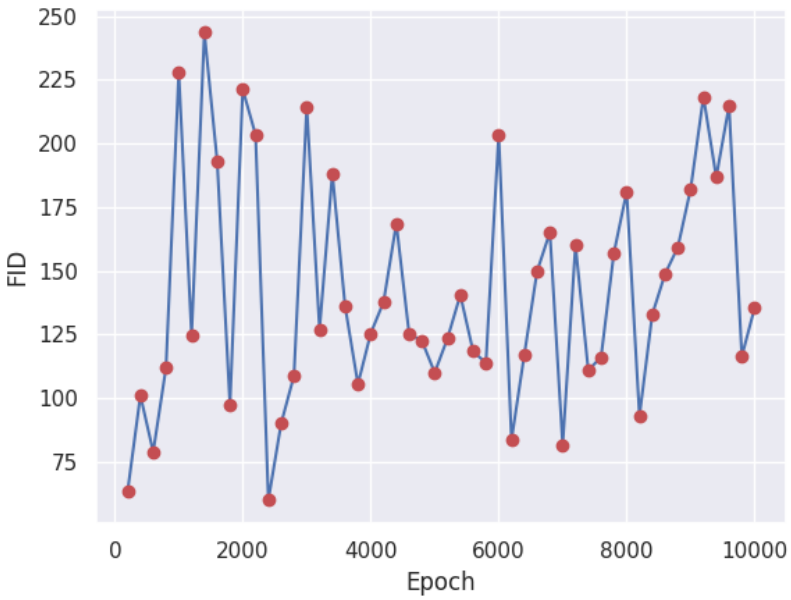
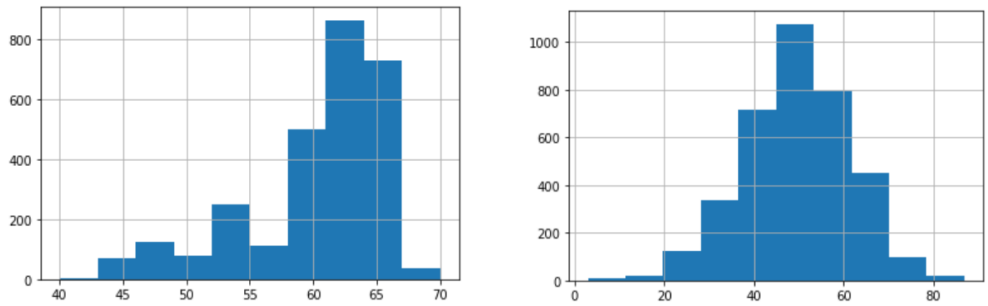


Figure 4.11: FID evaluated over the epochs

It is clear to see that the best performing model, i.e. the one with the lowest FID value, was produced at epoch 2400, and therefore this model was chosen to carry out the experiments.

Furthermore, with the selected model, a fake dataset is generated over which are computed the metrics defined in section 2.5.2. The dataset contains 700 samples per class, with a total of 2800 samples that have been generated using the approach

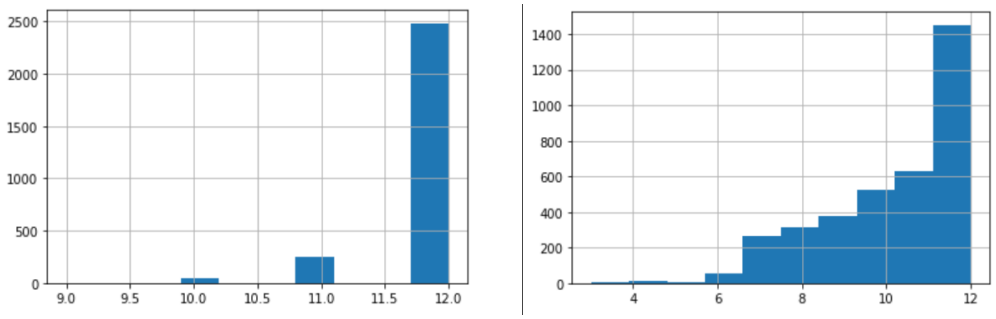
explained in the third experiment (sect. 3.4.3). Here is reported the comparison between the true and generated datasets for the various metrics.



(a) Generated **(b)** True

Figure 4.12: Pitch Range

From the pitch range in Figure 4.12 it is possible to notice that for the generated dataset it spans from 40 to 70, while in the true dataset it spans from 5 to 90. Furthermore, the mean of the pitch range is 60 for the generated dataset and 50 for the true one. The pitch range is correlated with the number of unique pitch classes used, indeed in Figure 4.13 is noticeable that the real dataset has a bigger variety of pitch classes for the different songs, while in the fake dataset the majority of the samples contain 12 pitch classes.

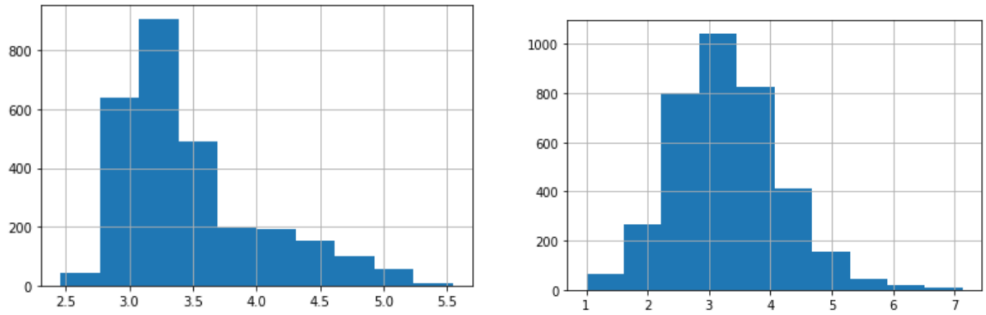


(a) Generated **(b)** True

Figure 4.13: Pitch classes used

Figure 4.14 depicts the polyphony; also in this case, the histogram of the generated samples ranges from 2.5 to 5.5, whereas the histogram of the real samples ranges from 1 to 7. It is reasonable to claim that real samples contain greater diversity than synthetic samples. On the other hand, the mean polyphony for

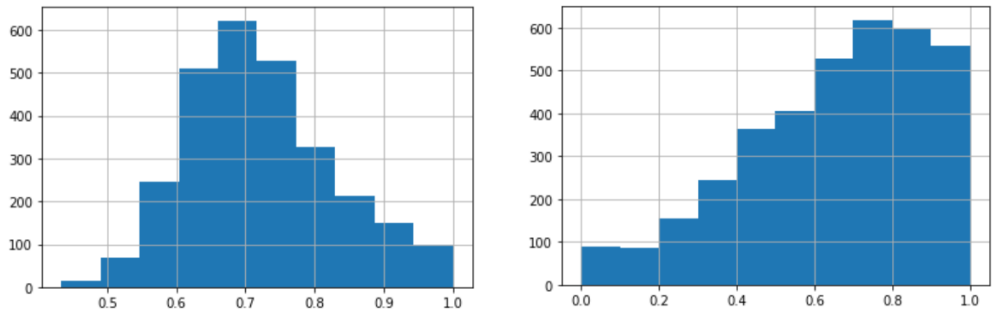
the two datasets is almost identical (about 3.2), indicating that on average the same number of pitches are played simultaneously for both the true and generated samples.



(a) Generated **(b)** True

Figure 4.14: Polyphony

The polyphony rate in Figure 4.15 represents the ratio between the number of timesteps in which multiple pitches are active and the total number of timesteps. By analyzing the histograms it is possible to see that the true dataset has again more variety in terms of the range of values. The mean of the produced samples is 0.7, whereas the mean of the real samples is 0.6; moreover, the minimum value for the true dataset is zero, indicating that there are certain songs in which only single notes are played. This, however, does not occur in the synthetic dataset, where the minimum value is 0.43, implying that each song in the generated dataset uses several notes played simultaneously at least 40% of the time.

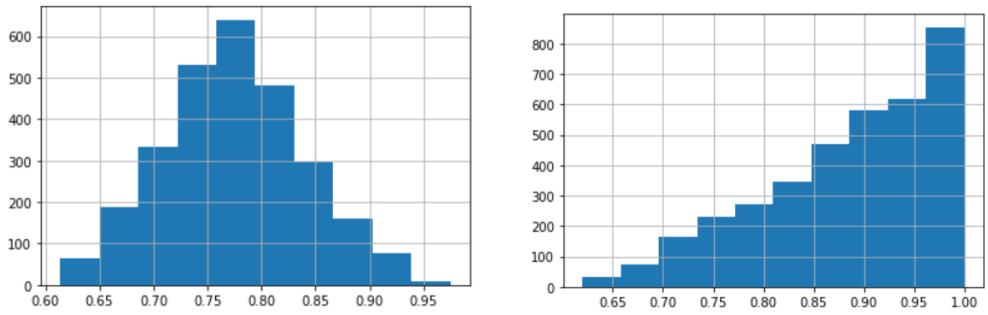


(a) Generated **(b)** True

Figure 4.15: Polyphony Rate

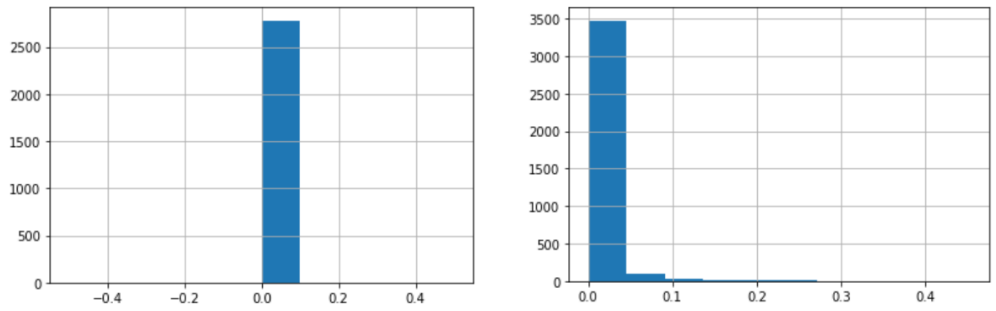
The scale consistency, shown in Figure 4.16, is an indicator of the fraction of

tones that are part of a standard scale, considering the best matching one. The real dataset has an average scale consistency of 0.88, which is greater than the generated dataset's average of 0.76. This discrepancy exists because the GAN did not learn how to produce scales; in fact, the idea of scale is not even considered throughout the generation. However, the achieved value of scale consistency is not too low, given that there are several songs in the original dataset with scale consistency in the range [0.7;0.85].



(a) Generated **(b)** True

Figure 4.16: Scale Consistency



(a) Generated **(b)** True

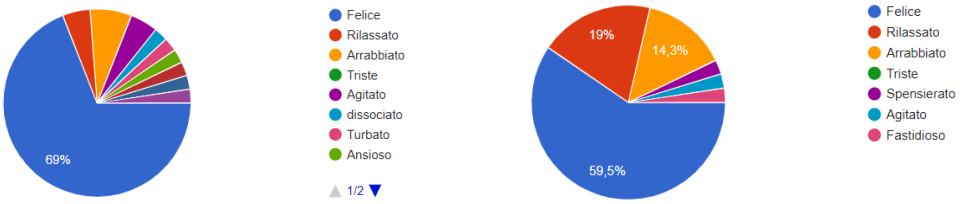
Figure 4.17: Empty Beat Rate

The last metric to be considered is the empty beat rate, which analyzes the ratio of empty beats to total number of beats, indicating the amount of "silence" in a song. The two histograms are quite similar, as seen in Figure 4.17. Indeed, the majority of songs have an empty beat rate between 0 and 0.1, despite the fact that a minority of pieces in the real collection have greater values.

4.4.2 Survey Results

The results of the survey presented in Section 3.5.2 are reported and discussed. A pie chart displaying the proportion of answers for each of the eight questions is provided. In addition, several statistics are generated by aggregating the findings. Because the results are displayed in Italian, a color legend is also provided:

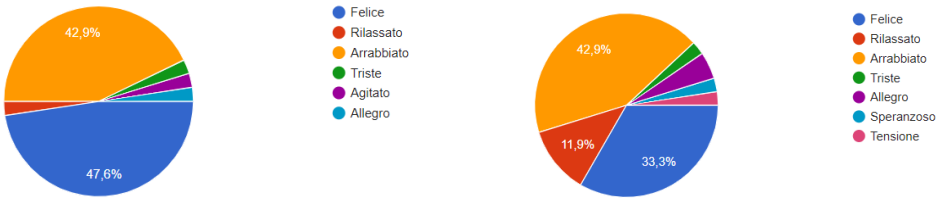
- Blue : Happy
- Red: Relaxed
- Orange: Angry
- Green : Sad



(a) Happy sample 1 **(b)** Happy sample 2

Figure 4.18: Answers related to "happy" samples

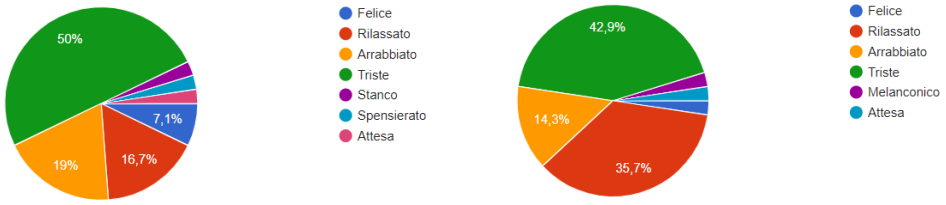
In Figure 4.18 are presented the responses linked to the happy pieces, which have been identified by an average of 65% of participants. The happy class outperformed the other classes, and because the mood classifier employed in the generation process has an accuracy of around 70%, the survey results are perfectly coherent with the expectations.



(a) Angry sample 1 **(b)** Angry sample 2

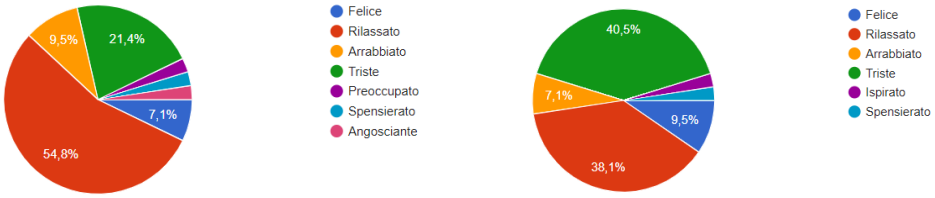
Figure 4.19: Answers related to "angry" samples

Figure 4.19 shows the results related to the angry pieces, about the 40 % of the participants recognized these pieces as actually angry ones. In addition, a large percentage misclassified angry pieces as happy ones.



(a) Relaxed sample 1 **(b)** Relaxed sample 2

Figure 4.20: Answers related to "relaxed" samples



(a) Sad sample 1 **(b)** Sad sample 2

Figure 4.21: Answers related to "sad" samples

Finally, Figure 4.20 and Figure 4.21 report the answers related to relaxed and sad samples respectively: it is possible to notice that relaxed samples are often confused with sad ones and vice versa.

Figure 4.22 reports the quality of the generated pieces in a 1 to 5 scale, as judged by the participants. The distribution of the scores almost resembles a normal distribution with mean 3. Thus, the overall quality was judged as neither exceptional neither too bad and the outcome of the work can be considered sufficient.

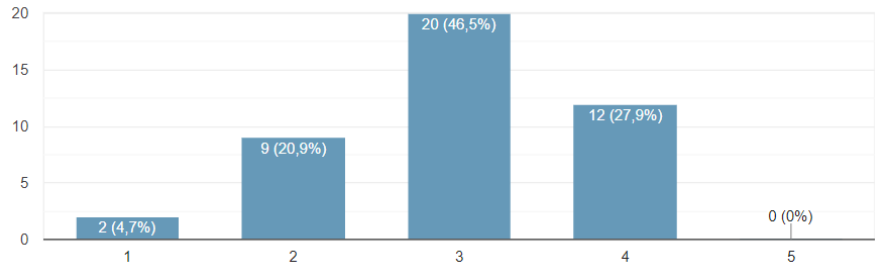


Figure 4.22: Quality of generated pieces from 1 to 5

Chapter 5

Conclusions

Dealing with the issues of affective music composition proved to be a difficult task. Various attempts towards this goal have been documented in the literature, with different degrees of success. The research in this field is constantly evolving, and there are several degrees of freedom in the design phase, such as the algorithm to use and the music representation to choose. This work represents an attempt to investigate GAN capabilities for this type of application.

The poor availability of labelled data is the bottleneck of this research, posing a significant constraint on the generation process. Indeed, GANs are "data hungry" models that require a large amount of data to be successfully trained. To address this limitation, the most basic way may be to collect more labeled data using various methodologies and tunes played by various instruments. Furthermore, one intriguing avenue to go is to annotate the pieces using functional magnetic resonance imaging (fMRI), which assesses the active regions of the brain by looking at blood flow. In fact, different emotions are thought to stimulate different regions of the brain, which might be an useful approach to evaluate emotions more objectively, indeed the human annotation of the pieces introduces a bias in the process. However, the data labeling procedure is typically time and resource intensive. Another interesting direction could be to study optimization of GAN training with limited data, which would need a more sophisticated architecture, as seen in [54]. Other approaches have been proposed, such as using a variational autoencoder to direct the generation process towards certain features [55].

The first and second experiments of this project are valid from a methodological point of view, indeed the application of such techniques has demonstrated to be successful in various scenarios, however due to the intrinsic nature of the task addressed their application failed in this case. A future improvement on the latent space exploitation could be given by modifying the training process, encouraging the disentanglement between the mood features. Similarly in [56] the authors propose a supervised method to explicitly map the latent space into a meaningfully

pre-defined semantic space.

The third experiment, which employs the mood classifier in a loop with the generator, produces the best results. This is due to the fact that its technique is best suited to this unique circumstance. Indeed, the success of a certain technique is inextricably linked to the characteristics of the situation in which it is used. One contribution given by this research is to demonstrate that a CNN based classifier is able to reach similar performance to the LSTM based mood classifier presented in [18], given that the two classifiers have been trained on the same dataset. Moreover, the unconditioned GAN model has proved to being able to learn and generate different moods, indeed all four classes are detected by the classifier.

The survey demonstrates that the proposed algorithm is able to generate recognizable happy pieces, since the participants classify them with the same accuracy of the mood classifier used. Regarding the other classes, the results are worse: angry pieces are sometimes confused with happy pieces, and the sad melodies are often confused with relaxed ones and vice versa.

The conclusion of this investigation demonstrates certain limits imposed by the very subjective nature of emotions. Even if the classifier is quite excellent at recognizing distinct emotions, the same piece triggers diverse feelings in the survey participants. As a result, the same tune may convey a variety of emotions depending on who is listening. Everyone may have a different interpretation of a musical piece, and it is quite difficult to say which one is the most correct.

It would be extremely fascinating to apply the experiments presented in this master's thesis to other emotion datasets in a variety of settings. This work is focused on symbolic music generation, dealing only with notes, however, the raw audio generation could be also explored. Furthermore, the lyric is crucial in establishing the emotional content of a song, and hybrid methods capable of processing words and music may produce songs with a certain mood. The potential of automatic music generation is enormous, and the the increased availability of data will boost the effectiveness of these types of algorithms even more.

Bibliography

[1] Hao-Wen Dong, Wen-Yi Hsiao, Li-Chia Yang, and Yi-Hsuan Yang. *MuseGAN: Multi-track Sequential Generative Adversarial Networks for Symbolic Music Generation and Accompaniment*. 2017. arXiv: 1709.06298 [eess.AS] (cit. on pp. 3, 5, 6, 19, 20, 22, 26, 30).

[2] Jian Wu, Changran Hu, Yulong Wang, Xiaolin Hu, and Jun Zhu. *A Hierarchical Recurrent Neural Network for Symbolic Melody Generation*. 2018. arXiv: 1712.05274 [cs.SD] (cit. on p. 4).

[3] Jean-Pierre Briot, Gaëtan Hadjeres, and François-David Pachet. *Deep Learning Techniques for Music Generation – A Survey*. 2019. arXiv: 1709.01620 [cs.SD] (cit. on pp. 4, 6).

[4] Hélio de Oliveira and Raimundo Oliveira. «Understanding MIDI: A Painless Tutorial on Midi Format». In: (May 2017) (cit. on p. 5).

[5] Allen Huang and Raymond Wu. *Deep Learning for Music*. 2016. arXiv: 1606.04930 [cs.LG] (cit. on p. 5).

[6] *MIDI Manufacturers Association et al. Midi specifications*. https://www.midi.org/specifications/. Accessed: 2021-20-06 (cit. on p. 5).

[7] Hao-Wen Dong, Wen-Yi Hsiao, Li-Chia Yang, and Yi-Hsuan Yang. *Pypianoroll: Open source python package for handling multitrack pianorolls*. 2017 (cit. on pp. 6, 25).

[8] Christian Walder. *Symbolic Music Data Version 1.0*. 2016. arXiv: 1606.02542 [cs.SD] (cit. on p. 6).

[9] Youngmoo Kim, Erik Schmidt, Raymond Migneco, Brandon Morton, Patrick Richardson, Jeffrey Scott, Jacquelin Speck, and Douglas Turnbull. «Music emotion recognition: A state of the art review». In: *Proceedings of the 11th International Society for Music Information Retrieval Conference, ISMIR 2010* (Jan. 2010) (cit. on p. 8).

[10] Yi-hsuan Yang, Yu-Ching Lin, Ya-Fan Su, and Homer Chen. «A Regression Approach to Music Emotion Recognition». In: *Audio, Speech, and Language Processing, IEEE Transactions on* 16 (Mar. 2008), pp. 448–457. DOI: 10.1109/TASL.2007.911513 (cit. on p. 8).

[11] Xin Liu, Qingcai Chen, Xiangping Wu, Yan Liu, and Yang Liu. *CNN based music emotion classification*. 2017. arXiv: 1704.05665 [cs.MM] (cit. on p. 8).

[12] Serhat Hizlisoy, Serdar Yildirim, and Zekeriya Tufekci. «Music emotion recognition using convolutional long short term memory deep neural networks». In: *Engineering Science and Technology, an International Journal* 24.3 (2021), pp. 760–767. ISSN: 2215-0986. DOI: https://doi.org/10.1016/j.jestch.2020.10.009. URL: https://www.sciencedirect.com/science/article/pii/S2215098620342385 (cit. on p. 8).

[13] Renato Panda, Ricardo Malheiro, and Rui Pedro Paiva. «Audio Features for Music Emotion Recognition: a Survey». In: *IEEE Transactions on Affective Computing* PP (Oct. 2020), pp. 1–1. DOI: 10.1109/TAFFC.2020.3032373 (cit. on p. 8).

[14] Duncan Williams, Alexis Kirke, Eduardo R Miranda, Etienne Roesch, Ian Daly, and Slawomir Nasuto. «Investigating affect in algorithmic composition systems». In: *Psychology of Music* 43.6 (2015), pp. 831–854. DOI: 10.1177/0305735614543282. eprint: https://doi.org/10.1177/0305735614543282. URL: https://doi.org/10.1177/0305735614543282 (cit. on p. 8).

[15] Marco Scirea, Julian Togelius, Peter Eklund, and Sebastian Risi. «Affective evolutionary music composition with MetaCompose». In: *Genetic Programming and Evolvable Machines* 18 (Dec. 2017), pp. 1–33. DOI: 10.1007/s10710-017-9307-y (cit. on p. 8).

[16] R. Madhok, Shivali Goel, and Shweta Garg. «SentiMozart: Music Generation based on Emotions». In: *ICAART*. 2018 (cit. on p. 8).

[17] Sanidhya Mangal, Rahul Modak, and Poorva Joshi. «LSTM Based Music Generation System». In: *IARJSET* 6.5 (May 2019), pp. 47–54. ISSN: 2393-8021. DOI: 10.17148/iarjset.2019.6508. URL: http://dx.doi.org/10.17148/IARJSET.2019.6508 (cit. on p. 8).

[18] Lucas N. Ferreira and Jim Whitehead. *Learning to Generate Music With Sentiment*. 2021. arXiv: 2103.06125 [cs.LG] (cit. on pp. 8, 10, 23, 59).

[19] Erion Çano and M. Morisio. «Music Mood Dataset Creation Based on Last.fm Tags». In: 2017 (cit. on pp. 9, 24).

[20] Kate Hevner. «Experimental Studies of the Elements of Expression in Music». In: *The American Journal of Psychology* 48.2 (1936), pp. 246–268. ISSN: 00029556. URL: http://www.jstor.org/stable/1415746 (cit. on p. 9).

[21] James Russell. «A Circumplex Model of Affect». In: *Journal of Personality and Social Psychology* 39 (Dec. 1980), pp. 1161–1178. DOI: 10.1037/h0077714 (cit. on pp. 9, 10).

[22] 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. 11).

[23] Andrew Brock, Jeff Donahue, and Karen Simonyan. *Large Scale GAN Training for High Fidelity Natural Image Synthesis*. 2019. arXiv: 1809.11096 [cs.LG] (cit. on p. 11).

[24] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. *Analyzing and Improving the Image Quality of StyleGAN*. 2020. arXiv: 1912.04958 [cs.CV] (cit. on p. 11).

[25] Aidan Clark, Jeff Donahue, and Karen Simonyan. *Adversarial Video Generation on Complex Datasets*. 2019. arXiv: 1907.06571 [cs.CV] (cit. on p. 11).

[26] Ian Goodfellow. *NIPS 2016 Tutorial: Generative Adversarial Networks*. 2017. arXiv: 1701.00160 [cs.LG] (cit. on p. 11).

[27] *Google - Generative Adversarial Networks*. https://developers.google.com/machine-learning/gan/gan_structure. Accessed: 2021-03-06 (cit. on p. 11).

[28] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. *Improved Techniques for Training GANs*. 2016. arXiv: 1606.03498 [cs.LG] (cit. on p. 12).

[29] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. *Progressive Growing of GANs for Improved Quality, Stability, and Variation*. 2018. arXiv: 1710.10196 [cs.NE] (cit. on pp. 12, 18).

[30] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. *Improved Training of Wasserstein GANs*. 2017. arXiv: 1704.00028 [cs.LG] (cit. on pp. 12, 15, 16, 32).

[31] Alec Radford, Luke Metz, and Soumith Chintala. *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*. 2016. arXiv: 1511.06434 [cs.LG] (cit. on pp. 13, 26, 32).

[32] Sergey Ioffe and Christian Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. 2015. arXiv: 1502.03167 [cs.LG] (cit. on p. 13).

[33] Vinod Nair and Geoffrey E. Hinton. «Rectified Linear Units Improve Restricted Boltzmann Machines». In: *Proceedings of the 27th International Conference on International Conference on Machine Learning*. ICML'10. Haifa, Israel: Omnipress, 2010, pp. 807–814. ISBN: 9781605589077 (cit. on p. 13).

[34] Andrew L. Maas. «Rectifier Nonlinearities Improve Neural Network Acoustic Models». In: 2013 (cit. on p. 13).

[35] Martin Arjovsky, Soumith Chintala, and Léon Bottou. *Wasserstein GAN*. 2017. arXiv: 1701.07875 [stat.ML] (cit. on pp. 14, 15).

[36] Y. Rubner, C. Tomasi, and L.J. Guibas. «A metric for distributions with applications to image databases». In: *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*. 1998, pp. 59–66. DOI: 10.1109/ICCV.1998.710701 (cit. on p. 14).

[37] Lilian Weng. *From GAN to WGAN*. 2019. arXiv: 1904.08994 [cs.LG] (cit. on p. 14).

[38] Mehdi Mirza and Simon Osindero. *Conditional Generative Adversarial Nets*. 2014. arXiv: 1411.1784 [cs.LG] (cit. on pp. 16–18).

[39] Hanock Kwak and Byoung-Tak Zhang. *Ways of Conditioning Generative Adversarial Networks*. 2016. arXiv: 1611.01455 [cs.LG] (cit. on p. 17).

[40] Augustus Odena, Christopher Olah, and Jonathon Shlens. *Conditional Image Synthesis With Auxiliary Classifier GANs*. 2017. arXiv: 1610.09585 [stat.ML] (cit. on p. 17).

[41] Minhyeok Lee and Junhee Seok. *Controllable Generative Adversarial Network*. 2019. arXiv: 1708.00598 [cs.LG] (cit. on p. 17).

[42] Logan Eisenbeiser. «Latent Walking Techniques for Conditioning GAN-Generated Music». In: *2020 11th IEEE Annual Ubiquitous Computing, Electronics Mobile Communication Conference (UEMCON)*. 2020, pp. 0548–0553. DOI: 10.1109/UEMCON51285.2020.9298149 (cit. on p. 18).

[43] Yujun Shen, Jinjin Gu, Xiaoou Tang, and Bolei Zhou. *Interpreting the Latent Space of GANs for Semantic Face Editing*. 2020. arXiv: 1907.10786 [cs.CV] (cit. on p. 18).

[44] Peiye Zhuang, Oluwasanmi O Koyejo, and Alex Schwing. «Enjoy Your Editing: Controllable {GAN}s for Image Editing via Latent Space Navigation». In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=HOFxeCutxZR (cit. on p. 18).

[45] Jesse Engel, Kumar Krishna Agrawal, Shuo Chen, Ishaan Gulrajani, Chris Donahue, and Adam Roberts. *GANSynth: Adversarial Neural Audio Synthesis*. 2019. arXiv: 1902.08710 [cs.SD] (cit. on pp. 18, 19, 34).

[46] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu¦ *WaveNet: A Generative Model for Raw Audio*. 2016. arXiv: 1609.03499 [cs.SD] (cit. on p. 18).

[47] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. *GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium*. 2018. arXiv: 1706.08500 [cs.LG] (cit. on pp. 21, 32).

[48] Hao-Wen Dong, Ke Chen, Julian McAuley, and Taylor Berg-Kirkpatrick. *MusPy: A Toolkit for Symbolic Music Generation*. 2020. arXiv: 2008.01951 [cs.SD] (cit. on p. 21).

[49] Olof Mogren. *C-RNN-GAN: Continuous recurrent neural networks with adversarial training*. 2016. arXiv: 1611.09904 [cs.AI] (cit. on p. 22).

[50] Casper Kaae Sønderby, Jose Caballero, Lucas Theis, Wenzhe Shi, and Ferenc Huszár. *Amortised MAP Inference for Image Super-resolution*. 2017. arXiv: 1610.04490 [cs.CV] (cit. on p. 32).

[51] *BCE with logits - Pytorch documentation*. https://pytorch.org/docs/stable/generated/torch.nn.BCEWithLogitsLoss.html. Accessed: 2021-17-06 (cit. on p. 33).

[52] *Sharon Zhou, Eda Zhou, Eric Zelikman, Build Basic Generative Adversarial Networks, Coursera [MOOC]*. https://www.coursera.org/learn/build-basic-generative-adversarial-networks-gans. Accessed: 2021-17-04 (cit. on p. 35).

[53] Leland McInnes, John Healy, and James Melville. *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction*. 2020. arXiv: 1802.03426 [stat.ML] (cit. on p. 48).

[54] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. *Training Generative Adversarial Networks with Limited Data*. 2020. arXiv: 2006.06676 [cs.CV] (cit. on p. 58).

[55] Manel Mateos, Alejandro González, and Xavier Sevillano. *Guiding GANs: How to control non-conditional pre-trained GANs for conditional image generation*. 2021. arXiv: 2101.00990 [cs.CV] (cit. on p. 58).

[56] Toan Pham Van, Tam Minh Nguyen, Ngoc N. Tran, Hoai Viet Nguyen, Linh Bao Doan, Huy Quang Dao, and Thanh Ta Minh. «Interpreting the Latent Space of Generative Adversarial Networks using Supervised Learning». In: *2020 International Conference on Advanced Computing and Applications (ACOMP)* (Nov. 2020). DOI: 10.1109/acomp50827.2020.00015. URL: http://dx.doi.org/10.1109/ACOMP50827.2020.00015 (cit. on p. 58).