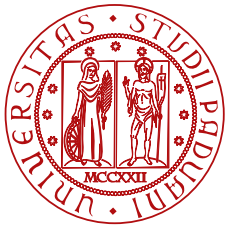

Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Master Thesis in Computer Engineering

An interesting title for the thesis

Master Candidate

Galli Filippo

Student ID 2120826

Supervisor

Prof. Antonio Rodà

University of Padova

Co-supervisor

Dott. Matteo Spanio

University of Padova

Academic Year
2024/2025

To my parents
and friends

Abstract

ABSTRACT IN INGLESE

L'avvento del deep learning ha rivoluzionato il campo della generazione audio e musicale, offrendo numerosi strumenti per creare suoni espressivi e realistici. Gli Autoencoder Variazionali (VAE), e in seguito gli Autoencoder Variazionali Condizionali (CVAE), sono emersi come una classe particolarmente significativa di modelli generativi in questo ambito, capaci di apprendere distribuzioni di dati complesse e generare nuovi campioni. Tuttavia questi modelli sono spesso considerati delle "scatole nere", in cui gli attributi che contribuiscono esplicitamente alla sintesi, codificati nello spazio latente, non sono facilmente interpretabili. Sebbene la ricerca esistente abbia esplorato come i dati di input influenzino lo spazio latente e la successiva generazione, l'interpretabilità e la controllabilità di questo spazio latente rimangono sfide fondamentali, in particolare nel contesto della musica.

Questo lavoro mira ad approfondire lo spazio latente, nello specifico quello degli Autoencoder Variazionali Condizionali (CVAE), per la generazione di timbri di strumenti musicali. L'obiettivo principale è sviluppare un CVAE che consenta un controllo maggiore sugli attributi musicalmente rilevanti, permettendo un processo di generazione musicale più interpretabile e controllabile. Condizionando la generazione su variabili latenti che rappresentano caratteristiche emotive specifiche, l'obiettivo è quello di comprendere come ogni attributo contribuisca al suono finale.

Nello specifico, l'idea è di concentrarsi sugli spazi latenti condizionati, cercando di capire come i diversi mood emotivi influenzano lo spazio latente e come il modello impara a separare le caratteristiche musicali in relazione a queste condizioni.

Sommario

Contents

List of Figures

List of Tables

List of Algorithms

List of Code Snippets

List of Acronyms

CSV Comma Separated Values

1 Introduction

Random citation [1].
Random footnote.[1]

1.1 Background and Motivation

Contesto Generale:

La musica rappresenta da sempre uno dei più potenti mezzi di espressione e comunicazione umana, un linguaggio universale in grado di evocare e modificare lo stato emotivo dell'ascoltatore. Negli ultimi anni, i significativi progressi nel campo dell'intelligenza artificiale e, in particolare, del Deep Learning, hanno aperto nuove frontiere nella creazione musicale. I sistemi di IA non sono più solo strumenti per analizzare o classificare la musica, ma sono diventati essi stessi dei creatori, capaci di comporre brani unici e originali. Questo ha generato un crescente interesse in molteplici settori: dall'intrattenimento, dove ad esempio colonne sonore per videogiochi e film possono essere generate dinamicamente per adattarsi all'azione [FONTI], all'ambito terapeutico, in cui la musica personalizzata può essere impiegata per la gestione dello stress e il miglioramento del benessere psicofisico [FONTI]. Oltre a ciò, questi strumenti possono rappresentare una risorsa di grande valore per la creatività, offrendo a compositori e artisti nuove fonti di ispirazione, permettendo allo stesso tempo a persone

[1] `https://lucamartinelli.eu.org`

comuni di comporre proprie opere artistiche, senza la necessità di specifiche abilità tecniche nella composizione musicale.

Problema Specifico:

All'interno di questo vastissimo campo, emerge la generazione di musica affettiva. La vera sfida, in questo contesto, non risiede solamente nel generare melodie tecnicamente corrette o armonicamente gradevoli, ma nel creare composizioni che siano cariche di un preciso intento emotivo che sia controllabile. Sebbene i modelli generativi attuali abbiano raggiunto ottimi risultati, permane un "gap" tra la capacità di produrre musica e quella di plasmarne finemente il contenuto espressivo. Il problema è duplice. Da un lato, vi è la difficoltà nell'ottenere un controllo granulare sulle sfumature emotive. Molti sistemi infatti si limitano a etichette generiche e discrete come "felice" o "triste", mentre la vera espressività risiede nella capacità di navigare lo spazio emotivo in modo continuo e dettagliato (ad esempio, modulando l'intensità o la valenza di un'emozione). Dall'altro lato, emerge la questione dell'interpretabilità: spesso questi sistemi operano come "scatole nere" (black box), rendendo difficile comprendere come le architetture di Deep Learning traducano un'intenzione emotiva in specifiche scelte musicali, quali tempo, modalità, progressione armonica o dinamica. Questa tesi si inserisce proprio in questo "gap", con l'obiettivo di esplorare e implementare un'architettura per la generazione di musica affettiva, nell specifico di un Conditional Autoencoder, concentrandosi sulle sfide legate alla generazione della musica stessa, al suo controllo affective e all'analisi dello stato latente del modello.

Soluzione Proposta (CVAE):

Introdurre i CVAE come una promettente direzione per affrontare queste sfide, grazie alla loro capacità di apprendere rappresentazioni latenti e di essere condizionati. (VALUTARE SE LASCIARE O PARLARNE DIRETTAMENTE NEL PROSSIMO CAPITOLO)

Research Question (Domanda di Ricerca):

Formula una o due domande di ricerca chiare e concise. **Esempio:** "Come può un Conditional Variational Autoencoder essere addestrato per generare

segmenti musicali audio raw che riflettano specifiche condizioni emotive (es. Valence-Arousal), e come possiamo interpretare il suo spazio latente per comprendere questa mappatura emotiva?"

Obiettivi (Objectives):

L'obiettivo della tesi è quello di implementare un Condidional Autoencoder per la generazione di musica condizionata che utilizzi come input dati audio raw, ovvero analizzare audio grezzi a partire dalla forma d'onda. Trovare i parametri ideali, analizzare la miglior rappresentazione di questi dati in modo da rendere efficace il training, capire miglior strategia di condizionamento e studiare lo spazio latente

obiettivi sotto forma di elenco:

- Progettare e implementare un modello CVAE per la generazione di musica affettiva da audio raw, condizionato su parametri di Valence e Arousal.
- studiare come meglio strutturare/gestire i dati di training in modod da rendere più efficace il training
- studiare miglior strategia di condizionamento (es. valutare se discretizzare valori di V/A continui, che layer ed architettura utilizzare)
- Investigare l'organizzazione dello spazio latente del CVAE in relazione alle condizioni emotive.
- Analizzare l'impatto delle singole dimensioni latenti sulla qualità emotiva e timbrica dell'audio generato.
- Valutare la capacità del modello di generare musica coerente con le emozioni target.
- Progettazione e implementare una proof-of-concept di un modello tramite transformer per la generazione di musica affettiva da MIDI, condizionato su parametri di Valence e Arousal.

1.2 (Research Question and Objectives (XAI EXPLAINABLE AI)

Example of list

- Item 1
- Item 2

Example of acronym

Comma Separated Values (CSV)

Example of enumeration

1. Item 1
2. Item 2

Example of quote

> *Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.*

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

2

Background Teorico e Stato dell'Arte

2.1 Rappresentazioni dell'Audio per Modelli Generativi

La scelta dell'architettura di un modello di deep learning è strettamente legata alla rappresentazione dell'audio che si intende utilizzare come input. Prima di definire l'architettura del modello, è quindi fondamentale selezionare la rappresentazione audio più adatta al compito specifico.

In questo lavoro, ci concentreremo sulla rappresentazione di segmenti musicali mediante audio raw e, secondariamente, tramite formato MIDI.

Di seguito una panoramica delle rappresentazioni audio più utilizzate:

2.1.1 Audio Raw

Nell'approccio basato sull'audio raw, il suono viene rappresentato direttamente attraverso la sua forma d'onda (waveform). Questa è costituita da una sequenza di valori numerici che rappresentano l'ampiezza del segnale audio a istanti temporali discreti, determinati da una specifica frequenza di campionamento (sample rate). Una frequenza di campionamento tipica per applicazioni audio è 44.1 kHz, che corrisponde a 44100 campioni al secondo.

Questo genere di rappresentazione sarà alla base nel nostro modello.

Tuttavia, come discusso in precedenza [o nel capitolo precedente, in base a come strutturo l'introduzione], le forme d'onda audio raw, pur essendo es-

tremamente ricche di informazioni (poiché rappresentano il segnale originale senza perdite significative dovute a trasformazioni), presentano numerose sfide a causa della loro elevata dimensionalità (decine di migliaia di campioni per secondo) e la limitata interpretabilità diretta delle sequenze di campioni, che rendono spesso difficile l'apprendimento efficace da parte dei modelli. Per superare questi limiti, l'audio raw viene spesso trasformato in rappresentazioni più compatte e strutturate, come quelle nel dominio tempo-frequenza.

2.1.2 RAPPRESENTAZIONI TEMPO-FREQUENZA

Tra le più comuni rappresentazioni tempo-frequenza troviamo:

SPETTROGRAMMA

Gli spettrogrammi sono ottenuti applicando la Trasformata di Fourier a Tempo Breve (Short-Time Fourier Transform, STFT) a segmenti sovrapposti della forma d'onda. Essi forniscono una rappresentazione visiva (una matrice o immagine) che mostra come il contenuto in frequenza (l'intensità delle diverse frequenze) del segnale audio varia nel tempo. Gli spettrogrammi sono efficienti dal punto di vista computazionale, catturando caratteristiche sonore cruciali come le frequenze fondamentali, le armoniche e il timbro.

MEL-SPETTROGRAMMA

I Mel-spettrogrammi sono una variante degli spettrogrammi che modifica l'asse delle frequenze utilizzando la scala Mel, in grado di approssimare meglio la percezione uditiva umana (l'orecchio umano è più sensibile alle variazioni di frequenza a basse frequenze che ad alte frequenze). Questo si ottiene applicando un banco di filtri spaziati secondo la scala Mel allo spettrogramma lineare. Questa rappresentazione comprime ulteriormente le informazioni sulle frequenze, enfatizzando le gamme percettivamente più rilevanti per l'uomo. Ciò può portare a un apprendimento e a una generazione più efficienti di suoni musicalmente significativi.

2.1.3 RAPPRESENTAZIONE ARMONICA (BASATA SU NOTE)

Un'altra rappresentazione degna di nota è quella basata sulle componenti armoniche delle note strumentali [9]. Questo tipo di rappresentazione, spesso

monofonica (cioè applicabile a una sola nota o melodia alla volta), si concentra sulla cattura della frequenza fondamentale (f_0) e delle ampiezze delle prime N armoniche (ad esempio, le prime 7 come nell'articolo citato). L'obiettivo è descrivere i timbri musicali in uno spazio dimensionalmente ridotto e musicalmente informativo, separando l'informazione timbrica da quella dell'altezza.

2.1.4 Rappresentazione Simbolica: MIDI

Infine, è importante menzionare le rappresentazioni simboliche, di cui il formato MIDI (Musical Instrument Digital Interface) è l'esempio più noto. A differenza delle rappresentazioni precedenti, che descrivono il segnale sonoro effettivo, il MIDI non codifica la forma d'onda, ma è invece un protocollo standard che descrive eventi musicali: quali note suonare (altezza), quando iniziano e finiscono (timing, durata), con quale intensità (velocity), e potenzialmente altre informazioni di controllo come il pitch bend, il vibrato, o la scelta dello strumento. È un formato estremamente compatto e strutturato, ampiamente utilizzato nella produzione musicale, nell'analisi musicologica e come input/output per alcuni modelli generativi.

Anche questo formato audio ci sarà molto utile per la generazione affective di musica.

2.2 Autoencoders e Varianti

Come detto in precedenza questa tesi si concentrerà principalmente sull'utilizzo degli Autoencoder Variazionali Condizionali (CVAE) per la generazione musicale controllabile, per poi passare, secondariamente, all'analisi dell'architettura dei Transformer. Per fornire il contesto necessario a questa scelta, di seguito viene presentata una panoramica dei modelli fondamentali, partendo dagli Autoencoder (AE) di base, evolvendo verso gli Autoencoder Variazionali (VAE) e infine introducendo i CVAE e i transformer.

2.2.1 Autoencoder (AE)

Gli Autoencoder [6] sono reti neurali non supervisionate progettate principalmente per apprendere rappresentazioni compresse dei dati. L'obiettivo fondamentale è catturare le caratteristiche più salienti dell'input in uno spazio

a dimensionalità ridotta, in modo tale da poter poi ricostruire l'input originale con la maggior accuratezza possibile a partire da questa rappresentazione compressa.

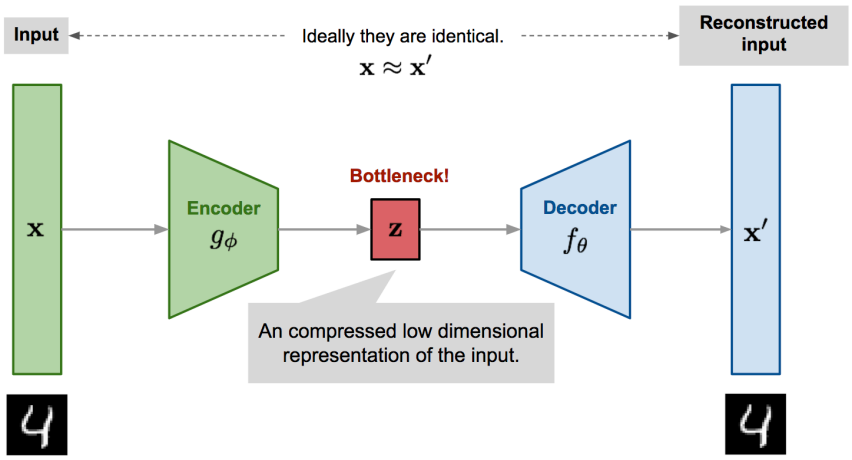

Figure 2.1: Autoencoder architecture [11]

Un Autoencoder è tipicamente costituito da due componenti principali:

- **Encoder:** Questa parte della rete mappa i dati di input x, appartenenti a uno spazio ad alta dimensionalità $\mathbb{R}^d$, in una rappresentazione latente z, appartenente a uno spazio a dimensionalità inferiore $\mathbb{R}^e$, dove $e < d$. Matematicamente, $z = f(x)$, dove f è la funzione implementata dall'encoder.
- **Decoder:** Questa parte della rete prende la rappresentazione latente z e tenta di ricostruire l'input originale x. La ricostruzione $\hat{x}$ (appartenente allo stesso spazio $\mathbb{R}^d$ dell'input) è ottenuta applicando la funzione implementata dal decoder: $\hat{x} = g(z)$. Il modello viene addestrato end-to-end minimizzando una funzione di costo (loss function) che misura la discrepanza tra l'input originale x e la sua ricostruzione $\hat{x}$.

Nonostante la loro efficacia nella riduzione della dimensionalità e nell'estrazione di caratteristiche, gli autoencoder standard presentano limitazioni per i compiti generativi. Lo spazio latente appreso spesso infatti non è né denso né continuo: regioni vuote possono separare le codifiche di campioni diversi, e l'interpolazione tra due punti z_1 e z_2 nello spazio latente non produce necessariamente un output $\hat{x}$ realistico o significativo. Campioni di input simili potrebbero inoltre essere mappati in regioni differenti dello spazio latente, ostacolando la generazione controllata e la sintesi di nuovi campioni variati.

Per superare queste limitazioni, sono stati sviluppati i VAE.

2.2.2 Autoencoder Variazionali (VAE)

Gli Autoencoder Variazionali (VAE) [7] affrontano le limitazioni degli AE standard introducendo un approccio probabilistico sia nella codifica che nella decodifica, basato sui principi dell'inferenza variazionale.

Architettura e Flusso di Lavoro

Invece di mappare un input x a un singolo punto deterministico z nello spazio latente, l'**encoder** di un VAE (parametrizzato da ϕ) mappa l'input x ai parametri di una distribuzione di probabilità sullo spazio latente. Comunemente, si assume che questa distribuzione sia una Gaussiana multivariata con indipendenza tra le componenti:

$$q_\phi(z|x) = \mathcal{N}(z|\mu_\phi(x), \text{diag}(\sigma^2_\phi(x))) \tag{2.1}$$

L'encoder impara quindi a calcolare il vettore delle medie $\mu_\phi(x)$ e il vettore delle varianze $\sigma^2_\phi(x)$ per un dato input x. La variabile latente z viene poi campionata da questa distribuzione: $z \sim q_\phi(z|x)$. Questa distribuzione $q_\phi(z|x)$ funge da approssimazione alla vera ma intrattabile distribuzione a posteriori $p(z|x)$.

Si definisce anche una **distribuzione a priori** sullo spazio latente, $p(z)$, solitamente scelta come una Gaussiana standard, $p(z) = \mathcal{N}(0, I)$, dove I è la matrice identità. Il **decoder** apprende a modellare la distribuzione di probabilità dei dati condizionata alla variabile latente, $p_\theta(x|z)$. Dato un z campionato dallo spazio latente, il decoder genera un output $\hat{x}$.

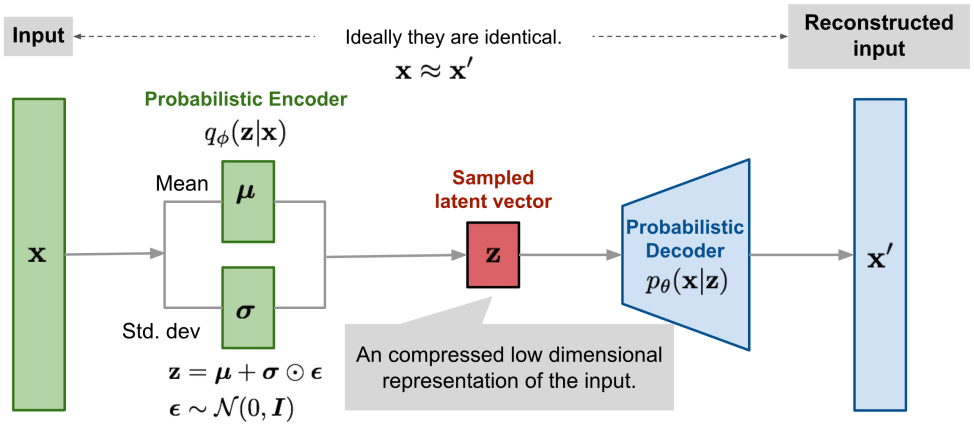

Figure 2.2: Variational autoencoder model [11]

Funzione di Perdita (ELBO)

L'obiettivo dell'addestramento di un VAE non è minimizzare direttamente l'errore di ricostruzione, ma massimizzare l'**Evidence Lower Bound (ELBO)** rispetto ai parametri ϕ dell'encoder e θ del decoder:

$$\mathcal{L}(\theta, \phi; x) = \mathbb{E}_{z \sim q_\phi(z|x)}[\log p_\theta(x|z)] - KL(q_\phi(z|x)||p(z)) \tag{2.2}$$

Questo ELBO è un limite inferiore per la log-verosimiglianza marginale dei dati, $\log p(x)$, ovvero $\mathcal{L}(\theta, \phi; x) \leq \log p(x)$.

L'ottimizzazione dell'ELBO persegue due obiettivi simultaneamente:

- Il primo termine, $\mathbb{E}_{z \sim q_\phi(z|x)}[\log p_\theta(x|z)]$, è il **termine di ricostruzione attesa**. Misura quanto bene, in media, il decoder riesce a ricostruire l'input x partendo da codifiche latenti z campionate dalla distribuzione approssimata $q_\phi(z|x)$. Corrisponde a massimizzare la verosimiglianza di osservare x dato z.
- Il secondo termine, $KL(q_\phi(z|x)||p(z))$, è la **divergenza di Kullback-Leibler** tra la distribuzione a posteriori approssimata $q_\phi(z|x)$ e la distribuzione a priori $p(z)$. Questo termine agisce come un regolarizzatore, penalizzando le distribuzioni $q_\phi(z|x)$ che si discostano troppo dall'a priori $p(z)$. Ciò incoraggia l'encoder a produrre distribuzioni vicine alla Gaussiana standard, favorendo la creazione di uno spazio latente più strutturato, continuo e denso, e rendendolo più adatto per la generazione e l'interpolazione.

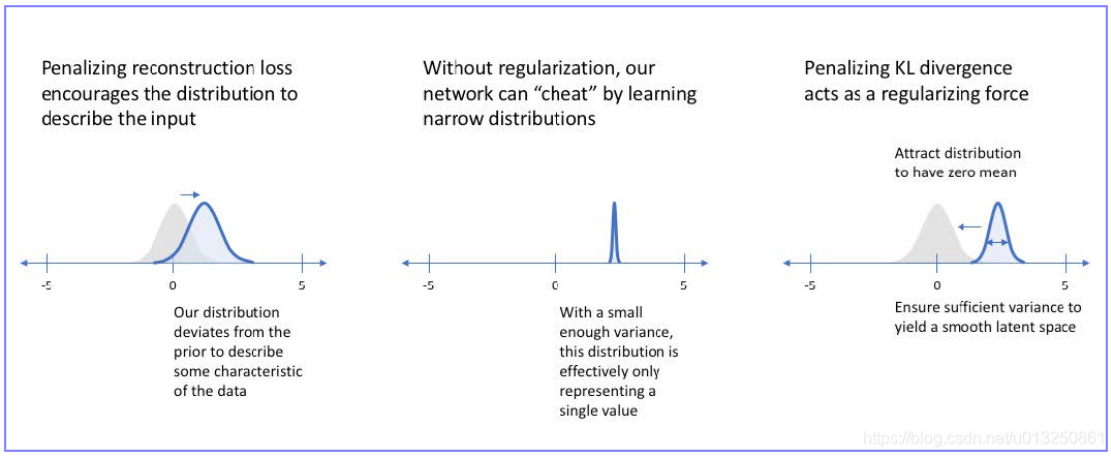

Figure 2.3: Reconstruction Loss - KL Divergence Tradeoff

Il Problema del Campionamento e il Reparameterization Trick

Durante il processo di addestramento di un VAE, il passo in cui un vettore latente z viene campionamento dalla distribuzione $q_\phi(z|x)$ (cioè $z \sim q_\phi(z|x) =$

$\mathcal{N}(z; \mu_\phi(x), \text{diag}(\sigma^2_\phi(x)))$ introduce una **stocasticità non differenziabile** nel grafo computazionale. La backpropagation tradizionale non può calcolare i gradienti attraverso un'operazione di campionamento casuale, impedendo l'aggiornamento dei parametri ϕ dell'encoder, che dipendono proprio da tale campionamento.

Per risolvere questo problema cruciale, si utilizza il Reparameterization Trick. L'idea chiave è quella di **isolare la casualità** in una variabile ausiliaria che non dipende dai parametri dell'encoder ϕ. Si procede come segue:

1. Si campiona una variabile di rumore ausiliaria ϵ da una distribuzione fissa e semplice, tipicamente una Gaussiana standard: $\epsilon \sim \mathcal{N}(0, I)$.

2. Si esprime z come una **funzione deterministica** di $\mu_\phi(x)$, $\sigma_\phi(x)$ e ϵ:
$$z = \mu_\phi(x) + \sigma_\phi(x) \cdot \epsilon$$

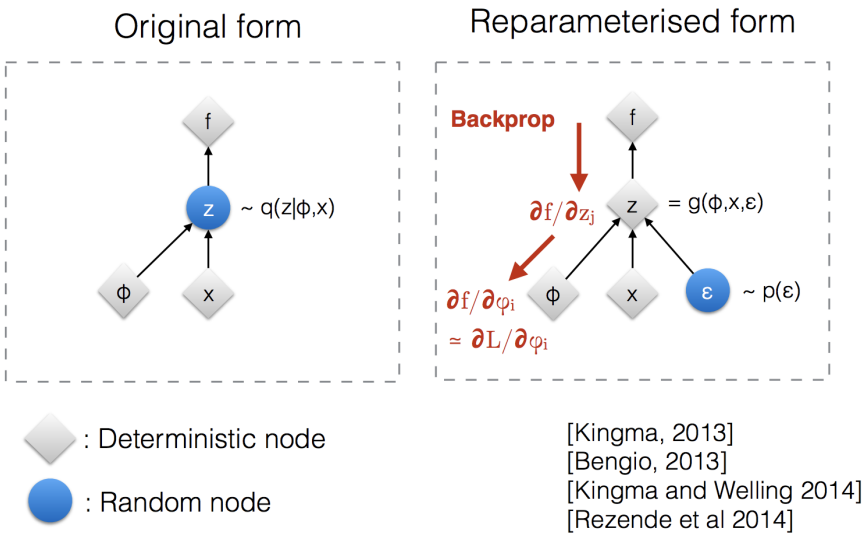

Figure 2.4: reparameterization trick. Source: Slide 12 in Kingmas NIPS 2015 workshop talk

Vantaggio: Ora, z è una funzione differenziabile rispetto a $\mu_\phi(x)$ e $\sigma_\phi(x)$. La casualità è interamente contenuta in ϵ, che è esterna al grafo computazionale che dipende da ϕ. Questo permette ai gradienti di fluire correttamente dalla funzione di perdita, attraverso z, fino ai parametri dell'encoder ϕ, rendendo possibile l'addestramento tramite backpropagation.

IMMAGINI DA: https://lilianweng.github.io/posts/2018-08-12-vae/

2.2.3 Autoencoder Variazionali Condizionali (CVAE)

Gli Autoencoder Variazionali Condizionali (CVAE) [**sohn2015learning**] estendono il framework dei VAE introducendo informazioni aggiuntive, dette **condizioni** c, sia nel processo di decodifica che, opzionalmente, in quello di codifica. Questo permette di guidare e controllare il processo di generazione in base a specifici attributi o etichette desiderate.

Nel contesto della generazione musicale, i CVAE possono essere condizionati su una vasta gamma di parametri musicali o metadati: ad esempio, il genere musicale (es. 'jazz', 'classica'), lo strumento principale (es. 'pianoforte', 'violino'), l'altezza di una nota (pitch), l'intensità (loudness), il tempo (BPM), l'emozione espressa (es. 'felice', 'triste'), o altre caratteristiche rilevanti estratte dalla musica stessa o fornite esternamente come input di controllo.

L'architettura di un CVAE è molto simile a quella di un VAE, ma sia il decoder che eventualmente l'encoder ricevono in input anche la condizione c.

- L'encoder apprende ora la distribuzione della variabile latente z condizionata sia all'input x che alla condizione c: $q_\phi(z|x,c)$.
- Il decoder apprende la distribuzione dei dati x condizionata sia alla variabile latente z che alla condizione c: $p_\theta(x|z,c)$.

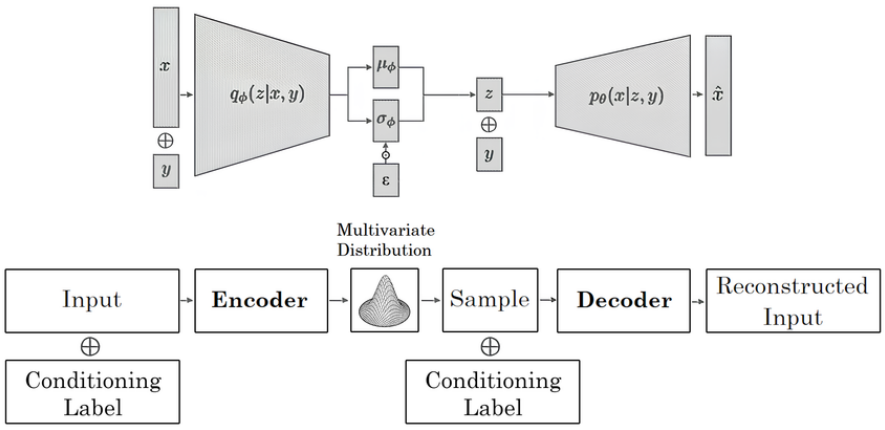

Figure 2.5: Conditional Variational Autoencoder Architecture [8]

Dal punto di vista implementativo, la condizione c (che può essere un vettore one-hot, un embedding appreso, o un insieme di valori numerici) viene tipicamente concatenata:

- All'input x prima di passarlo all'encoder.

- Alla variabile latente z prima di passarla al decoder.

Questo permette al modello di apprendere come generare output x che siano coerenti sia con le caratteristiche generali codificate in z sia con gli attributi specifici imposti dalla condizione c, abilitando così la generazione controllabile.

2.2.4 Latent Space

Un concetto centrale per comprendere il funzionamento intrinseco di architetture come gli Autoencoder Variazionali (VAE) e le loro varianti è quello di **spazio latente**.[2] Immaginiamo i dati di input (come segmenti audio, immagini, o, nel nostro caso, brani musicali) come punti in uno spazio ad altissima dimensionalità. Lo spazio latente è una rappresentazione compressa, a dimensionalità significativamente inferiore, di questi dati. Le "coordinate" di un punto in questo spazio latente sono definite dalle **variabili latenti**, che sono caratteristiche fondamentali e spesso non direttamente osservabili, apprese dal modello per catturare l'essenza e la struttura sottostante dei dati originali, tralasciando rumore e ridondanze. L'obiettivo primario di un autoencoder è, quindi, quello di apprendere una mappatura efficiente da e verso questo spazio latente.

Come detto precedentemente, in un VAE standard l'encoder non mappa l'input x a un singolo punto nello spazio latente, bensì ai parametri di una distribuzione di probabilità su tale spazio. Da questa distribuzione viene poi campionato un vettore latente z, che il decoder utilizzerà per tentare di ricostruire l'input originale. La funzione di perdita del VAE, attraverso il termine di divergenza KL, incoraggia queste distribuzioni a rimanere vicine a una distribuzione a priori (solitamente una Gaussiana standard).

Quando introduciamo il concetto di Autoencoder Variazionale Condizionale (CVAE), la natura e l'interpretazione dello spazio latente z vengono influenzate in modo cruciale dalla modalità con cui l'informazione condizionale c viene integrata nel modello.

Analisi spazio latente

Per investigare questa dinamica, possiamo adottare le seguenti metodologie:

- **Sensitivity Analysis**: Analizzare come l'output varia in risposta ai cambiamenti dei suoi parametri, attraverso una perturbazione delle singole dimensioni latenti e l'osservazione dei cambiamenti nell'audio generato. Misurando l'errore di ricostruzione è possibile identificare quali dimensioni latenti hanno l'impatto più significativo sul suono in uscita.

- **Feature Ablation**: Rimuovere selettivamente alcune features dello spazio latente e osservare come impatta sulla generazione di suoni, permettendo di comprendere il contributo di ciascuna dimensione latente alla sintesi complessiva.

- **Descriptive Statistics and Visualization**: Analizzare le proprietà statistiche dello spazio latente, come se fosse un insieme di punti. Tecniche di visualizzazione, come t-SNE o PCA, potrebbero essere utilizzate per proiettare lo spazio latente ad alta dimensionalità in dimensioni inferiori per l'identificazione di cluster correlati a specificche caratteristiche o strumenti.

CVAE con Condizionamento nell'Encoder e nel Decoder: Se l'informazione condizionale c viene fornita come input aggiuntivo sia all'encoder (insieme ai dati x) che al decoder (insieme al campione latente z), lo spazio latente z subisce una trasformazione significativa nel suo ruolo. Poiché l'encoder "conosce" già la condizione c associata a x, non ha più la necessità di codificare primariamente l'informazione relativa a c all'interno di z. Invece, la variabile latente z impara a rappresentare le caratteristiche specifiche dell'input x che sono pertinenti dato quella specifica condizione c.[10] Ad esempio, nel contesto della generazione di musica emotiva, se c rappresenta l'emozione "gioia", z non codificherà primariamente il fatto che il brano è gioioso (poiché questa informazione è già esplicitata da c). Piuttosto, z catturerà le sfumature stilistiche, le variazioni melodiche, ritmiche o timbriche che rendono un particolare brano gioioso unico rispetto ad altri brani gioiosi. Questo può portare a uno spazio latente più "disentangled" rispetto alla condizione c, dove l'informazione direttamente correlata a c è, idealmente, fattorizzata e gestita esplicitamente, permettendo a z di modellare altri aspetti della variabilità dei dati.

CVAE con Condizionamento solo nel Decoder: Consideriamo ora uno scenario alternativo in cui l'informazione condizionale c è fornita unicamente al decoder, mentre l'encoder opera come in un VAE standard, processando solamente l'input x per produrre la distribuzione su z. In questa configurazione, lo spazio latente z generato dall'encoder ($q_\phi(z|x)$) deve necessariamente tentare di catturare tutte le caratteristiche salienti di x, incluse quelle che potrebbero essere implicitamente correlate alla condizione c (ad esempio, le caratteristiche musicali che oggettivamente definiscono un brano come "triste"), poiché l'encoder

non ha alcuna conoscenza a priori della specifica condizione c che verrà utilizzata in fase di generazione. Il decoder riceve quindi questo vettore latente z (che implicitamente contiene informazioni, ad esempio, sul "mood" originale del brano x) e la condizione target c (ad esempio, l'emozione desiderata per l'output). Il compito del decoder diventa quello di generare un output che, pur basandosi sulle caratteristiche generali apprese e codificate in z, si conformi il più possibile al mood specificato dalla condizione esplicita c. In questo caso, lo spazio latente z è meno direttamente "informato" dalla condizione durante la sua creazione, e il "lavoro" di adattamento alla condizione ricade interamente sul decoder, che deve imparare a modulare l'output basandosi su z e c.

Valence e Arousal

https://psychologyfanatic.com/circumplex-model-of-arousal-and-valence/

2.3 Stato dell'Arte nella Generazione di Musica Affettiva con Deep Learning

Come detto in precedenza, la modellazione diretta delle forme d'onda audio grezze (raw waveform) tramite VAE presenta sfide assai significative a causa dell'altissima dimensionalità temporale, delle dipendenze a lungo termine e della difficoltà nel catturare dettagli percettivamente rilevanti senza introdurre artefatti. Questa sezione esplora lo stato dell'arte dei VAE applicati alla generazione musicale, analizzando diversi modelli che hanno cercato di superare queste sfide, migliorando la qualità della sintesi, l'efficienza computazionale e il controllo sulla generazione.

2.3.1 ENCODEC

2.3.2 Efficient Neural Music Generation (Melody)

2.3.3 RAVE

Un modello particolarmente degno di nota nello sforzo di ottenere sintesi audio neurale di alta qualità e veloce è RAVE (Realtime Audio Variational autoEncoder), introdotto da [4]. L'obiettivo principale di RAVE è consentire la sintesi di forme d'onda audio fino a 48kHz in tempo reale anche su hardware

comune come una CPU, superando i limiti dei precedenti modelli generativi audio, che erano spesso o computazionalmente intensivi (come i modelli autoregressivi), o operavano a basse frequenze di campionamento, sacrificando la qualità audio per ottenere velocità.

RAVE si basa su un'architettura VAE ma introduce diverse innovazioni chiave:

- **Procedura di Addestramento in Due Fasi:**:
  - **Fase 1: Apprendimento della Rappresentazione (Representation Learning)**: In questa fase, il modello (encoder e decoder) viene addestrato come un VAE standard. L'obiettivo è apprendere una buona rappresentazione latente dell'audio. Come funzione di perdita per la ricostruzione, RAVE utilizza una distanza spettrale multiscala (basata sulla STFT a diverse risoluzioni), che è più percettivamente rilevante rispetto a una semplice L1/L2 sulla forma d'onda grezza e tollera meglio le differenze di fase. Viene incluso anche il termine di regolarizzazione KL divergence tipico dei VAE.
  - **Fase 2: Fine-tuning Avversariale (Adversarial Fine-tuning)**: Una volta che la rappresentazione latente è considerata adeguata, l'encoder viene "congelato" (i suoi pesi non vengono più aggiornati). Il decoder viene ulteriormente affinato utilizzando un obiettivo avversariale, simile a quello delle Generative Adversarial Networks (GAN). Viene introdotto un discriminatore che cerca di distinguere tra l'audio reale e quello generato dal decoder a partire dai codici latenti. Il decoder viene addestrato a "ingannare" il discriminatore, migliorando notevolmente la naturalezza e la qualità dell'audio sintetizzato. La perdita include anche la distanza spettrale e una feature matching loss,

- **Decomposizione Multibanda:** Per gestire l'alta frequenza di campionamento (es. 48kHz) in modo efficiente, RAVE applica una decomposizione multibanda alla forma d'onda in ingresso che scompone l'audio in diverse sotto-bande di frequenza, ciascuna con una frequenza di campionamento inferiore. L'encoder e il decoder operano su queste sotto-bande. Questo riduce drasticamente la dimensionalità temporale che il modello deve gestire, permette l'uso di campi recettivi temporalmente più ampi con minore complessità computazionale e contribuisce significativamente alla velocità di sintesi del modello. Il decoder genera le sotto-bande, che vengono poi ricombinate per produrre la forma d'onda finale.

- **Analisi e Compattezza dello Spazio Latente Post-Addestramento:** RAVE propone un metodo per analizzare lo spazio latente dopo l'addestramento e per controllarne la dimensionalità effettiva. Utilizzando la Singular Value Decomposition (SVD) sulla matrice delle medie dei codici latenti prodotti dall'encoder per un set di dati, è possibile identificare le dimensioni latenti più "informative". Introducendo un parametro di fedeltà (f), si può decidere quante dimensioni mantenere, permettendo un trade-off esplicito

tra la fedeltà della ricostruzione e la compattezza della rappresentazione latente.

- **Architettura del Decoder:** Il decoder di RAVE è ispirato ad architetture GAN per l'audio, presentando allo stesso tempo alcune modifiche. Il modello include strati di upsampling e blocchi residuali e l'output finale non è direttamente la forma d'onda, ma viene prodotto da tre sotto-reti: una per la forma d'onda multibanda (waveform) (attivazione `tanh`), una per un inviluppo di loudness (attivazione `sigmoid`) che modula la prima, e una per un sintetizzatore di rumore (noise) che aggiunge rumore filtrato.

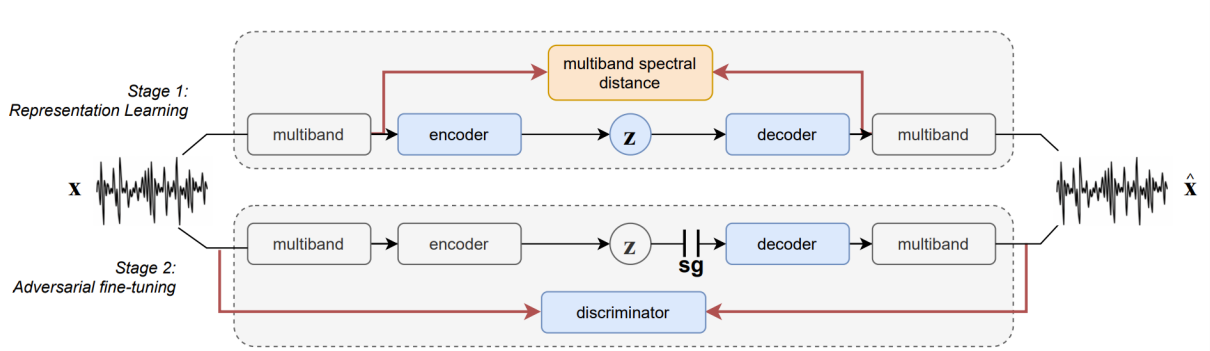

Figure 2.6: RAVE architecture [4]

Risultati e Applicazioni:

Oltre alla sintesi di alta qualità, RAVE si presta bene ad applicazioni come:

- Compressione Audio Neurale: La rappresentazione latente compatta (specialmente dopo la riduzione dimensionale basata sulla fedeltà f) può essere usata come un codec audio neurale efficiente.
- Timbre Transfer: Utilizzando l'encoder per codificare un suono e il decoder (magari addestrato su un altro tipo di suono) per ricostruirlo, si possono ottenere effetti di trasferimento di timbro.

Il modello RAVE rappresenta una significativa fonte di ispirazione per il mio progetto. La sua capacità di coniugare alta qualità audio con un'efficienza computazionale accessibile (a differenza dei precedenti modelli allo stato dell'arte analizzati) e la pertinenza delle sue applicazioni rispetto agli obiettivi del nostro progetto, lo rendono un riferimento particolarmente valido.

Di conseguenza, la nostra architettura generale si baserà sull'impianto VAE di RAVE. In particolare, trarremo ispirazione dall'utilizzo della multiscale spectral loss, e dalla struttura di alcuni layer dell'encoder e del decoder, quali i blocchi residuali e le tecniche di upsampling/downsampling. Tuttavia, per mantenere

una complessità gestibile e per focalizzare lo sviluppo sull'integrazione efficace del meccanismo di conditioning, alcune caratteristiche distintive di RAVE verranno omesse. Nello specifico, non implementeremo la decomposizione multibanda, l'addestramento avversariale con un discriminatore dedicato, né la scomposizione dell'output del decoder nelle tre sotto-reti (forma d'onda, loudness, noise).

Figure 2.7: Example of image

3 Metodology

3.1 Architettura del Modello CVAE

Il Conditional Variational Autoencoder implementato è progettato per la generazione e ricostruzione di segnali audio monodimensionali, con la capacità di condizionare sia il processo di encoding che di decoding tramite un vettore di condizione c. L'architettura si articola in tre componenti principali: un Encoder, un Decoder e una funzione di Loss composita

3.1.1 Encoder

L'Encoder ha il compito di mappare il segnale audio di input `x` (e opzionalmente il vettore di condizione c) in una rappresentazione latente parametrizzata da una media μ e una log-varianza $\log \sigma^2$, estraendo progressivamente caratteristiche significative dall'audio e riducendone la dimensionalità.

La sua architettura è la seguente:

1. **Input**: Il modello riceve in input un **segnale audio `x`**, rappresentato come un tensore con dimensioni `[Batch, NChannels, Time*Sample_rate]`, processando quindi batch di segnali audio, ognuno con un certo numero di canali (es. 1 per mono, 2 per stereo) e una lunghezza totale data dalla durata per la frequenza di campionamento.

   Se il flag `condition_encoder` è attivo riceve inoltre un **vettore di condizione `c`** (tensore `[Batch, CondDim]`).

2. **Convoluzione Iniziale**:

Il segnale audio `x` passa attraverso un layer di convoluzione 1D normalizzata (`wn_conv1d`) con un kernel di dimensione 7 e padding 3 che analizza finestre locali del segnale audio.

Lo scopo è quello di estrarre le features a basso livello dal segnale audio grezzo, mappando gli `n_channels` originali a un numero maggiore di canali, definito da `capacity`. Questo aumento della dimensionalità delle features permette una rappresentazione più ricca nei passaggi successivi.

3. **Proiezione della Condizione**:
   - **Elaborazione di `c`**: Il vettore `c` viene processato da un blocco `cond_proj_encoder`. Questo blocco è composto da due layer lineari (fully connected) separati da funzioni di attivazione LeakyReLU. Esso ha lo scopo di trasformare il vettore di condizione `c` da `cond_dim` a una nuova dimensionalità `encoder_cond_proj_dim`, rendendolo più adatto a essere combinato con le features audio.
   - **Fusione con l'Audio**: L'output `h_c_enc` di questa proiezione viene "espanso" (replicato) lungo la dimensione temporale per corrispondere alla lunghezza delle features audio estratte dalla convoluzione iniziale. Successivamente, viene concatenato a queste features audio lungo la dimensione dei canali. Dopo questa operazione, il numero di canali di input per i blocchi successivi diventa `capacity + encoder_cond_proj_dim`.

4. **Blocchi di Downsampling (Compressione delle Features)**:
   - Segue una sequenza di `n_downsamples` blocchi, ognuno dei quali ha il compito di ridurre la dimensionalità temporale del segnale e di raffinare ulteriormente le features. Ogni blocco è composto da:
     - Una **funzione di attivazione** LeakyReLU che introduce non-linearità.
     - Un layer convoluzionale 1D con kernel di dimensione 4, stride 2, e padding 1. Lo `stride=2` dimezza la lunghezza temporale della rappresentazione ad ogni blocco. Contemporaneamente, il numero di canali in uscita viene raddoppiato. Questo bilanciamento tra riduzione della risoluzione temporale e aumento della ricchezza delle features viene utilizzata per catturare informazioni a diverse scale.
     - **ResidualUnit**: Blocco residuo (descritto più avanti)

5. **Proiezione Finale (Mappatura allo Spazio Latente)**: Un layer convoluzionale finale, con un kernel di dimensione 1x1 agisce come un proiettore, trasformando le features ad alta dimensionalità (provenienti dai blocchi di downsampling) in un numero di canali pari a `latent_dim * 2`.

   I canali risultanti dalla proiezione finale vengono infine divisi a metà lungo la dimensione dei canali per generare i due tensori μ (media) e $\log \sigma^2$ (logaritmo della varianza) della distribuzione Gaussiana nello spazio latente. Entrambi avranno dimensioni `[Batch, LatentDim, ReducedTime]`.

ResidualUnit

Ogni unità residua (`ResidualUnit`) è un blocco di costruzione utilizzato all'interno dei blocchi di downsampling. La sua struttura interna è progettata per raffinare le features in modo efficace attraverso:

- **Normalizzazione**: Normalizza le attivazioni all'interno di ciascun canale stabilizzando l'addestramento e migliorando la generalizzazione
- **Attivazione (`LeakyReLU(0.2)`)**: Introduce non-linearità.
- **Connessione Residua**: L'input originale del blocco `ResidualUnit` viene sommato all'output prodotto dai layer interni del blocco. Questa è la caratteristica distintiva delle unità residue. Permette al network di imparare più facilmente una "funzione identità" (cioè, se la trasformazione ottimale è non fare nulla o fare molto poco, il blocco può impararlo facilmente). Ciò mitiga il problema della scomparsa del gradiente (vanishing gradient) in reti molto profonde, facilitando l'addestramento e permettendo di costruire modelli più potenti. In sostanza, il blocco impara a modificare l'input (il "residuo") piuttosto che dover apprendere l'intera trasformazione da zero.

3.1.2 Decoder

Il Decoder riceve un campione `z` dallo spazio latente (ottenuto tramite riparametrizzazione da μ e $\log \sigma^2$) e, opzionalmente, il vettore di condizione `c`, con l'obiettivo di ricostruire il segnale audio `x_recon`.

La sua architettura è la seguente:

1. **Input**: Campione latente `z` (tensore `[Batch, LatentDim, ReducedTime]`) e, se l'uso del conditioning è `True`, un vettore di condizione `c` (tensore `[Batch, CondDim]`).
2. **Proiezione della Condizione**: Il primo passo consiste nell'elaborare il vettore di condizione `c`. Questo vettore viene passato attraverso un blocco `cond_proj`, una piccola rete neurale composta da due layer lineari intervallati da funzioni di attivazione `LeakyReLU`. L'obiettivo di questo blocco è proiettare `c` dalla sua dimensione originale `cond_dim` a una dimensione più adatta alla concatenazione, definita da `decoder_cond_proj_dim`. L'output di questa proiezione, chiamato `h_c`, viene ulteriormente scalato mediante un fattore `scale_factor`. Questo scalamento permette di modulare l'impatto della condizione sul processo di generazione.
3. **Preparazione Input per Convoluzione Iniziale**: Una volta ottenuta la condizione proiettata `h_c` (di dimensione `[Batch, DecoderCondProjDim]`), questa viene preparata per essere combinata con il campione latente `z`.

Dato che `z` ha una dimensione temporale (`ReducedTime`), anche `h_c` deve essere espansa lungo una dimensione temporale per allinearsi. Successivamente, `h_c` viene concatenata al campione latente `z` lungo la dimensione dei canali. Questa operazione unisce esplicitamente le informazioni latenti del segnale con le informazioni condizionali, creando un input unificato per la rete. Di conseguenza, l'input alla convoluzione iniziale avrà un numero di canali pari a `latent_dim + decoder_cond_proj_dim`.

4. **Convoluzione Iniziale**: L'input combinato (`z` concatenato con `h_c`) viene quindi passato a un layer convoluzionale 1D iniziale (`init_conv`). Questo layer, con un kernel di dimensione 7 e padding 3, ha il compito di espandere il numero di canali, mappando l'input ai canali di partenza per la fase di upsampling.

5. **Blocchi di Upsampling**: Il cuore del decoder è costituito da una sequenza di `n_upsamples` blocchi di upsampling. Ogni blocco è progettato per aumentare progressivamente la risoluzione temporale del segnale e ridurre il numero di canali, "dispiegando" il segnale latente compresso in una rappresentazione più dettagliata:

   - **Upsampling**: Ciascun blocco inizia con un layer `nn.Upsample` che raddoppia la dimensione temporale del feature map, utilizzando la modalità 'nearest'. Questo aumenta la risoluzione spaziale del segnale.
   - **Convoluzione**: Segue un layer `wn_conv1d` (convoluzione 1D con Weight Normalization) con un kernel di dimensione 3 e padding 1. Questo strato elabora i feature upsamplati e, ad ogni blocco, dimezza il numero di canali in uscita, garantendo che non scenda mai al di sotto del valore minimo `capacity`.
   - **Unità Residua**: Viene applicata una `ResidualUnit` (come descritto in dettaglio nella Sezione **??**). Questa unità è cruciale per migliorare la stabilità dell'addestramento e facilitare il flusso dei gradienti, permettendo al modello di apprendere mappature complesse e profonde.
   - **Attivazione**: Ogni blocco si conclude con una funzione di attivazione `LeakyReLU(0.2)`, che introduce la non-linearità necessaria per la capacità espressiva del modello.

   Questo processo iterativo di upsampling e trasformazione garantisce che il decoder possa ricostruire i dettagli fini nel segnale audio a risoluzioni sempre crescenti.

Blocco Finale di Output

Dopo aver attraversato tutti i blocchi di upsampling, il feature map risultante passa attraverso un blocco finale che genera il segnale audio grezzo:

- Inizia con una funzione di attivazione `LeakyReLU(0.2)`.

- Segue un layer `wn_conv1d` con un kernel di dimensione 7 e padding 3. Questo layer finale è responsabile di mappare il numero di canali attuali (che sarà pari a `capacity`) al numero di canali di output desiderato per il segnale audio (`n_channels`, tipicamente 1 per audio mono).
- L'ultimo passo è l'applicazione di una funzione di attivazione `nn.Tanh()`. Questa funzione scala l'output nell'intervallo `[-1, 1]`, un formato standard per le rappresentazioni di forme d'onda audio.

Output del Decoder

L'output finale del decoder è il tensore `x_recon`, che rappresenta il segnale audio ricostruito. Questo tensore avrà la forma `[Batch, NChannels, Time]`, e il suo contenuto è il risultato della decodifica del campione latente `z`, guidata e modellata in modo specifico dalle informazioni contenute nel vettore di condizione `c`.

3.1.3 Losses

La funzione di loss totale (*total loss*) utilizzata per l'addestramento del CVAE è una combinazione pesata della loss di ricostruzione e della divergenza di Kullback-Leibler (KL) ed è definita come: `loss = recon_loss + β * kl_loss`.

- **Loss di Ricostruzione (`recon_loss`)**: È una somma ponderata di due componenti principali:
  - **`MultiScaleSpectralLoss`**: Questa loss opera nel dominio della frequenza, confrontando gli spettrogrammi del segnale originale e quello ricostruito su diverse scale spettrali. Per ciascuna scala, calcola le seguenti componenti:
    * Una loss L1 sulla magnitudine logaritmica dello spettro STFT
    * Una loss L1 sulla magnitudine lineare dello spettro STFT.

    Le componenti di loss calcolate su ogni scala vengono mediate, e le due componenti (logaritmica e lineare) sono sommate con pesi `loss_sc_weight` e `loss_mag_weight`.
    I parametri STFT per le diverse scale sono configurati come segue:
    * `fft_sizes`: `[2048, 1024, 512, 256]`
    * `hop_sizes`: `[240, 120, 50, 25]`
    * `win_lengths`: `[1200, 600, 240, 100]`

    Di seguito, una breve descrizione di ciascun parametro:

Descrizione dei Parametri

* **win_lengths** Questo parametro definisce la lunghezza, in campioni, della finestra temporale che viene applicata a ciascun segmento del segnale audio prima di calcolare la FFT.
  - Una finestra più lunga offre una migliore risoluzione in frequenza, permettendo di distinguere frequenze più vicine tra loro. Tuttavia, peggiora la risoluzione temporale, poiché gli eventi sonori rapidi (es. attacchi di note) sono "spalmati" su un periodo più lungo.
  - Una finestra più corta cattura meglio i cambiamenti temporali rapidi, ma sacrifica la precisione nella determinazione delle frequenze.

* **hop_sizes** Indica il numero di campioni di cui la finestra di analisi si sposta in avanti per generare il segmento successivo. Determina il grado di sovrapposizione tra i frame consecutivi.
* **fft_sizes** Questo parametro specifica il numero di punti per il calcolo della Trasformata di Fourier Discreta (DFT) di ciascun segmento finestrato. Non deve necessariamente essere uguale a `win_length`.

– **`time_loss`**: È una **loss L1** calcolata direttamente nel dominio del tempo tra il segnale originale `x` e quello ricostruito `x_recon`. La loss L1, o Mean Absolute Error (MAE), misura la somma delle differenze assolute tra i valori corrispondenti di due segnali. In questo contesto, $L1(A, B) = \frac{1}{N} \sum_{i=1}^{N} |A_i - B_i|$.
Mentre le loss spettrali si concentrano sulla rappresentazione in frequenza del suono, la `time_loss` agisce direttamente sulla forma d'onda campionata, contribuendo a mantenere la forma complessiva del segnale ricostruito il più vicino possibile all'originale e assicurandosi una maggior fedeltà nella riproduzione di attacchi rapidi (come percussioni)
Questa componente è opzionale e viene aggiunta alla MultiScaleSpectralLoss.
La formula finale per la loss di ricostruzione è:

`recon_loss` = `MultiScaleSpectralLoss`+`hparams.time_loss`·L1_loss

3.2 Preprocessing

Il preprocessing dei dati rappresenta una fase critica e fondamentale nella creazione di un'architettura ottimale. Questo processo include la valutazione, il filtraggio, la manipolazione e la codifica delle informazioni grezze, svolti con

l'obiettivo primario di renderle comprensibili, coerenti e ottimali per l'addestramento di architetture neurali complesse. L'obiettivo primario di questa fase è risolvere le problematiche intrinseche ai dati grezzi, come possono essere incompletezza, disomogeneità o rumore, elevandone complessivamente la qualità.

Nel contesto specifico del nostro modello di generazione musicale, che opera su dati audio raw, le azione di preprocessing applicate sono le seguenti:

- **Conversione da Stereo a Mono:** Tutte le tracce audio MP3 raw, originariamente registrate in formato stereo, sono state convertite in mono facendo una media delle due tracce. Questa operazione dimezza la dimensionalità dei dati di input, semplificando il compito del modello senza compromettere le informazioni acustiche.
- **Uniformità della Frequenza di Campionamento:** Per garantire la massima compatibilità, è stata uniformata la frequenza di campionamento (sampling rate) per tutte le tracce audio presenti nel dataset.
- **Rimozione Bordi:** Per mitigare l'influenza di dissolvenze, silenzi o rumori non pertinenti spesso presenti all'inizio e alla fine delle registrazioni, sono stati rimossi i primi e gli ultimi 10 secondi di ogni traccia.
- **Normalizzazione:** Ogni segmento audio è statonormalizzato, scalando i valori di ampiezza di ciascun segmento all'interno di un intervallo predefinito ([-1, 1]), prevenendo che picchi di volume eccessivi in singole tracce dominino il processo di apprendimento.
- **Segmentazione con Contesto Temporale:** Le tracce audio, di lunghezza variabile, sono state segmentate in porzioni fisse di pochi secondi. Questo approccio è necessario poiché le reti neurali operano tipicamente con input di dimensione fissa e per aumentare significativamente il numero di campioni di training disponibili.

  Durante questa segmentazione, per ogni segmento è stata mantenuta una brevissima porzione (overlap) del segmento precedente e successivo. Questa tecnica è stata implementata per fornire al modello un contesto temporale più ampio, permettendogli di comprendere meglio le transizioni e la coerenza melodica e armonica tra i segmenti.

3.3 Datasets

Per lo sviluppo e la validazione del modello di generazioni di musica condizionata proposto in questa tesi, sono stati utilizzati diversi dataset musicali, contenenti segmenti di audio raw e informazioni sullo stato emotivo dell'audio stesso.

3.3.1 Jamendo Dataset

Il Jamendo Dataset [3] è una vasta collezione di 55.000 tracce audio in formato MP3, arricchite da 195 tag che ne categorizzano genere musicale, strumenti e diversi mood.

Nel nostro caso specifico, ovvero quello della musica affective, ci siamo concentrati su un sottoinsieme di questo dataset, appositamente selezionato ed utilizzato per il "Emotion and Theme recognition in Music Task" del MediaEval 2019-2021. Questo sottoinsieme comprende 18.486 tracce audio, tutte annotate con tag di mood/theme, per un totale di 57 tag unici.

Esempi di questi tag descrittivi del mood/theme dei brani sono:

- `mood/theme-calm`
- `mood/theme-dark`
- `mood/theme-dramatic`
- `mood/theme-epic`
- `mood/theme-love`
- `mood/theme-party`

Al fine di ottimizzare i tempi di addestramento e ridurre lo spazio di archiviazione necessario, le tracce audio di questo sottoinsieme sono state scaricate in una versione a più bassa qualità. Questa scelta ha ridotto il volume totale dei dati, passando da 152 GB a circa 46 GB, pur mantenendo la completezza del contenuto musicale in termini di durata.

Dato che questi tag originali sono puramente descrittivi (semantici) e il nostro obiettivo prevede il condizionamento del modello tramite le dimensioni di Valence e Arousal, abbiamo associato, dove possibile, dei valori numerici di Valence e Arousal ad ogni tag di mood/theme. Questi valori sono stati assegnati manualmente sulla base di un'interpretazione consolidata delle emozioni evocate da ciascun tag, permettendo una rappresentazione più quantificabile dello stato emotivo.

Di seguito, alcuni esempi di questa associazione:

- `mood/theme-calm` → (**Valence**: 0.6, **Arousal**: -0.7)"
- `mood/theme-epic` → (**Valence**: 0.6, **Arousal**: 0.8)"
- `mood/theme-game` → (**Valence**: N/A, **Arousal**: N/A) - "Non classificabile"
- `mood/theme-love` → (**Valence**: 0.8, **Arousal**: 0.4)

Quantitativamente, il sottoinsieme del Jamendo Dataset utilizzato consiste in circa 25.815 minuti (circa 430 ore) di musica. Come accennato, l'ingombro su disco di questa versione a bassa qualità è di circa 46 GB. Tuttavia, per via di esigenze legate allo spazio disco e, in particolare, alle limitate risorse computazionali in termini soprattutto temporali, si è reso necessario utilizzare per l'addestramento una porzione significativamente inferiore di tale dataset.

3.3.2 DEAM Dataset

Composizione

Il dataset DEAM [5] è un'aggregazione di dati raccolti durante le campagne di benchmarking MediaEval "Emotion in Music" tra il 2013 e il 2015. Esso comprende oltre 1800 brani musicali royalty-free con una frequenza di campionamento uniforme di 44100Hz, provenienti da diverse fonti come ad esempio freemusicarchive.org, jamendo.com e il dataset MedleyDB.

Sia gli estratti che i brani interi sono disponibili in formato MP3 e hanno un peso complessivo di 1.7 GB.

Annotazioni Emotive: Arousal e Valence

Il dataset fornisce inoltre annotazioni emotive continue lungo le dimensioni di valence e arousal, normalizzate nell'intervallo [-1, +1]. Queste annotazioni sono disponibili in formato CSV e si distinguono in due tipologie principali: statiche e dinamiche.

Annotazioni Statiche: Le annotazioni statiche forniscono un singolo valore medio di valence e arousal per l'intera durata di ciascun brano.

Annotazioni Dinamiche: Le annotazioni dinamiche catturano l'evoluzione temporale delle emozioni percepite durante l'ascolto, fornendo un valore di valence e arousal ogni 0.5 secondi. Nel caso di annotazioni dinamiche, il modello eseguirà una media dei valori di valence e arousal lungo la lunghezza del segmento fissato.

3.4 Training del Modello

Framework e librerie (PyTorch Lightning, Wandb, ecc.) Hardware utilizzato (GPU a40, Multi-GPU training "ddp") Dettagli sull'ottimizzatore (Adam, SGD),

learning rate, scheduler. Batch size = 16. Numero di epoche (50 per fase di test, 200 trovati i giusti iperparametri), eventuali strategie di early stopping. Come hai diviso i dataset (training 0.85, validation 0.15, test sets 0.0).

3.5 Metriche di Valutazione (Evaluation Metrics)

Algorithm 1 An algorithm with caption

Require: $n \geq 0$
Ensure: $y = x^n$
  $y \leftarrow 1$
  $X \leftarrow x$
  $N \leftarrow n$
  while $N \neq 0$ **do**
    if N is even **then**
      $X \leftarrow X \times X$
      $N \leftarrow \frac{N}{2}$ {This is a comment}
    else if N is odd **then**
      $y \leftarrow y \times X$
      $N \leftarrow N - 1$
    end if
  end while

$$e^{j\pi} + 1 = 0 \tag{3.1}$$

4

Risultati e Analisi

Mostra esempi audio (magari tramite link a una pagina web o repository). Visualizza le analisi dello spazio latente. Riporta i risultati delle metriche oggettive e soggettive in tabelle e grafici chiari. Analizza criticamente i risultati: Cosa funziona bene? Cosa meno? Perché? Quali sono i limiti?

Lo Studio dello Spazio Latente come Strumento Diagnostico:

Perché la ricostruzione funziona bene? Il tuo studio dello spazio latente può comunque analizzare come il modello codifica l'informazione per la ricostruzione. Ci sono cluster riconoscibili? L'interpolazione nello spazio latente porta a ricostruzioni intermedie sensate? Questo di per sé è un risultato.

Perché il condizionamento e il campionamento falliscono? Qui lo studio dello spazio latente diventa uno strumento per diagnosticare il problema. Invece di mostrare come funziona bene, mostri perché non funziona.

Visualizzazioni (t-SNE, PCA): Se colori i punti nello spazio latente in base alle condizioni emotive (Valence/Arousal), cosa vedi? Se le diverse emozioni sono tutte sovrapposte e non formano cluster distinti, questo è un risultato e spiega perché il condizionamento è difficile.

Interpolazione Condizionata: Cosa succede se provi a interpolare tra due punti nello spazio latente che dovrebbero corrispondere a emozioni diverse, o se campioni da regioni che dovrebbero corrispondere a una certa emozione? Se l'output non riflette la condizione, l'analisi dello spazio latente può aiutare a illustrare questa disconnessione.

Sensitivity Analysis / Feature Ablation: Anche se il condizionamento generale fallisce, ci sono singole dimensioni latenti che, se perturbate, hanno un

impatto (anche se non quello desiderato o non abbastanza forte) su aspetti percepiti come emotivi? O magari su altre qualità timbriche? Questo può rivelare se il modello sta tentando di apprendere qualcosa legato alle condizioni, ma non ci riesce in modo efficace.

KL Vanishing / Posterior Collapse: Il tuo studio dello spazio latente potrebbe indicare che il termine KL nella loss del VAE sta "vincendo" troppo, costringendo lo spazio latente ad assomigliare troppo a una gaussiana standard e ignorando le informazioni di input o di condizione. Questo è un problema comune nei VAE. Se le tue codifiche latenti sono tutte ammassate in una piccola regione, indipendentemente dall'input o dalla condizione, questo è un risultato importante.

4.1 Experimental Setup

In this section, we provide a summary of the experiments that lead us to choose our final parameters

4.1.1 Evaluation measures

ESISTONO EVALUATION OGGETTIVE?

4.2 Analisi

Ricostruzione: Mostra che per la ricostruzione lo spazio latente "funziona" (se è così). Visualizzazioni: Anche se non mostrano la separazione desiderata per le emozioni, le visualizzazioni sono importanti per mostrare questa mancanza di separazione. Tentativi di Campionamento: Mostra esempi di cosa succede quando campioni. Se i risultati non sono buoni, documentalo. Discussione Critica: Questa è la parte fondamentale. Dopo aver presentato i risultati (anche quelli negativi), devi discuterli criticamente. Perché pensi che il modello abbia fallito in questi aspetti? Quali sono le possibili cause (architettura, loss, dati, complessità intrinseca del problema audio raw)?

4.3 Rappresentazione spazio latente

Paragonare spazio in cui reconstruction è perfetta e spazio in cui è pessima ma con beta alto (basarsi su file audio che ho nella cartella, usare quasi solo quelli volendo che sono tanti e hanno numerose informazioni) Ho modificato il modello però, non è piu utilizzabile forse. capire come fare

4.4 Analisi spazio latente

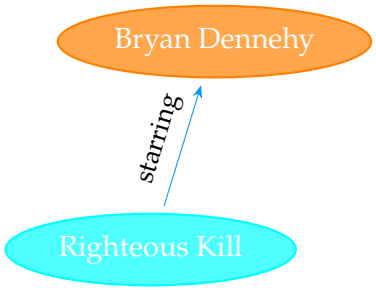

Figure 4.1: Image created with TikZ

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

```
import numpy as np

def incmatrix(genl1,genl2):
    m = len(genl1)
    n = len(genl2)
    M = None #to become the incidence matrix
    VT = np.zeros((n*m,1), int)  #dummy variable

    test = "String"

    #compute the bitwise xor matrix
    M1 = bitxormatrix(genl1)
    M2 = np.triu(bitxormatrix(genl2),1)
```

```
14
15      for i in range(m-1):
16          for j in range(i+1, m):
17              [r,c] = np.where(M2 == M1[i,j])
18              for k in range(len(r)):
19                  VT[(i)*n + r[k]] = 1;
20                  VT[(i)*n + c[k]] = 1;
21                  VT[(j)*n + r[k]] = 1;
22                  VT[(j)*n + c[k]] = 1;
23
24                  if M is None:
25                      M = np.copy(VT)
26                  else:
27                      M = np.concatenate((M, VT), 1)
28
29                  VT = np.zeros((n*m,1), int)
30
31      return M
```

Code 4.1: Code snippet example

4.5 MIDI Transformer

Collegamento con il Lavoro MIDI/Transformer: L'analisi delle difficoltà incontrate con il CVAE su audio raw può fungere da motivazione ancora più forte per il tuo lavoro esplorativo con MIDI e Transformer (che presenterai nelle conclusioni/future works). Potresti argomentare che le problematiche riscontrate nello spazio latente del CVAE (es. entanglement delle features, difficoltà a separare contenuto e attributi emotivi nell'audio raw) suggeriscono che rappresentazioni più strutturate e simboliche (come il MIDI) potrebbero essere più adatte per un controllo emotivo fine, specialmente con architetture potenti come i Transformer.

5

Conclusions and Future Works

Riassumi brevemente il problema e gli obiettivi. Sintetizza i risultati principali ottenuti (sia positivi che negativi/limitazioni). Rispondi alle tue domande di ricerca. Evidenzia il contributo della tesi. Discuti le limitazioni del tuo lavoro. Struttura per Future Works: Idee per migliorare il modello CVAE attuale. Esplorazione approfondita dell'approccio MIDI+Transformer (qui puoi espandere molto, dati i tuoi commenti). Altre direzioni di ricerca.

In quanto insoddisfatto dai risultati concreti del modello e volendo avere un qualcosa di cocnreto da adcoltare ho provato a svolgere lo stesso lavoro (quindi comditioninf tramite V/A) utilizzando una rappresentazione simbolica, in particolare MIDI utilizzando come architettura un transformer. In questo caso i risultati sono stati molto soddisfacenti, grazie sia a una rappresentazione molto molto meno complessa e facilmente tokenizzabile, sia tramite l'uso dei transformer, tecnologia che ha rivoluzionato il campo dell' IA

L'esperienza di ricerca condotta per questa tesi, in particolare nell'affrontare la complessità intrinseca dei dati audio raw senza l'ausilio di pre-elaborazioni massive o modelli pre-addestrati, ha permesso di toccare con mano una delle sfide più significative nell'attuale panorama dell'Intelligenza Artificiale applicata. Il mancato raggiungimento di risultati pienamente comparabili a quelli ottenuti da colossi tecnologici come Meta e Google, pur lavorando su problematiche concettualmente simili, evidenzia in modo netto il divario abissale in termini di risorse computazionali disponibili. A fronte delle imponenti infrastrutture che permettono a questi attori industriali di addestrare modelli su scale immense talvolta migliaia di volte superiori a quelle accessibili in contesti

accademici la ricerca universitaria si scontra con limiti oggettivi di scala. Questo scenario porta a una crescente concentrazione del potere computazionale nelle mani di pochi attori privati, ridefinendo di fatto il ruolo dell'accademia. La ricerca universitaria nel campo dei modelli di IA di frontiera si troverà forse a doversi concentrare maggiormente sull'analisi critica dei modelli pre-esistenti, sul loro adattamento (fine-tuning) per applicazioni di nicchia o specialistiche, o sullo studio di aspetti teorici, etici e metodologici, piuttosto che sulla competizione diretta nella creazione di modelli fondamentali su larga scala partendo da zero. Sebbene questo non sminuisca il valore fondamentale della ricerca accademica in altri ambiti dell'IA, solleva interrogativi cruciali sull'accesso alle tecnologie di punta, sulla democratizzazione della ricerca e sulla potenziale centralizzazione dell'innovazione fondamentale nel settore."

Opzione 2 (Leggermente più focalizzata sul trend generale, usando la tesi come esempio): "I risultati ottenuti (o la loro assenza significativa in relazione agli obiettivi iniziali) nel corso di questa tesi, lavorando con la notevole complessità dei dati audio raw, offrono una chiara esemplificazione di un trend più ampio e preoccupante nel campo dell'Intelligenza Artificiale: la crescente, e apparentemente inarrestabile, concentrazione del potere computazionale nelle mani di un ristretto numero di grandi aziende tecnologiche. Il confronto con i successi ottenuti da entità come Meta e Google su problematiche analoghe rese possibili dall'impiego di risorse computazionali vastissime, ordini di grandezza superiori a quelle tipiche di un ambiente accademico mette in luce una disparità di scala che incide profondamente sulla natura stessa della ricerca. In un tale ecosistema, l'università e i ricercatori accademici si trovano inevitabilmente limitati nella loro capacità di addestrare nuovi modelli di grandi dimensioni partendo da zero, specialmente per compiti computazionalmente esigenti come l'elaborazione di dati raw complessi. Il loro ruolo è pertanto destinato a evolvere, focalizzandosi verosimilmente sempre più sull'analisi approfondita e sulla validazione dei modelli già esistenti, sull'applicazione e l'adattamento (fine-tuning) di tali modelli a specifici domini di ricerca o problematiche settoriali, e sullo sviluppo teorico e metodologico che non dipenda in modo critico dalla disponibilità di supercomputer. Questa dinamica, se da un lato spinge verso nuove forme di collaborazione e specializzazione accademica, dall'altro solleva serie preoccupazioni riguardo l'accesso equo alle infrastrutture di ricerca avanzate e il rischio di una 'privatizzazione' della ricerca fondamentale sull'IA su larga scala."

A	**B**
C	D
E	F
G	H

Table 5.1: Table example

References

[1] Marco Alecci et al. "Development of an IR System for Argument Search." In: *CLEF (Working Notes)*. 2021, pp. 2302–2318.

[2] Dave Bergmann. *What is latent space?* Jan. 28, 2025. URL: https://www.ibm.com/think/topics/latent-space.

[3] Dmitry Bogdanov et al. "The MTG-Jamendo Dataset for Automatic Music Tagging". In: *Machine Learning for Music Discovery Workshop, International Conference on Machine Learning (ICML 2019)*. Long Beach, CA, United States, 2019. URL: http://hdl.handle.net/10230/42015.

[4] Antoine Caillon and Philippe Esling. "RAVE: A variational autoencoder for fast and high-quality neural audio synthesis". In: *arXiv preprint arXiv:2111.05011* (2021). arXiv: 2111.05011 [cs.LG]. URL: https://arxiv.org/abs/2111.05011.

[5] Florian Eyben et al. "Recent Developments in openSMILE, the Munich Open-source Multimedia Feature Extractor". In: *Proceedings of the 21st ACM International Conference on Multimedia*. MM 13. Barcelona, Spain: ACM, 2013, pp. 835–838. ISBN: 978-1-4503-2404-5. DOI: 10.1145/2502081.2502224. URL: http://doi.acm.org/10.1145/2502081.2502224.

[6] G. E. Hinton and R. R. Salakhutdinov. "Reducing the dimensionality of data with neural networks". In: *Science* 313.5786 (2006), pp. 504–507. DOI: 10.1126/science.1127647. URL: https://pubmed.ncbi.nlm.nih.gov/16873662/.

[7] Diederik P Kingma and Max Welling. "Auto-Encoding Variational Bayes". In: *arXiv preprint arXiv:1312.6114* (2013). DOI: 10.48550/arXiv.1312.6114. arXiv: 1312.6114 [stat.ML]. URL: https://arxiv.org/abs/1312.6114.

[8] Eoin MacDomhnaill et al. *Conditional Variational Autoencoder (CVAE) Architecture. Figura da 'Modelling forest fire dynamics using conditional variational autoencoders'*. 2024. URL: https://www.researchgate.net/figure/Conditional-Variational-Autoencoder-CVAE-Architecture-The-encoder-qphzx-y_fig17_381655760.

[9] Anastasia Natsiou, Seán O'Leary, and Luca Longo. "An Exploration of the Latent Space of a Convolutional Variational Autoencoder for the Generation of Musical Instrument Tones". In: *Explainable Artificial Intelligence: International Workshop, xAI 2023, Valetta, Malta, August 24, 2023, Proceedings*. Ed. by Luca Longo. Vol. 1903. Communications in Computer and Information Science. Springer Nature Switzerland AG, 2023, pp. 470–486. DOI: 10.1007/978-3-031-44070-0_24. URL: https://doi.org/10.1007/978-3-031-44070-0_24.

[10] Tim Rose. *Conditional Variational Autoencoders with Learnable Conditional Embeddings*. Jan. 8, 2024. URL: https://towardsdatascience.com/conditional-variational-autoencoders-with-learnable-conditional-embeddings-e22ee5359a2a/.

[11] Lilian Weng. "From Autoencoder to Beta-VAE". In: *lilianweng.github.io* (2018). URL: https://lilianweng.github.io/posts/2018-08-12-vae/.

Acknowledgments