Neural audio synthesis of realistic piano performances

Lenny Renault

Doctoral Thesis from Sorbonne Université

Spécialité **Informatique**
École Doctorale Informatique, Télécommunications et Électronique de Paris (EDITE - ED130)
Sciences et Technologies de la Musique et du Son (STMS - UMR 9912)
Institut de Recherche et de Coordination Acoustique Musique (IRCAM)
Équipe Analyse et Synthèse des Sons

Neural Audio Synthesis of Realistic Piano Performances

Author:
Lenny RENAULT

Under the supervision of:
Dr. Rémi MIGNOT (IRCAM)
Dr. Axel ROEBEL (IRCAM)

Defended on the 8th of July 2024 before the Jury composed of:

Reviewers:
Pr. **Mark SANDLER** — Professor, Queen Mary University of London
Dr. **Mathieu LAGRANGE** — HDR Researcher, CNRS, École Centrale de Nantes

Examiners:
Pr. **Gaël RICHARD** - *President of the Jury* — Professor, Télécom Paris
Dr. **Jesse ENGEL** — Staff Research Scientist, Google DeepMind
Dr. **Juliette CHABASSIER** — Research Scientist, Modartt

Director:
Dr. **Axel ROEBEL** — Research Director, IRCAM

Horizon 2020
European Union funding
for Research & Innovation

À Yeye, Gros-Père et Granny.

Abstract

Musician and instrument make up a central duo in the musical experience. Inseparable, they are the key actors of the musical performance, transforming a composition into an emotional auditory experience. To this end, the instrument is a sound device, that the musician controls to transcribe and share their understanding of a musical work. Access to the sound of such instruments, often the result of advanced craftsmanship, and to the mastery of playing them, can require extensive resources that limit the creative exploration of composers. This thesis explores the use of deep neural networks to reproduce the subtleties introduced by the musician's playing and the sound of the instrument, making the music realistic and alive. Focusing on piano music, the conducted work has led to a sound synthesis model for the piano, as well as an expressive performance rendering model. DDSP-Piano, the piano synthesis model, is built upon the hybrid approach of Differentiable Digital Signal Processing (DDSP), which enables the inclusion of traditional signal processing tools into a deep learning model. The model takes symbolic performances as input and explicitly includes instrument-specific knowledge, such as inharmonicity, tuning, and polyphony. This modular, lightweight, and interpretable approach synthesizes sounds of realistic quality while separating the various components that make up the piano sound. As for the performance rendering model, the proposed approach enables the transformation of MIDI compositions into symbolic expressive interpretations. In particular, thanks to an unsupervised adversarial training, it stands out from previous works by not relying on aligned score-performance training pairs to reproduce expressive qualities. The combination of the sound synthesis and performance rendering models would enable the synthesis of expressive audio interpretations of scores, while enabling modification of the generated interpretations in the symbolic domain.

Résumé

Musicien et instrument forment un duo central de l'expérience musicale. Indissociables, ils sont les acteurs de la performance musicale, transformant une composition en une expérience auditive émotionnelle. Pour cela, l'instrument est un objet sonore que le musicien contrôle pour retranscrire et partager sa compréhension d'une oeuvre musicale. Accéder aux sonorités d'un tel instrument, souvent issu de facture poussée, et à sa maîtrise de jeu, requiert des ressources limitant l'exploration créative des compositeurs. Cette thèse explore l'utilisation des réseaux de neurones profonds pour reproduire les subtilités introduites par le jeu du musicien et par le son de l'instrument, rendant la musique réaliste et vivante. En se focalisant sur la musique pour piano, le travail réalisé a donné lieu à un modèle de synthèse sonore pour piano ainsi qu'à un modèle de rendu de performances expressives. DDSP-Piano, le modèle de synthèse de piano, est construit sur l'approche hybride de Traitement du Signal Différentiable (DDSP) permettant d'inclure des outils de traitement du signal traditionnel dans un modèle d'apprentissage profond. Le modèle prend des performances symboliques en entrée, et inclut explicitement des connaissance spécifiques à l'instrument, telles que l'inharmonicité, l'accordage et la polyphonie. Cette approche modulaire, légère et interprétable synthétise des sons d'une qualité réaliste tout en séparant les différents éléments constituant le son du piano. Quant au modèle de rendu de performance, l'approche proposée permet de transformer des compositions MIDI en interprétations expressives symboliques. En particulier, grâce à un entraînement adverse non-supervisé, elle dénote des travaux précédents en ne s'appuyant pas sur des paires de partitions et d'interprétations alignées pour reproduire des qualités expressives. La combinaison des deux modèles de synthèse sonore et de rendu de performance permettrait de synthétiser des interprétations expressives audio de partitions, tout en donnant la possibilité de modifier, dans le domaine symbolique, l'interprétation générée.

Remerciements

Tout d'abord, je tiens à remercier profondément mes encadrants, Axel et Rémi, pour m'avoir guidé tout au long de ces années avec bienveillance, en me laissant de nombreuses libertés pour explorer la recherche, tout en étant au rendez-vous lorsque j'avais besoin de leurs conseils.

Je tiens également à exprimer ma gratitude à tous les chercheurs qui m'ont encadré, dont l'aide a été précieuse avant, durant, et à la fin de cette thèse : Andea, Romain et l'équipe de Deezer Research, qui m'ont formé à mes débuts ; Geoffroy et Philippe, qui m'ont suivi et conseillé pendant ces années ; Mathieu et Mark, qui ont examiné ce manuscrit en long, en large et en travers ; ainsi que Gaël, Jesse et Juliette, qui ont généreusement accepté de lire ce manuscrit et d'assister à la soutenance en tant que membres du jury de thèse.

Un grand merci aux membres de l'équipe Analyse et Synthèse de Sons avec qui j'ai partagé les serveurs GPU, et surtout d'innombrables discussions passionnantes : Daniel, Nicolas, Guillaume D., Frédéric, Frederik, Clément, Antoine C., Antoine L., Léane, Giovanni, David, Nils, Sarah, Théodore, Maximino, Théo, Balthazar, Pierre-Hugo et Luc. Mention spéciale aux meilleurs co-bureaux que je pouvais avoir et qui ont rendu ce périple inoubliable : Alice, Mathilde et Yann.

Une chaleureuse pensée envers les autres doctorants de l'Ircam avec qui j'ai pu partager cette aventure : Victor R., Claire, Constance, Valérian, Baptiste, Gonzalo, Nadia, Apolline et l'inarrêtable Paul. En particulier, Salah, Loïc, Colette, Aliénor, Thomas R., Vincent et Victor P. pour cette palpitante organisation des JJCAAS. Merci également à la gentillesse du personnel de l'Ircam : entre autres, Hugues, Deborah, Éric, Léo et Brigitte.

Je remercie également Fred et ADAPTAC Paris 13, ainsi que les Marchands de Groove, pour m'avoir permis de m'épanouir sportivement et musicalement en dehors du cadre de la thèse. Merci à Florence d'avoir éveillé ma curiosité musicale et de m'avoir transmis cette fascination pour ce bel instrument qu'est le piano.

Grosse pensée pour les Télécommiens avec qui j'ai partagé tant de moments précieux : GJ, Iann, Thierry, Maxime, Guillaume J., Mattias, Dimitri, Thomas B., Louis, Tiphaine, Bernardo, Lucas, Solène, Louis-André, Salomé et Louison.

Je remercie immensément ma famille, les Renault, les Seneau, les Xu et les Delmas, pour le soutien qu'ils m'ont apporté toutes ces années. Je ne remercierai jamais assez mes parents, Patrik et Shaohui, pour m'avoir offert des conditions de vie au-delà de la perfection pendant toutes mes études.

Enfin, mes remerciements les plus profonds vont à Ginger, qui m'a aidé, soutenu, secoué, motivé et j'en passe. Ensemble, nous avons traversé l'aventure de la thèse et j'ai hâte de vivre les prochaines aventures en ta compagnie.

Acknowledgements

This work was supported by European Union's Horizon 2020 research and innovation programme under grant number 951911 - AI4Media.

List of Acronyms

`CC` Control Change 29, 89

`PC` Program Change 29

ACPAS Aligned Clasical Piano Audio and Scores 64

AI Artificial Intelligence 20, 22–24

AMT Automatic Music Transcription 62, 64, 77, 126, 129

ASAP Aligned Scores and Performances 64–66, 120, 121, 126

ATEPP Automatically Transcribed Expressive Piano Performance 65, 66, 82, 126, 128

CIPI Can I Play It? 65, 66, 126

CNN Convolutional Neural Network 55–57, 75, 118, 129

CPM Classical Piano MIDI 62, 64

CPU Central Procesing Unit 78

CRNN Convolutional Recurrent Neural Network 118

CV Computer Vision 31, 49, 71

DAW Digital Audio Workstation 17, 21, 29, 31, 78, 116

DDSP Differentiable Digital Signal Processing 67, 73–76, 79, 81, 84–86, 88–98, 100–104, 106, 108–115, 123–126, 128–133

DFT Discrete Fourier Transform 34–37, 74, 77

DL Deep Learning 20, 23, 24, 73, 104, 114, 131, 132

DNN Deep Neural Network 20–24, 89, 92, 104, 114, 131, 132

FAD Fréchet Audio Distance 77, 78

FDN Feedback Delay Network 102, 103, 106, 108–111, 113

FiLM Feature-wise Linear Modulation 102

FIR Finite Impulse Response 36–38, 56, 75, 89, 102, 106, 113

GAN Generative Adversarial Network 49, 57, 58, 71–73, 116, 119, 120, 122, 126

GP GiantMIDI-Piano 64–66

GPU Graphical Processor Unit 54, 78

GRU Gated Recurrent Unit 55, 56, 95, 118, 119

IIR Infinite Impulse Response 36, 37, 75, 76

IOI Inter-Onset-Interval 31, 117, 118, 120

JTFS Joint Time-Frequency Scattering 77, 113

LS Least-Square 58, 119, 126

LSTM Long Short-Term Memory 55, 56

LTI Linear Time-Invariant 36–38, 73, 85

LTV Linear Time-Varying 36–38, 73, 89

MAESTRO MIDI and Audio Edited for Synchronous TRacks and Organization 64, 66, 72, 89, 91, 93, 94, 96, 98, 104–108, 114, 120, 121, 126

MAPS MIDI Aligned Piano Sounds 43, 62, 64, 66, 93, 126

MIDI Musical Instrument Digital Interface 24, 27, 29–32, 46, 60, 62, 64–67, 70–73, 75, 79–82, 84–87, 89, 91, 97, 98, 100, 101, 103, 104, 114–118, 120, 121, 123–126, 128, 129, 131–133

MIR Music Information Retrieval 76

MLP Multi-Layer Perceptron 92, 104, 118

MOS Mean Opinion Score 78, 83, 95, 121

MSS Multi-Scale Spectral 73, 76, 77, 89, 93, 94, 101, 103, 104, 108, 113

MUSHRA MUltiple Stimuli with Hidden Reference and Anchor 79, 83

NAS Neural Audio Synthesis 71, 112, 114, 126, 129

NLP Natural Language Processing 31, 49

NSF Neural Source Filter 72, 73, 91

RAM Random Access Memory 78

ReLU Rectifier Linear Unit 54

RNN Recurrent Neural Network 55, 56, 75, 81, 87, 88, 103, 125, 129

SMD Saarland Music Data 64

STFT Short-Term Fourier Transform 34–38, 74, 89

TTS Text-to-Speech 71–73, 78, 81, 91, 96, 103, 106, 108, 111, 124

VAE Variational Auto-Encoder 72, 82, 84

VQ Vector Quantized 82

List of Symbols

B Inharmonicity coefficient

C Context vector dimension

$\mathcal{C}$ Context Network

D Dimension

$\mathcal{D}$ Discriminator

$\mathbb{D}$ Distance

$\mathbb{E}$ Expectancy

F Sampling rate

I Number of recording environments

J Jacobian matrix

K Number of partials

$\mathcal{L}$ Loss function

$\mathcal{M}$ Monophonic Network

N Number of notes

P Polyphony

$\mathcal{P}$ Data distribution

Q Filter order

$\mathcal{R}$ Performance Rendering Model

$\mathcal{S}$ Audio Synthesis Model

T Number of time frames

W Model weights

X Note-wise symbolic music

Z Embedding dimension

a Global amplitude

b Inharmonicity modifier

$\mathbf{c}$ Context vector

d Single note duration

f Frequency

$\mathbf{h}$ Partial amplitude distribution

i Recording environment index

j Imaginary number

k Partial index

m Discrete Fourier Transform size

n Note index

o Inter-Onset Interval

p Note pitch OR Polyphony index

t Time frame index

$\mathbf{u}$ White noise spectrum

v Note velocity

w Window function

x Frame-wise symbolic music

y Audio signal

$\mathbf{z}$ Embedding

α Linear asymptote coefficient

β Linear asymptote bias

δ Deviation

η Noise filter magnitudes

κ Optimization step

λ Loss weighting

ρ Octave tuning

σ Activation function

τ Sub-time frame index

Φ Instantaneous phase

Contents

List of Acronyms **7**

List of Symbols **10**

1 Introduction **16**
1.1 Music Generation: a Multi-Stage Process 16
1.1.1 Musical Composition 17
1.1.2 Musical Performance 17
1.1.3 Sound Synthesis 18
1.1.4 Sound Processing 18
1.1.5 Music Listening 19
1.2 Automating and Assisting Music Production Frameworks 20
1.2.1 Simulating and Automating Individual Stages 20
1.2.2 Multi-layered Automation 21
1.2.3 Adoption of the Tools by Practitioners 22
1.3 Research Objectives and Contributions 23
1.3.1 Main Research Objectives 23
1.3.2 Contributions 24
1.3.3 Thesis Structure 25

2 Technical Background for Music Processing **27**
2.1 Symbolic Music Processing 27
2.1.1 The Music Score 27
2.1.2 The MIDI Protocol 29
2.1.3 Efficient Representations of Symbolic Music 29
2.2 Audio Digital Signal Processing basics 33
2.2.1 Waveform representation 33
2.2.2 Time-Frequency representation 34
2.2.3 Linear Filtering 36
2.2.4 The Spectral Modeling Paradigm 38
2.3 Piano Mechanisms 40
2.3.1 A Brief History of Piano Craftsmanship 40

2.3.2 A Monophonic String Model . . . 41
2.3.3 Soundboard . . . 44
2.3.4 Polyphony . . . 45
2.4 Deep Learning for Music Processing . . . 49
2.4.1 Data-driven Optimization with Gradient Descent . . . 49
2.4.2 Neural Networks . . . 54
2.4.3 Generative Adversarial Networks . . . 57
2.4.4 Domain-knowledge inclusion in Deep Neural Networks . . . 59
2.5 In short . . . 60

3 State-of-the-Art 61
3.1 Piano Performance Datasets . . . 61
3.1.1 Synthetic MIDI Performance Datasets . . . 62
3.1.2 MIDI Recorded Performance Datasets . . . 62
3.1.3 MIDI Transcribed Performance Datasets . . . 64
3.1.4 Other Piano Datasets . . . 65
3.2 Polyphonic Instrument Audio Synthesis . . . 67
3.2.1 Parametric models . . . 67
3.2.2 Data-driven models . . . 70
3.2.3 Differentiable Digital Signal Processing . . . 73
3.2.4 Evaluating Sound Synthesis . . . 77
3.3 Performance Rendering . . . 80
3.3.1 Traditional Approaches to Performance Rendering . . . 81
3.3.2 Neural Approaches to Performance Rendering . . . 81
3.3.3 Evaluating Expressivity . . . 82
3.4 In short . . . 84

4 DDSP-Piano: a Neural Piano Synthesizer informed by Instrument Knowledge 85
4.1 First Iteration of DDSP-Piano . . . 85
4.1.1 Model architecture . . . 85
4.1.2 Model Training . . . 89
4.1.3 Evaluation . . . 90
4.1.4 Qualitative Results: Comparison with Known Behaviors . . . 97
4.2 Improving DDSP-Piano . . . 101
4.2.1 Architectural Changes . . . 101
4.2.2 Revised Training Procedure . . . 103
4.2.3 Evaluation . . . 106
4.3 Discussion . . . 112
4.3.1 Some Takeaway Lessons . . . 112
4.3.2 Future works . . . 113
4.4 In short . . . 114

5 Piano Performance Rendering from Unpaired Data 115
5.1 Motivation . . . 115
5.2 Render Performances in the Symbolic Modality . . . 117
5.2.1 Models Architecture . . . 117
5.2.2 Training Strategy . . . 119
5.2.3 Results and Analysis . . . 120
5.3 Cross-Modal Extension . . . 123
5.3.1 From note-wise to frame-wise Conditioning . . . 124
5.3.2 Training Setup . . . 125
5.3.3 Qualitative Analysis . . . 126
5.4 Discussion . . . 129
5.5 In short . . . 130

6 Conclusion 131

Chapter 1

Introduction

1.1 Music Generation: a Multi-Stage Process

Creating music out of the organization of sounds is an artistic endeavor that involves multiple stages. The listening experiences can be extremely diverse, from lighthearted nursery rhymes to shattering contemporary symphonies through contemplative soundscapes accompanying pictures. From an imagined musical idea to its emotional reaction provoked by the perceived sound, the musical experience can be viewed, in a sense, as a message conveyed from a musician to a listener, following the communication theory of Shannon (1948). This message can take multiple forms and be handled by different agents, each of whom can transform the musical idea at various levels, intentionally or not. Like in any art form, such intermediary transformations are the siege of techniques derived from a consensus refined by cultural practice, or shaken by groundbreaking and innovative technologies.

The full process, illustrated in Figure 1.1, is an extended version of the one proposed by Oore et al. (2020). The remainder of this Section will briefly describe each stage composing the full music generation process, from their objective, the experts associated with them, and the challenges they arise.

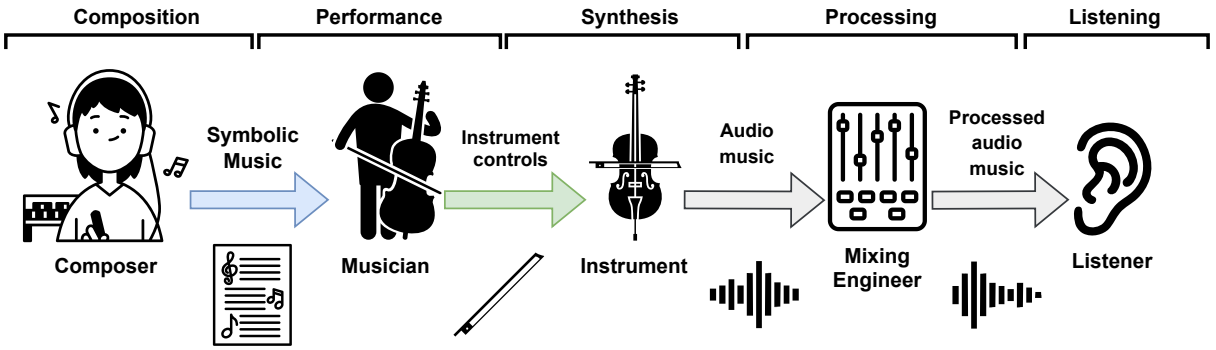


Figure 1.1: The music generation multi-stage process. Each role can be endorsed by the same or different individuals throughout the process.

1.1.1 Musical Composition

Musical composition consists in the forethought, the planning of what intended music will be produced. It encompasses the selection of notes, lyrics and/or sound characteristics that will be played and how they will be played. Such planning can be transmitted to provide a road-map, which is especially useful for synchronizing multiple individuals towards a common purpose. The communication of the musical planning can be written down into symbols, through a music sheet, or transmitted orally by imitation or telling a chord progression for example.

In the case of writing a musical piece, this stage is traditionally attributed to a **composer** that follows a global directive from a command, a style exercise or a personal message and/or story to convey. The creative planning can be supported by music theory knowledge, such as the harmonic rules of classical western music. Nowadays, most Digital Audio Workstations (DAWs) and music notation software provide enhanced experiences of the composition process by synthesizing online audio feedback of the programmed musical sequences.

The compositions can be planned well ahead of their realisation, as classical pieces from centuries ago are still played to this day. However, the musical planning can also be short-termed, especially in the context of improvisation as, for example, jazz soloists develop musical phrases for what is perceived as "on the spot" based on the immediate context, but often are aware of the incoming musical structure and repurpose musical licks and knowledge they have honed.

1.1.2 Musical Performance

Score and audio music share a similar relationship to that of text and speech: both score and text are a symbolic and compact representation of a particular kind of sound (being music and speech respectively), that can be restored by a human using appropriate tools (e.g. a musical instrument or the voice). However, the information contained in the symbolic representation may not fully convey the intended sound imagined by the writer or composer: the interpreter thus chooses to produce one of the possible sounds with respect to the written text or score.

Palmer (1997) formalized the cognitive process of performing a music piece: the interpretation is the physical execution of the **musician**'s understanding of the composition and its structure, eventually guided by a **conductor** and style-specific actions and constraints. The execution is manifested through the *controls* inputted to the musician's instrument. Such controls are exclusive to the instrument at hand, which can be keyboard playing, wind blowing, vocal track, or drum hitting for traditional instrument playing. But this also includes **sound designers** and **music producer** that control modern instruments through parameterizing hardware and software synthesizers.

However, there can also be differences between the interpretation intended by the performer and what they actually do during the performance, due to mistakes or limited mastery of their instrument. To gain such mastery, musicians often practice their instruments for years before being able to faithfully convert their musical intents into appropriate instrumental controls.

1.1.3 Sound Synthesis

Upon adequate controls inputted by the musician, the **instrument** will produce a sound, leveraging a certain phenomenon. Traditional instruments make use of physical vibratory phenomenons, such as string vibration (in stringed instruments such as guitars, violins and harps), air flows for the voice, wind and brass instruments, and resonating objects (substantially drums and other percussions). Then, in the 20th century, analog synthesizers and even magnetic tapes offered new sonic possibilities by manipulating electrical signals rather than acoustic signals. They were soon followed by digital synthesizers and software that leveraged the new capabilities emerging from computers.

Even if it is a non-human object that produces the musical sound (with the exception of the voice), knowledge and craftsmanship for building such objects have been long developed by **luthiers**, **manufacturers** (both for acoustic and electrical instruments) and **programmers** to refine the instrument sound or innovate on new interfaces and instruments.

1.1.4 Sound Processing

The sound produced by the source instrument is systematically altered by different phenomenons before being perceived by the listener. These alterations can be induced by the acoustic environment (namely room acoustics) or manually applied in the form of signal processing. Indeed, as sound propagates in the open air, the environment acoustics can alter it through echoes, filtering and reverb until reaching the listener. Such environmental settings may be designed by **architectural acoustic engineers** to conceive "pleasing" sounding rooms, buildings or landscapes.

On the other hand, musical sounds can also be recorded and processed by a **mixing engineer** that try to shape multiple individual tracks together to create a coherent and harmonious global mix. They chose to apply different technical and creative transformations to follow an artistic vision, such as selective filtering, dynamic processing and choice of recording devices (microphones, and recorders). A **mastering engineer** can

also be involved in the process to standardize the final mix which can be then stored and distributed through streaming platforms mostly at this day and age. Afterwards, offline listening can also be altered by the sound emitting device (by loudspeakers, earphones).

Live mix engineers are located in between these two conditions as they also use sound capturing, mixing and emitting devices for enhancing the live experience, while accounting for the natural acoustics.

Professional actors in this stage leverage their technical knowledge, ear training and artistic vision to enhance and sublime the individual raw sounds in order to deliver a coherent and harmonious mix to the listener.

1.1.5 Music Listening

The music is ultimately perceived and processed by the listener. They can perceive it passively, while focusing on another medium such as the pictures while watching a movie. Passive listening does not mean that the music is completely ignored, as it can enhance or perturb the main experience. Active listening on the other hand aims to assess and appreciate the quality of the overall music or one of its characteristics: for example in music competitions, **jury members** can evaluate the interpretations of a common piece proposed by different performers.

However, active listeners can also be **any agent from the previous stages**. Indeed, the final audio is the ultimate medium for appreciating the music and every stage along the process influences the final result. Knowing how their actions influence the final results and how to adjust them from such feedback is crucial for honing their expertise and use it to achieve an artistic vision.

Also, although hearing is the main sense with which music is experienced, research on its haptic perception (Richards, 2023) and links with visual cues (Platz and Kopiez, 2012) suggest that music appreciation can be a multi-sensory experience.

Section Summary - Music Generation

The musical generation process involves multiple intermediary steps that ultimately leads to the transmission of a sound idea from a composer to a listener. Along this transmission, several agents and elements can alter the initial musical idea and imbue it with their own artistic effort. Namely, between the composer and the listener, performers can interpret the music score, the instrument produces sounds following the controls inputted by the performer and the sound is modified by the environment and/or the effects selected by a mix engineer. All these stages involves appropriate tools and knowledge in order to achieve a final result reflecting the intentions of each actor.

1.2 Automating and Assisting Music Production Frameworks

As seen previously, music creation is both: an art form and an engineering endeavor, as eclectic skills and knowledge can be leveraged to achieve an artistic vision. Learning and applying such knowledge can be tedious, time-consuming and/or resource-intensive, with dedicated professionals honing and living off such expertise that they have developed. Yet, having to apply such technical expertise can hinder the creative process of both amateur and professional artists. For example, amateurs often lack access or understanding of professional tools, which limits the sound quality of their creations. As for experts, before tackling the creative aspect of their work, they have to apply non-creative but necessary operations such as maintenance and tuning of instruments, cleaning audio recordings, staging of mixer gains, or calibration of live equipment.

Recently, technological progress has led to the democratization and the streamlining of expert tools, making it possible to produce professional-sounding music from a personal computer, instead of relying on expensive recording studios. Especially in the era of Artificial Intelligence (AI) and Deep Learning (DL), more complex and high-level operations can be automated, alleviating manual actions from the user and simplifying the music creation process. However, as with other media impacted by the development of AI, there seems to be a reluctance among a large portion of users to fully adopt and/or accept these new tools.

The following will present a quick portfolio of musical tasks that automated systems and DL models can accomplish. Numerous works have used them for simulating or automating individual stages of the presented music generation process, but others have also tackled multiple stages at once for higher-level manipulation. These works are more extensively discussed in the survey of Ji et al. (2020) and Bazin (2023). Then, we will delve into the reasons why music practitioners may want to adopt or not such systems.

1.2.1 Simulating and Automating Individual Stages

Each stage of the music creation process is characterized by specific properties introduced by the composers, musicians, instruments, or effects that an automated system should imitate in order to sound realistic and be usable.

Music composition is often entailed to abstract concepts specific to the art, such as harmony and rhythm. Western music theories offer frameworks and rules for explaining and building new compositions: such rules can be formalized and integrated in the form of algorithmic processes, which fostered the field of algorithmic music composition from the 80s onward (Nierhaus, 2009). In continuity, Deep Neural Networks (DNNs) have been applied for accompanying the composition process with various depths of involvement (Ji et al., 2023; Hernandez-Olivan and Beltrán, 2023): simple melody generation, harmonization from a melody prompt, re-orchestration from a composition draft or even full multi-track generation from scratch, inpainting or a textual prompt.

However, writing music in the symbolic domain reduces its complexity into a compact form that simplifies certain musical aspects to fit understandable symbols (such as

crescendo and slur markings) and music concepts (such as the music grid, or measures). As such, direct rendition of a music score lacks the interpretative and subtle nuances introduced by musicians, through micro and macro displacements of note timings with regards to the metrical grid, and the variety of loudness levels. **Performance rendering** systems that can reproduce such a complex and artistic behavior can find its usage in assisting composers for obtaining musical renditions of their pieces. As such, certain DAWs and virtual instruments (such as drum samplers) can propose a "humanize" or "swing" feature to render symbolic music with stochastic micro-displacements from the temporal and velocity grids.

Instruments and single **effect simulations** may be the most widespread usage of automated system in music production: from plates and springs used as proxy for room reverberation in the first half of the 20th century where, to guitar amplifiers emulated in today's personal computers with DNNs, through analog synthesizer presets imitating organs and string ensemble. Emulations of physical instruments and analog music hardware allows for the democratization of such tools to a wider user group that can not access to the real versions. Instead, they can be integrated as plug-ins running in real-time in most DAWs, offering ease of usage and combination. However, faithfully reproducing all the nuances of the real instrument or effect is a challenging endeavor for the manufacturer, often restrained by the limited computing capabilities of computers and thus, has to introduce modeling approximations in order to maintain the real-time compatibility. Several brands have successfully leveraged the modeling capabilities of DNNs into plugins, but they have also been used for the development of effects and instruments, notably for with timbre transfer[1][2]. Conversely, neural networks have also been employed for the inverse task of **sound matching**, which is finding the parameters in order for a given model to reproduce a (single) target sound (Esling et al., 2020; Han et al., 2023; Masuda and Saito, 2023).

1.2.2 Multi-layered Automation

The sub-stages of music generation are not hermetic to one another, as task-specific agents can influence each other in order to elaborate on the larger direction of the music being produced. In that regard, automatic systems have also been developed to deal with several stages at the same time, especially with DNNs that can model more complex and non-linear phenomenons, eventually involving multiple modalities.

In the manner of musicians improvising musical pieces, **performance generation** systems can produce symbolic performances with expressive qualities without being fed with an input composition (Huang et al., 2019; Oore et al., 2020). Still, they can also be steered to complete an input excerpt (composed or improvised) and allow for co-composition and co-improvisation experiences (Bazin, 2023). Concurrently, **performance synthesis** models, such as the one proposed by Wang and Yang (2019), can directly produce the expressive audio rendition of a score. They merge the musician-instrument duo into a single entity to model: since musicians do not express their intents as controls in themselves,

[1] https://magenta.tensorflow.org/ddsp-vst
[2] https://neutone.ai/

but rather aim at a target sound emitted by their instruments, such approach bypasses the need to predict instrument controls and directly outputs audio performances. Other works can also handle the task in a hierarchical approach by stacking modules for each sub-task, and allow for better manipulation of intermediary predictions (Wu et al., 2022b).

Hereafter, automating the mixing and mastering steps represents an opportunity for composers and musicians to get acceptable sounding quality for their music without learning and manually applying the underlying technicality. Such tools can help streamline and democratize music creation by lowering the barrier to accessing high-quality sound. Various works have proposed such systems that apply a cascade of effects for processing the sound of one or several audio tracks, in order to match certain audio quality criteria (De Man et al., 2017) and/or a reference mix (Colonel and Reiss, 2021; Steinmetz et al., 2022).

Finally, with the ever-increasing capabilities of DNNs, more recent approaches address the full **audio music generation** task by producing complete musical excerpts, embodying realistic choices and characteristics from all stages. Such large and hierarchical models can generate coherent musical audio, by jointly selecting and interpreting notes, generating and singing lyrics, while applying relevant and textured sound design and all eventually in a multi-track setting. They are mostly conditioned on high-level control inputs, such as lyrics, artist and genre styles (Dhariwal et al., 2020) or free-form textual prompts (Agostinelli et al., 2023; Copet et al., 2023). They notably offer a solution for non-musician content creators to get relevant accompanying music for their main content, without relying on copyright-protected music (Frid et al., 2020).

1.2.3 Adoption of the Tools by Practitioners

Sai Vanka et al. (2023) have interviewed different user groups regarding their adoption of automatic AI-based mixing and mastering systems. They have noticed that amateurs were more inclined to use such systems as they compensate for their lack of expertise in the mixing and mastering stages and help achieve decent sound quality, without extensive effort. On the other hand, most professional mixing engineers are skeptical with respect to the ability of such systems to fully reproduce the emotional intent of a human-made mix. Therefore, they want to still be able to express their own artistic vision, through intuitive and/or familiar controls, in order to adopt these systems into their established framework. In this manner, professional engineers would rather use systems allowing adjustments and co-creativity in a collaborative setting, rather than an unsteerable "single push button" replacing their skills and knowledge developed for years.

General remarks from these interviews, specific to the mixing stage, can be transferred to the other stages of the music generation process. All agents in the process have to accomplish domain-specific knowledge and workload in order to manifest their creative intent, such as harmonization, arrangement, instrument practice, or mixing. Automated systems can alleviate such workload and help streamline the creative process, but only up to a certain threshold where the user may feel excluded from the decision-making. Indeed, music being an art form, personal expression is key to the artist making it, and to an extent, to the audience perceiving it. However, this threshold is user-specific and

often correlates with the expertise they already have on the task: amateurs can embrace high-level AI-automated tools more easily than experts having their own framework and technical preferences. As such, automated music tools should simplify and/or enhance the creative process, while accounting for the target users' framework and needs for control and customization.

Section Summary - Automating and Assisting Music Production Frameworks

Since music production often requires applying different technical knowledge at multiple stages, several automated systems have been proposed to alleviate such workload and streamline the realization of one's musical intent. Such systems can be designed to handle specific tasks, such as single-effect simulation, instrument synthesis, or melody harmonization. However, with the advent of DL and the increase in computational resources, more complex and sophisticated systems can tackle multiple stages at once, from automated multi-track mixing and generation of performances to full audio music generation from high-level descriptors. Yet, while these advanced systems allow for the democratization of music creation to non-expert users, their adoption by expert music practitioners trails as the systems offer limited customization options for re-introducing individual knowledge and preferences. Instead, expert users would advocate for more transparent intelligent systems with fine-grained controls to allow interactive co-creation through personal expression accelerated by the capabilities of the models.

1.3 Research Objectives and Contributions

1.3.1 Main Research Objectives

In the context of music generation, the scope of this thesis is to explore novel automated methods that improve the creative process in the established music production framework. The focus is set on **piano music**, with an emphasis towards the **audio synthesis** and **performance rendering** stages.

Both tasks are involved in the physical realisation of a musical composition. Namely, *performance rendering* takes a symbolic composition as input and makes an expressive rendition out of it, as if a musician performed it. Then, *instrument audio synthesis* transforms controls inputted by a musician and produces an audio signal, in the manner of an acoustic musical instrument.

Both the musician and the instrument introduce their specific factors that contribute to the overall sound quality. Reproducing such **realism**, in terms of fidelity and expressivity, is a challenging endeavor in order for automated systems to be worthy of usage in the musical creation process. To this end, DNNs seem eligible for imitating such qualities, as they exhibit impressive capabilities for modeling complex and non-linear relationships between different domains and modalities, directly from data.

One of the key factors that will be considered during this manuscript is the ease with

which the DNN-based methods can be integrated into the workflow of music practitioners. In the context of the **AI4Media** European project (Tsalakanidou et al., 2021), which has fully funded this thesis, one of the objectives is the advance of ethical and trustworthy AI serving the sectors of various media, including music.

In sum, the main research questions surrounding this thesis are:

- How to achieve intelligent musical systems that can reproduce the realistic features introduced by traditional agents and tools?
- Where to integrate such systems into the global process of musical creation?
- How to design the systems in order for them to be controllable, and ultimately adopted, by users?

1.3.2 Contributions

Following the research context and in an attempt to address the interrogations, multiple contributions have been proposed during this thesis.

The main contribution is the development of **DDSP-Piano**, a neural piano synthesizer that produces realistic piano audio from polyphonic symbolic controls in Musical Instrument Digital Interface (MIDI). The model is built on top of a hybrid framework combining signal processing and DL tools. Leveraging the interpretability and modularity of this framework, the approach further includes high-level knowledge of the instrument to obtain a lightweight, interpretable, and realistic-sounding piano synthesizer. Consequently, the approach can be evaluated in a targeted manner, to examine the behaviors it has learned and put them into relation with known properties of the instrument.

The second contribution is the proposal of an **unsupervised piano performance rendering model** in a **low-informed setting**. In the continuity of performance rendering models, it enhances symbolic music compositions with expressive qualities, still in the symbolic modality. However, the model is easier to integrate into modern composition frameworks as it does not rely on markings specific to music scores, but only on simple MIDI data. Furthermore, thanks to adversarial training, it can learn to imitate expressive qualities in real symbolic interpretations without seeing how they differ from a source score. Finally, some preliminary works have been done to extent the model to also learn expressive features from audio recordings: by bridging the symbolic and audio modalities with the DDSP-Piano synthesizer, this cross-modal extension could render expressive audio performances while still providing the symbolic transcripts.

Peer-Reviewed Publications

The work conducted during this thesis has led to the following list of peer-reviewed publications:

- Renault et al. (2022) - **Best Paper Award** – Lenny Renault, Rémi Mignot, Axel Roebel, “Differentiable Piano Model for MIDI-to-Audio Performance Synthesis”, Proceedings of the 25th International Conference on Digital Audio Effects (DAFx20in22), Vienna - Austria, (September 2022).

- Renault et al. (2023a) – Lenny Renault, Rémi Mignot, Axel Roebel, "DDSP-Piano: a Neural Sound Synthesizer Informed by Instrument Knowledge", Journal of the Audio Engineering Society, vol. 71, no. 9, pp. 552-565, (September 2023).

- Renault et al. (2023b) (Demo & Late-Breaking Results track) – Lenny Renault, Rémi Mignot, Axel Roebel, "Expressive Piano Performance Rendering from Unpaired Data", Proceedings of the 26th International Conference on Digital Audio Effects (DAFx23), Copenhagen - Denmark, (September 2023).

Oral presentations were delivered for the conference papers (Renault et al., 2022, 2023b) during their respective conferences.

Notably, the first published work (Renault et al., 2022) was given the Best Paper Award at the 25th International Conference on Digital Audio Effects (DAFx20in22), and further extended into a journal article (Renault et al., 2023a) in the *Special Issue on New Trends in Audio Effects II* of the Journal of the Audio Engineering Society.

Presentations in Seminars

Presentations of the works accomplished during this thesis was also given in the following scientific gatherings:

- Deezer Research 10th aniversary DaRe Seminar, Paris - France, (14th April 2022).

- GdR ISIS "Méthode en traitement du signal pour l'écoute artificielle", Paris - France, (12th May 2022).

- AI4Media Consortium Meetings: Pisa - Italy, (3rd October 2023); Thessaloniki - Greece, (21st October 2022).

- Doctoral Day: STMS, Paris - France, (2nd December 2021); STMS Joint with AI4DM (QMUL), Paris - France, (13th February 2023).

- Journées Jeunes Chercheur·es·s en Audition, Acoustique musical et Signal Audio (JJCAAS), Paris - France, (5-7 April 2023).

1.3.3 Thesis Structure

After this introductory Chapter, and until the concluding Chapter 6, this manuscript is organized in a classical bottleneck fashion, by first presenting broad scientific concepts and methods, from which the specific tasks literature can then be built, paving the scope and motivating the contributions of this thesis.

Chapter 2 will lay down the scientific background necessary for understanding the tools and methods mainly employed through this thesis. Namely, the multiple modalities and the multi-disciplinary approaches for handling piano music call for concise presentations of symbolic music processing, audio signal processing, piano instrument crafting, and deep learning through data-driven optimization.

Chapter 3 will present the state-of-the-art for each task under study. In particular, since data-driven methods will be used in this thesis, a survey on piano music datasets will be conducted. Then, previous works for conceiving and evaluating piano audio synthesis and performance rendering models are reported, regrouped by their different methodologies. A particular emphasis will be made towards deep learning-based approaches in order to better position the thesis contributions.

In Chapter 4, the piano audio synthesis task is addressed with the proposed DDSP-Piano, a hybrid synthesizer that, in the continuity of DDSP, incorporates high-level modeling knowledge of the instrument into a deep learning framework. With dedicated modules handling specificities of the instrument sound (such as polyphony, partials inharmonicity and detuning), the model is described, analyzed and evaluated against other synthesis methods, yielding realistic sounding piano performances from polyphonic symbolic inputs. Notably, with significantly less parameters, it surpasses a pure deep learning-based benchmark. In continuity, the targeted analysis provides several paths for improvement, motivating a second iteration of the model that is presented and partially evaluated.

Chapter 5 introduces the work conducted on piano performance rendering, by first highlighting the usual training needs of data-driven methods for the task, being gathering symbolic performances aligned with compositions with score markings. This motivates the design of an unsupervised method, inspired by the domain transfer literature, to make expressive renditions out of symbolic compositions, using an adversarial training on unaligned datasets of compositions and performances. The subjective evaluation suggests that the realism of the rendered performances can be improved in order to match those from a supervised benchmark and real musicians, but the method employed still opens up interesting research directions. In particular, a cross-modal extension is introduced by incorporating DDSP-Piano into the approach, in order to also include raw audio data into the training scheme.

Finally, Chapter 6 concludes the manuscript, summarizing its main content and contributions while also providing several future research directions.

Readers are encouraged to visit the webpage[3] accompanying this manuscript in order to listen to audio examples of the conducted work, and also to some incomplete side experiments.

[3] http://renault.gitlab-pages.ircam.fr/thesis-support

Chapter 2

Technical Background for Music Processing

This chapter aims to introduce the necessary scientific knowledge and tools that are relevant for efficiently building performance rendering and musical audio synthesis systems. It notably introduces the mathematical notations that will be preserved throughout the manuscript. While the covered topics may seem eclectic, the main purpose of this chapter is to lay down the foundations, to be later on connected in the following chapters covering the state-of-the-art and the thesis contributions.

First, the background is set for manipulating the two main modalities of music: Section 2.1 presents the different symbolic representations of western music, while Section 2.2 introduces the mathematical theories and tools for understanding and reproducing audio signals. Then, as it is the main object of study in this thesis, the piano and its mechanisms are laid down in Section 2.3. For tackling the tasks, the methods employed in this thesis rely on deep learning, the background of which is set out in Section 2.4. Section 2.5, finally summarizes and concludes the chapter.

2.1 Symbolic Music Processing

Symbolic encoding of music aims to reduce the high-dimensional music signal into a combination of symbols, embodying concepts understandable by practitioners. This section will cover different symbolic representations of western music that are commonly used, namely the music score and the MIDI protocol.

2.1.1 The Music Score

Keeping a written trace of musical pieces is an ancestral endeavor, with music-annotated tablets dating back over a millennium BC (West, 1994). Nowadays, the standard written music notation is the **sheet music**, or **music score**, which is heavily entailed to the western classical music culture. The piano sheet music illustrated in Figure 2.1 shows notations and symbols embodying multiple concepts of western musical theories.

Notably, the four main axes chosen for describing music are time (by reading the sheet from left to right), frequency (low to high pitch notes are arranged from bottom to high

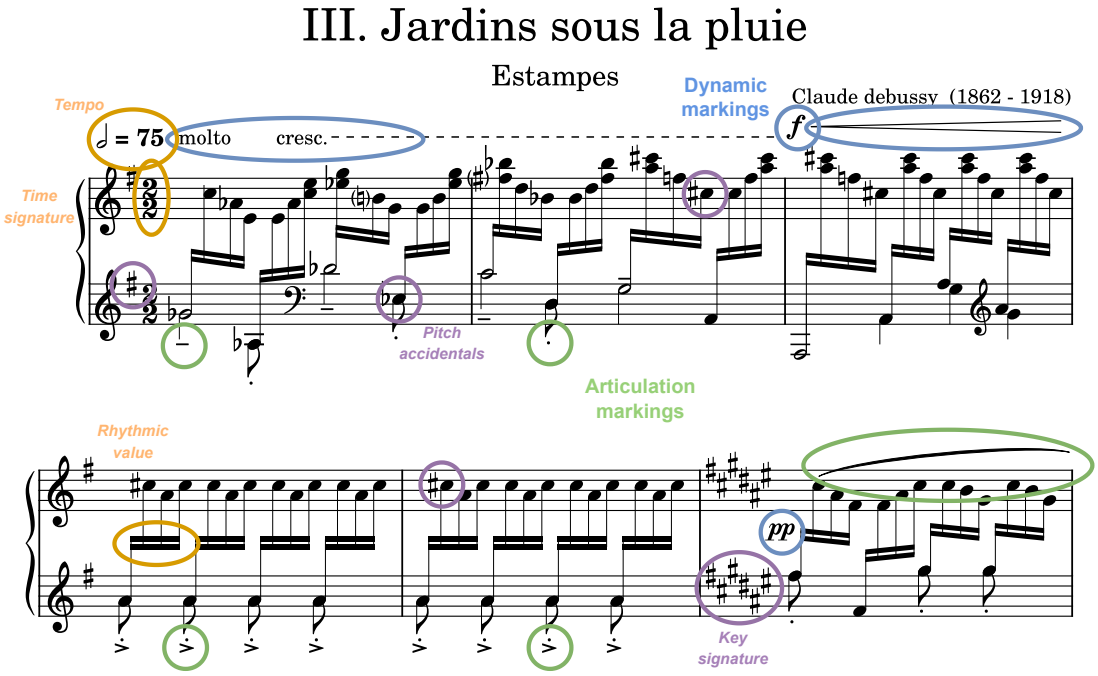


Figure 2.1: Extract of a piano music sheet. Notations related the encoding of time are encircled in orange, while pitch-related notations are in purple and loudness-related ones are in blue. Green markings can influence both time and timbre features.

along the staff) loudness, and timbre (with the choice of instruments). Each dimension can be encoded at multiple granularities, ranging from low-level note-wise markings to high-level markings describing the structural understanding of the piece. For example, a single note duration can be retrieved from its rhythmic value with regards to the tempo, but the time signatures and the segmentation of the piece into measures indicate an organization of the temporal piece structure, through a beat value and a "rhythmic feel". Similarly, note pitches are read from their vertical position along the staff, but the pitch accidentals and key signature markings put them in the context of a musical scale governing the section. Dynamic markings specify the global loudness and its evolution in a section, but note-specific indications can be made with articulation markings. Other markings (through symbols or text), which are not necessarily shown in the given example, can further specify playing techniques (such as ornaments and pedal marks), aggregate notes into similar groups (with chords and slurs), or make reading easier (with octave and repetition signs).

Digital editing and storage of a music score is usually done through the XML protocol, with the MusicXML file extension (Good, 2001).

Other writable symbolic representations encompass the piano roll (to be played by an automatic player piano), the lead sheet or chord chart (simply transcribing a melody line and the accompanying chord progression) and guitar tablatures (more adapted for guitar pieces, with fingering indications).

2.1.2 The MIDI Protocol

During the advent of electronic-based instrument manufacturing, users felt the lack of a standardised means to synchronize their musical devices as each synthesizer company had developed their own standards. To mitigate this issue, the communication protocol MIDI (Hoffmann, 2004) was elaborated in the early 80s (Smith and Wood, 1981) mainly by manufacturers Roland and Sequential Circuits. Simple yet flexible, it is now the main standard for communication between electronic instruments, audio devices and computers.

MIDI allows the transmission of 7-bit messages in real-time from a MIDI-sending device (such as keyboard, or sequencer) to a MIDI-receiving one (synthesizer, DAW, digital effects, ...). Note event messages are used for conveying sequences of notes: when a note is initiated by the sending device, a `note on` event is transferred, parameterized by the note number (from 0 to 127) and its velocity (also encoded from 0 to 127). A `note off` event is sent when the note previously initiated by a `note on` event is finished. It is parameterized by the same note number and velocity as the associated `note on` event. Note numbers quantize the note frequencies based on the equal temperament system, which spaces every semi-tone with a constant frequency ratio. By using the A3 note at 440Hz as reference for the note number 69, a note with frequency f_0 can be converted into its MIDI pitch x^{pitch} as follow:

$$x^{\text{pitch}} = 69 + \text{nint}(12 \log_2(\frac{f_0}{440})), \tag{2.1}$$

with nint the rounding to nearest integer operation. Inversely, the equal-temperament frequency $f_{0,\text{ET}}$ can be retrieved from the MIDI pitch x^{pitch} using the formula:

$$f_{0,\text{ET}}(x^{\text{pitch}}) = 440 \times 2^{\frac{1}{12}(x^{\text{pitch}}-69)}. \tag{2.2}$$

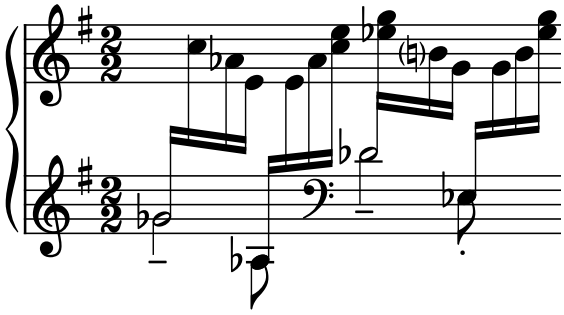
The Control Change (`CC`) messages are sent for updating auxiliary control parameters of the receiving device. They are parameterized by a control number and a control value, each from 0 to 127. Most commonly used controls are the modulation control, and most importantly for this thesis, the piano pedal controls. Pitch bends on the other hand are not encoded through `CC` but have their own dedicated message type: they allow for filling the frequency spectrum in between equal temperament frequencies.

Program Change (`PC`) messages allow for the receiving device to change its configuration from one of its 128 presets. All messages can be assigned to one or several of the 16 available MIDI channels, which is useful when connecting several devices together (in chain or parallel) and sending dedicated messages to each of them.

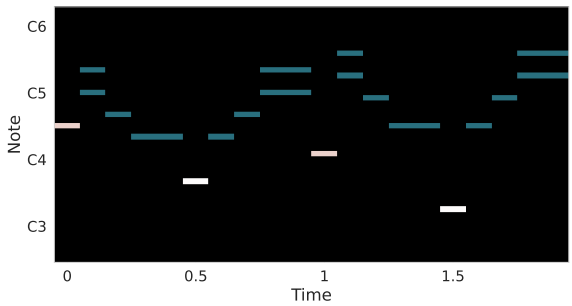
A sequence of MIDI messages can be stored to and read from a MIDI file, recognizable by their `.mid` extension and widely used thanks to their compact size. On top of their parameters, the messages are stored with timestamps and can be organized into channels (one per instrument for example). In particular, arrangements can be efficiently stored with tempo, time signatures and keys changes using messages specific to the file format.

2.1.3 Efficient Representations of Symbolic Music

While music scores were designed to be read by musicians and the MIDI protocol by electronic devices in real-time environments, computers can more efficiently process the



(a) Score.



(b) Active velocities piano roll.

event(value)
velocity(84)
note_on(66)
time_shift(31)
note_off(66)
velocity(45)
note_on(72)
time_shift(32)
note_off(72)
...

(c) MIDI-like tokenization.

pitch	IOI	duration	vel
66	0.0	0.125	84
72	0.125	0.125	45
76	0.0	0.125	45
68	0.125	0.125	45
64	0.125	0.125	45
56	0.125	0.125	90
56	0.0	0.125	96
64	0.125	0.125	45
...	...	...	...

(d) Note-wise encoding.

Figure 2.2: Visualizations of different symbolic representations for the same piece of music.

contained information with modified encodings that exhibit different relationships embedded in the composition or performance. Different representations of the same music piece are illustrated in Figure 2.2. More extensive explanations and usages can be found in the survey conducted by Ji et al. (2023), along with other encodings not presented in this manuscript.

The closest encoding to the raw MIDI protocol is the **MIDI-like tokenization** first introduced by Oore et al. (2020). The data is represented as a sequence of one-hot encoded tokens in a vocabulary of events, covering MIDI messages with their possible values that are relevant for a given task. For example, 128 tokens in the vocabulary can be allocated for indicating a change of property of a note number, followed by one of 2 tokens whether it is a note-on or note-off event and one of 128 tokens for the velocity. Tokens can also be assigned for indicating a change of instrument for multi-instrument tasks (Hawthorne et al., 2022), and time-shift tokens allow to move forward in time and specify the duration separating two events. While the absolute time representation is convenient for encoding the fine time-granularity of performances, score-related tasks may prefer a more musically meaningful encoding of time by using bar and position tokens, as proposed by Huang and Yang (2020).

This representation of a stream of tokens can create very long sequences and tends to

spread note information apart, as a long sustained note can have its note-on and note-off tokens separated by lots of tokens of simultaneous short notes. Also, note insertion or deletion can require the modification of some time-shift events in order to preserve the timestamps of future events. Overall, the stream of tokens representation can be difficult to manipulate while maintaining strict coherence, but it benefits tasks and system architectures relying on discrete representations and has thus found success with models mostly inspired by Natural Language Processing (NLP), capable of modeling long-term dependencies (Oore et al., 2020; Huang and Yang, 2020; Gardner et al., 2022; Hawthorne et al., 2022).

In a compound words manner (Hsiao et al., 2021), the **note-wise encoding** increases the abstraction level by grouping events into notes with properties. The most commonly used properties for a single note are its pitch, onset time, duration and velocity. But there can be differences between the properties of grid-locked notes in music scores and those of fine-grained time notes in music performances. For score notes, the onset time can be represented by the note position in the measure grid, and one can also add note properties with regard to the environment (Jeong et al., 2019a; Dong et al., 2023; Rhyu et al., 2022), such as the type of instrument, the position in a chord, the number of other simultaneous notes, the staff, if it is on a (down-)beat, ... Performance notes on the other hand can be described solely with pitch, onset, duration and velocity properties: in particular, onsets are represented by the Inter-Onset-Interval (IOI), which indicates the time elapsed between the onsets of the current and previous notes, as the notes are sorted by their onset time. Note-wise representations are particularly efficient for modeling and manipulating notes individually, which is relevant for performance rendering among other tasks.

More visually friendly, **piano rolls** represent symbolic music in a 2D matrix $X \in [0,1]^{T\times 88}$ with time and pitch axis. Time is unrolled at a constant frame rate F_{frame}, while the pitch axis spans over the effective MIDI pitch range (88 for the piano). The most commonly used piano roll is the active velocity piano roll, as most DAWs offer a graphical interface for MIDI edition. A bin $X(t,p)$ in the roll indicates if the note with pitch p is being played or not at time t (if $X(t,p) > 0$) and with which initial velocity. In this representation, one cannot distinguish repeated notes inside sustain pedal from long pressed notes (Kim et al., 2019) and thus may prefer combining it with other types of piano rolls, such as the onset velocity roll (non-zero only at onset times) or the activity roll (similar to the active velocity roll but without velocity information). Multiple instrument-specific piano rolls can be stacked together for multi-track settings. The piano roll representations are useful when relying on Computer Vision (CV)-inspired model architectures (Brunner et al., 2018) and/or when leveraging the constant time step (e.g. synthesis with upsampling (Kim et al., 2019; Cooper et al., 2021)). However in practice, the piano roll encodings are often sparse matrices, making them inefficient in terms of memory usage.

Finally, graph-based methods have also been applied for symbolic music processing by encoding scores as **graphs** (Jeong et al., 2019b). Notes are nodes connected with different types of edges according to their relationship with other elements in the score (following notes, rests, measures, slurs, voice,...).

Section summary - Symbolic music processing

Western music has long been made, transmitted and analyzed using music scores between human composers, musicians and musicologists. However, new instruments leveraging technological progress fostered the creation of low-level machine-readable representations of music, with the MIDI protocol being their flagship. As such, different encodings have been proposed for processing these representations, with different relationship granularities, ranging from the grounded MIDI-like tokenization to the graph-based encoding, through the more visual piano rolls. For this manuscript, **note-wise** sequences and **piano rolls** will be preferred as they can both meaningfully encode piano performances and be extracted from raw MIDI files.

2.2 Audio Digital Signal Processing basics

Music is essentially consumed by listening to it: like any other sound, it is transmitted from the source (a voice, an instrument, a loudspeaker,...) to the listener by disturbing the air molecules due to the propagation of the soundwave. These local variations of pressure can then be measured and converted to an electrical signal by an electro-mechanical system (the ear, a microphone, a sensor), to be then further processed and eventually recorded and understood by the brain or a computer.

Signal processing is the scientific sub-field that has been developing mathematical properties, theories and operations for explaining and manipulating the information contained in signals. While applicable to many types of signals (text, videos, electrical currents, stock exchanges, weather data, etc...), the signal processing elements presented in the following section are of particular interest for audio signals and will serve as a basis for the remainder of the manuscript. After introducing the waveform and time-frequency representations of 1D signals, filtering operations for signal manipulation will be defined, and all will finally be combined for presenting the spectral-modeling paradigm for sound re-synthesis.

2.2.1 Waveform representation

Sound signals, whether acoustic or electrical, are sequences of measurements of physical quantities that evolve over time. However, computers can only process and store data with discrete values, by manipulating binary registers. Thus, physical signals need to be discretized, or digitalized, by *sampling* in time and *quantization* in amplitude. Sampling and amplitude quantization are commonly parameterized by the **sampling rate** F_{audio} and the **bit-depth** respectively.

Bit-depth indicates the number of bits, and concurrently the number of possible values, used for representing amplitudes. Whereas storage of uncompressed digital signals usually uses quantification with fixed point numbers defined by the bit-depth (typically 16, 24 or 32-bits), computer digital signal processing usually uses floating point numbers, typically 32 or 64-bits. Generally, the larger the size of the encoded numbers, the smaller the quantification noise will be.

Likewise, the sampling rate defines the number of evenly spaced measurements of the audio signal per second. The information in between these audio samples is lost during the process. Thankfully, the Nyquist-Shannon sampling theorem (Shannon, 1949) provides a mathematical criterion for sampling signals while maintaining informational integrity. It states that no information is lost in the sampling process if the maximum frequency contained in the original signal does not exceed $F_{\text{audio}}/2$, commonly referred as the Nyquist frequency. If this criterion is not met, aliasing can appear in the digitalized signal, distorting it through the creation of unwanted frequencies. In that respect, an anti-aliasing filter can be applied before quantization to remove unwanted frequencies above the Nyquist frequency. As human auditory is sensible in the 20Hz to 20kHz frequency range, 44.1 kHz and 48kHz are common sampling frequencies used for faithfully encoding audio signals in CDs, mp3 files, ... Nonetheless, auditory perception studies (French and Steinberg, 1947) have shown that information below 4kHz is sufficient for understanding

speech. Hence, reduced sample rates of 8kHz or 16kHz are also common practice in telephone transmissions and in the speech processing literature respectively.

2.2.2 Time-Frequency representation

While the raw waveform representation reflects the way sound is physically measured, it is not the best for highlighting the properties to which auditory perception is sensitive. For sounds and especially for music, the audio signal often contains repeating patterns in the waveform. The number of times these patterns loop per second corresponds to the frequency of the pattern, and human auditory is particularly responsive in tracking frequencies. On that account, time-frequency representations based on Fourier analysis aim to exhibit the main frequency components embedded in audio signals.

Discrete Fourier Transform (DFT)

Here, we are given a digital audio signal $\{y(t)\}_{t\in\mathbb{Z}}$, sampled at frequency F_{audio} and with finite support of size $m \in \mathbb{N}^*$, meaning that the signal is zero outside of a finite time interval of size m. Without loss of generality, the signal can be translated in order to be non-zero for $t \in [\![0, m[\![$.

The DFT is a mathematical operator aiming to extract the spectral content of the signal y, by "measuring" the presence of certain frequencies in it. The DFT of y is expressed as $\forall f \in [\![0, m[\![$:

$$\text{DFT}[y](f) \coloneqq \sum_{t=0}^{m-1} y(t) e^{-2j\pi \frac{f}{m} t}. \tag{2.3}$$

A single element $\text{DFT}[y](f) \in \mathbb{C}$ indicates the amplitude and phase of the sinusoidal component with frequency $F_{\text{audio}} \times f/m$ contained in the signal y. It can be seen that the signal length m directly influences the resolution of the extracted spectral content, as higher m values project the signal y into a larger basis of exponential frequencies $\{e^{-2j\pi \frac{f}{m} t}\}_{f\in[\![0,m[\![}$ and reduce the gap F_{audio}/m between two neighboring frequencies. A DFT of size larger than m can be computed by means of padding a number of zeros at $t \geq m$ after the signal y: this is equivalent to resampling the frequency spectrum.

The DFT of a signal can be used to visualize, analyse and further process its frequency content. In the latter case, it can be useful to revert back to the audio modality from a DFT representation, which can be done with the Inverse Discrete Fourier Transform DFT^{-1}, defined as $\forall t \in [\![0, m[\![$:

$$\text{DFT}^{-1}[\text{DFT}[y]](t) \coloneqq \frac{1}{m} \sum_{f=0}^{m-1} \text{DFT}[y](f) e^{2j\pi \frac{f}{m} t} = y(t). \tag{2.4}$$

Short-Term Fourier Transform (STFT)

Signals are usually non-stationary and their frequency content evolves over time. For examples, the recording of a musical performance includes multiple successive notes and

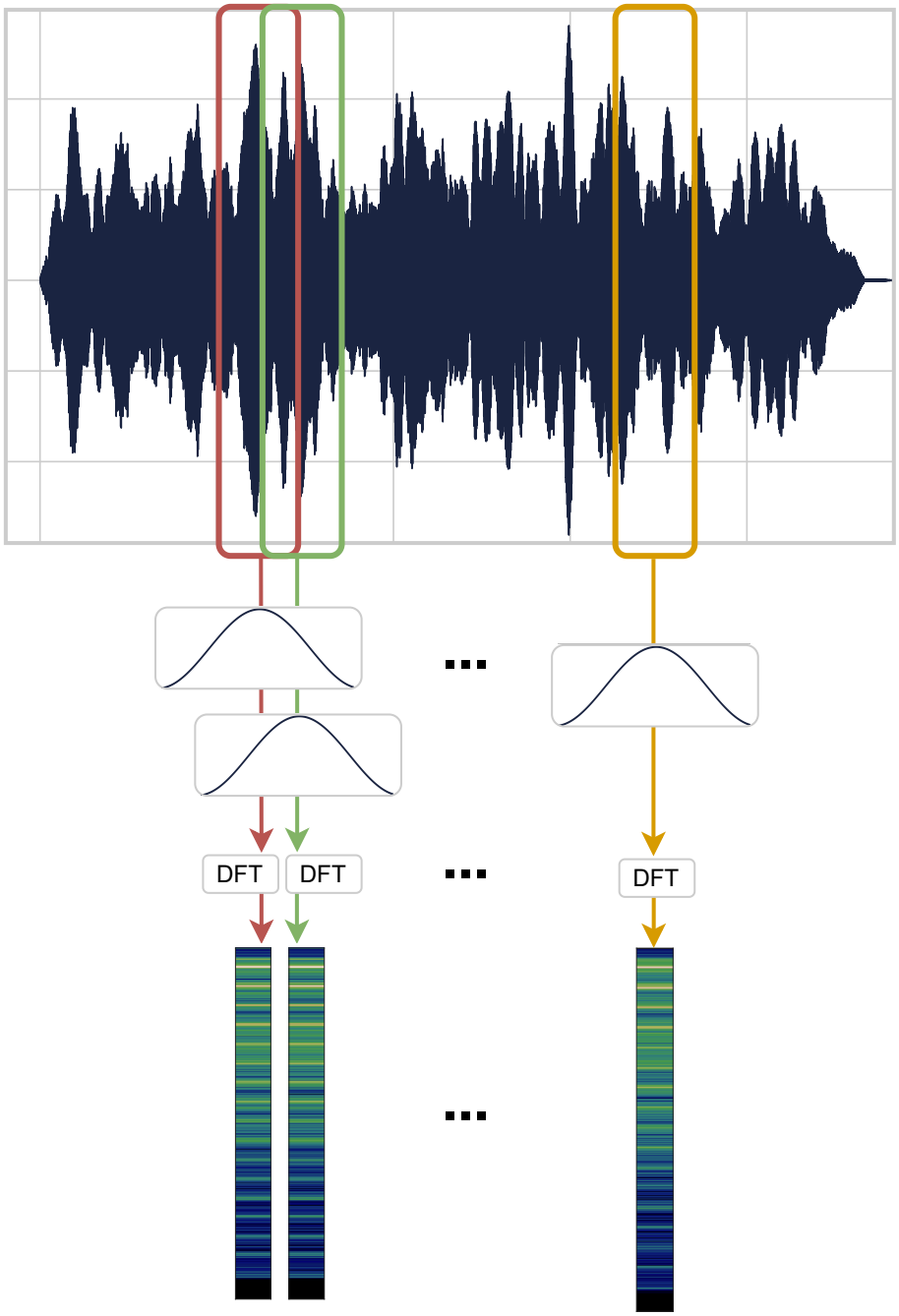


Figure 2.3: Extraction of the magnitude spectrogram from an audio waveform.

a voice articulates different vowels in sequence. Performing a DFT over such signals would entangle the contents of each note/word and not provide an informative representation on the full signal.

To mitigate this issue, the STFT is a classic signal processing tool that performs multiple DFTs on successive chunks of signal that are supposed to be locally stationary. The chunks are extracted by multiplying the signal with a sliding window function w :

$\mathbb{Z} \to \mathbb{R}$ that has finite support of size m and is centered around zero. The sliding is done by successively shifting the window in time by a chosen number of samples $h_w \in [\![1, m]\!]$ called the **hop size**. By naming the shifted window $w_\tau : t \mapsto w(t - h_w\tau)$, the STFT can be defined $\forall \tau, f \in \mathbb{Z} \times [\![0, m[\![$:

$$\begin{aligned} \mathrm{STFT}[y](\tau, f) &:= \mathrm{DFT}[y \times w_\tau](f) \\ &= \sum_{t \in \mathbb{Z}} y(t) w(t - h_w\tau) e^{-2j\pi \frac{f}{m} t}. \end{aligned} \tag{2.5}$$

The simplest window function is the rectangular function (1 for $t \in [\![-m/2, m/2]\!]$, 0 elsewhere) but it introduces noticeable artifacts in the DFT of the extracted signal (Gottlieb and Shu, 1997). To mitigate this issue, several window functions have been proposed in the signal processing literature (Harris, 1978). In particular, the Hann window function has seen widespread uses:

$$w^{\mathrm{Hann}}(t) := \begin{cases} \frac{1}{2}(1 - \cos(2\pi \frac{t}{m}) & \text{if } t \in [\![-\frac{m}{2}, \frac{m}{2}]\!] \\ 0 & \text{elsewhere} \end{cases} \tag{2.6}$$

On top of the window function choice, the parametrization h_w, m determines the time-frequency resolution of the STFT. The hope size h_w sets simultaneously the **overlapping ratio** $1 - (h_w/m)$ and the **temporal precision** h_w/F_{audio} between two STFT frames. It notably determines the ability to reconstruct the original signal y from its STFT, but this property will not be addressed in this thesis.

On the other hand, m is commonly referred as the DFT size as it extracts a sub-signal with support of size m from a signal of arbitrary length, thus conditioning the frequency resolution of the local DFT. The choice of its value is associated with the time-frequency trade-off of the STFT resolution: a high value of m enables analyzing the signal through more frequencies, but if m is too large, it can violate the local stationarity hypothesis of the signal chunk and introduce "blurriness" in time.

Finally, it is more convenient to process real-valued data rather than complex-valued, and it is common practice to use the **magnitude spectrogram** $|\mathrm{STFT}|$, which is the absolute magnitude of the STFT. Even though the absolute phase information is not relevant for auditory perception, the difference of phases is, and retrieving the discarded phase raises a challenge notably for audio synthesis from/through magnitude spectrograms.

2.2.3 Linear Filtering

While the DFT and STFT serve as valuable tools for analyzing and visualizing the frequency content of a signal, linear filters can be used to shape it. With a view to alter the audio signals, designing linear filters is a valuable asset as they can enhance or attenuate the amplitude of certain frequencies without creating nor modifying the frequency values themselves: they have found usages for numerous tasks, such as mixing (see Section 1.1.4), sound design, speech enhancement, noise suppression, etc...

Filters have either an Infinite Impulse Response (IIR) or a Finite Impulse Response (FIR) and are Linear Time-Invariant (LTI) or Linear Time-Varying (LTV), which makes four possible combinations: each property will be presented in the following.

Finite Impulse Response (FIR) vs Infinite Impulse Response (IIR)

Linear filters can be defined in the waveform/audio modality by their impulse response IR, or in the frequency modality by their frequency response $\boldsymbol{\eta} \coloneqq \text{DFT}[\text{IR}]$.

A FIR filter has coefficients $\{\text{IR}(t)\}_{t\in\mathbb{Z}}$ with finite support of size Q, the support size determining the **filter order**. Applying such a filter to a signal $\{y(t)\}_{t\in\mathbb{Z}}$ gives $\forall t \in \mathbb{Z}$:

$$\tilde{y}(t) = (y * \text{IR})(t) = \sum_{t'\in\mathbb{Z}} \text{IR}(t')y(t-t'), \tag{2.7}$$

with $\tilde{y}$ the filtered audio signal and $*$ the discrete convolution operation. Filtering a signal with a FIR filter means replacing each sample by a linear combination of it and its neighboring samples. If $\forall t \in \mathbb{Z}^*_-, \text{IR}(t) = 0$, the output sample $\tilde{y}(t)$ is computed using only input samples from the current and previous timesteps: the filter is said to be **causal**.

The impulse response of an IIR filter has an infinite sized support. Recursive filters are a practical sub-family of IIR filters where past output samples are also used to compute the current output sample. A recursive filter can be defined by its coefficients $\{a_t\}_{t\in[\![0,Q_1[\![}$ and $\{b_t\}_{t\in[\![0,Q_2[\![}$, with Q_1 and Q_2 the feedforward and feedback filter orders respectively. $\forall t \in \mathbb{Z}$, the relationship between the input signal y and the filtered output $\tilde{y}$ is given by:

$$\sum_{t'=0}^{Q_1-1} a_{t'}\tilde{y}(t-t') = \sum_{t'=0}^{Q_2-1} b_{t'}y(t-t') \tag{2.8}$$

Taking benefit of the convolution theorem of the Fourier Transform, FIR filters can also be applied in the frequency modality with the product of the respective DFTs:

$$\text{DFT}[y * \text{IR}] = \text{DFT}[y] \times \text{DFT}[\text{IR}], \tag{2.9}$$

while ensuring the DFT size is suitable with regards to the support sizes of IR and y to avoid time aliasing.

Such a formulation facilitates a more intuitive filter design process: a desired frequency response can be achieved by sampling the frequency range into m linearly spaced bands and specifying the magnitude and phase of each of them, and then computing the inverse DFT. The impulse response obtained by such a method is clearly finite with support of size m. Similarly, **frequency sampling** can evaluate the frequency response of an IIR filter at multiple frequencies, yielding a FIR approximation of the IIR filter, with guaranteed stability at the risk of potential aliasing in the time modality.

Linear Time-Invariant (LTI) vs Linear Time-Varying (LTV)

The FIR and IIR filters presented previously shape the input signal with the same response through the full signal length: they are called LTI. But similarly to the STFT performing successive DFTs over a non-stationary signal, they can also be designed as LTV and have a non-stationary behavior by allowing their coefficients to evolve over time.

A way of applying a LTV-FIR filter for example would be to define a frame rate F_{frame} at which the filter coefficients are updated. The hop size between two updates is

$h_w = F_{\text{audio}}/F_{\text{frame}}$, which also corresponds to the upsampling ratio from the frame rate to the audio sampling rate. Here, the LTV filter is a set of LTI filters $\{\text{IR}_\tau\}_{\tau\in\mathbb{Z}}$ of common order Q, applied one after the other ordered by their frame index τ. The filtered audio $\tilde{y}$ would be $\forall t \in \mathbb{Z}$:

$$\begin{aligned}\tilde{y}(t) &= (y * \text{IR}_{\lfloor t/h_w \rfloor})(t) \\ &= \sum_{t'\in\mathbb{Z}} \text{IR}_{\lfloor t/h_w \rfloor}(t')y(t-t'),\end{aligned} \tag{2.10}$$

with $\lfloor . \rfloor$ the floor function. Manipulating Equation 2.9, a LTV-FIR filter can also be designed and applied in the frequency modality.

2.2.4 The Spectral Modeling Paradigm

An application of audio signal processing is the synthesis-by-analysis of sounds, or *sound matching*, which aims to find the parameters of a certain audio synthesis model for reproducing a target sound. By leveraging Fourier theory and analysis tools outlined in Section 2.2.2, a periodic signal can be decomposed into a sum of its elementary frequency components. On the other hand, additive synthesizers produce sounds by stacking multiple sinusoidal signals, parameterized by their evolving amplitudes, frequencies and phases. Therefore, as vocal and instrumental signals contain periodic phenomenons, such as glottal pulses, vibrating string or air column modes, they are suitable for being reproduced by additive synthesis.

The sinuses-plus-noise, or *spectral modeling* synthesis method was formalized by Serra and Smith (1990). It reproduces an audio signal by summing an additive signal y^{additive} with a residual noise signal y^{noise}:

$$\begin{aligned}y(t) &\approx y^{\text{additive}}(t) + y^{\text{noise}}(t) \\ &= \sum_{k=1}^{K} A_k(t)(\sin(2\pi \sum_{t'=0}^{t} f_k(t') + \Phi_k(0))) + \text{DFT}^{-1}[\boldsymbol{\eta}_{\lfloor t/h_w \rfloor}\boldsymbol{u}_{\lfloor t/h_w \rfloor}](t),\end{aligned} \tag{2.11}$$

with $K \in \mathbb{N}^*$ the number of pure sinuses (or **partials**) in the additive signal, and A_k and f_k their time-varying amplitude and frequency controls. The instantaneous phase of the k-th partial can be noted $\Phi_k(t) \coloneqq 2\pi \sum_{t'=0}^{t} f_k(t') + \Phi_k(0)$, with $\Phi_k(0)$ the initial phase. $\boldsymbol{\eta}_{\lfloor t/h_w \rfloor}$ is the instantaneous frequency response at frame $\tau = t/h_w$ of a LTV filter shaping a white noise STFT spectrum $\boldsymbol{u}$ in the frequency modality. While the additive signal reproduces the periodic phenomenons, the residual noise encompasses stochastic elements in the signal, such as wind, bow or transient noise. An example of sinuses and noise decomposition of a signal is shown in Figure 2.4.

The partials often have an harmonic relationship, meaning that their frequencies are integer multiples of a fundamental frequency f_0: $\forall k \in [\![1, K]\!], f_k(t) = kf_0(t)$. In this case, the partials are called **harmonics** and the complexity of the synthesizer is reduced as only the fundamental frequency needs to be estimated. Pitch is commonly associated with the fundamental frequency, even if studies (Oxenham, 2012) have demonstrated other properties responsible for the perception of pitch.

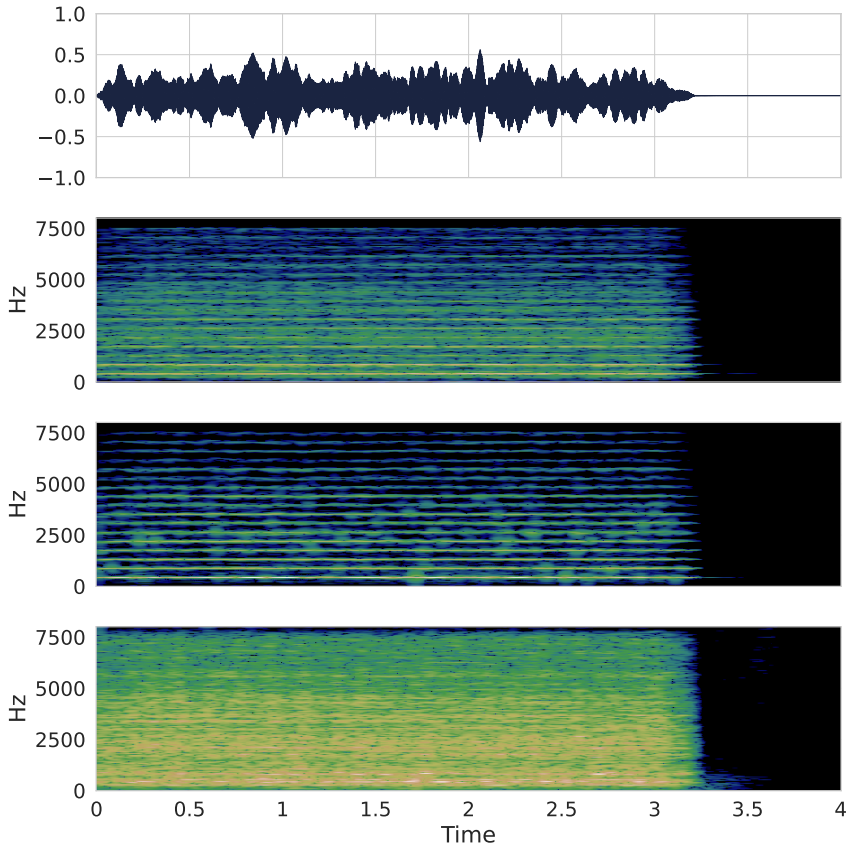


Figure 2.4: Waveform (first row) and spectrogram (second row) of a violin note decomposed into its harmonic content (third row) and residual noise (last row).

For sound matching, the analysis process is of utmost importance in order to faithfully reproduce the target sound. Various methodologies, including peak detection and trajectory tracking, have been devised to accurately estimate synthesizer controls, and will be briefly discussed in the literature review Section 3.2.2.

Section summary - Audio digital signal processing basics

The scientific field of signal processing has developed mathematical tools and properties for the analysis and manipulation of notably audio signals. For digital processing, the signals have to be quantized in amplitude and time (usually at 16kHz, 24kHz or 48kHz). Through Fourier analysis, time-frequency representations can exhibit frequency components in the signal and linear filters can shape them locally or globally in time. The spectral modeling synthesis method leverages these tools to decompose an audio signal into the sum of a filtered noise and partials with varying amplitudes and frequencies.

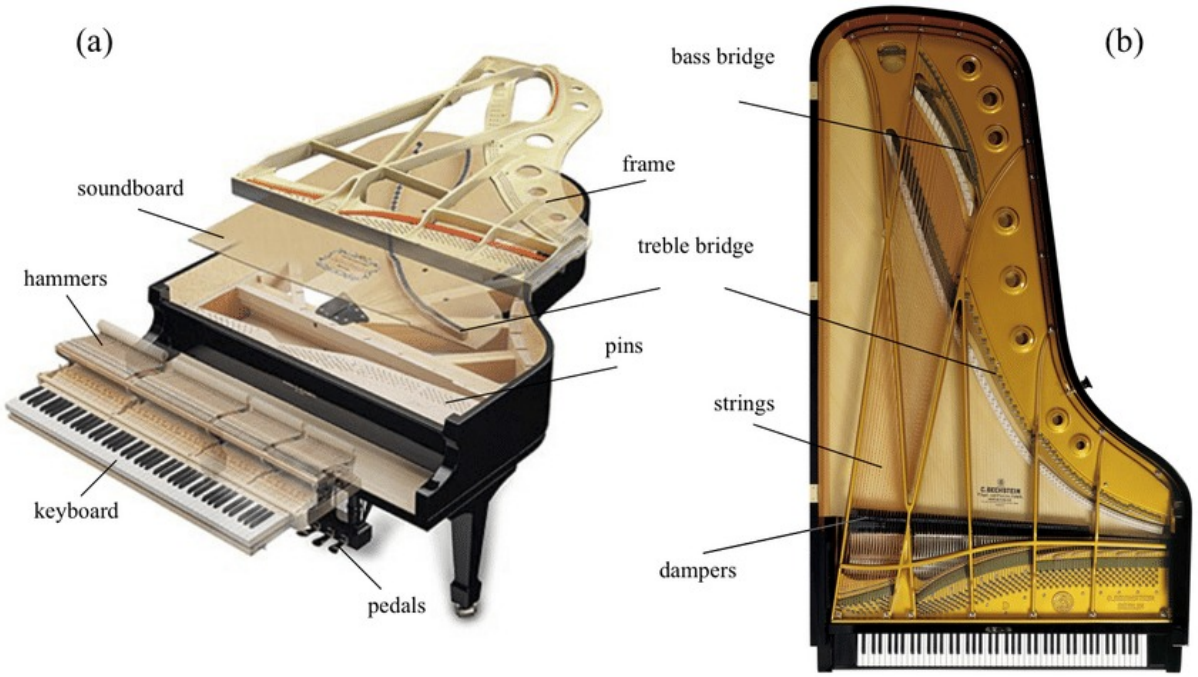


Figure 2.5: Grand piano structure. Taken from Rigaud (2013), with Figure (a) sourced from www.pianotreasure.com and Figure from (b) www.bechstein.de

2.3 Piano Mechanisms

The piano is one of the most popular instruments in the western music history, as it has spawned a large repertoire of solo works since its inception and has been featured in numerous multi-instrument formations in a broad variety of musical genres (classical, pop, jazz, funk, etc...). While quite easy to start playing with, professional performances can be complex: the instrument popularity lies in its flexibility, as its offers both polyphony and a wide range of pitches and dynamics.

This section will cover the necessary elements to understand the mechanisms behind the piano. After a quick historical overview of the instrument, the roles of each of its main component (strings, pedal, soundboard) are presented, eventually with some physical modeling results. Notably, the large frequency range of the instrument introduces sizeable challenges for piano makers and tuners. The reader may refer to Figure 2.5 for a comprehensive illustration of the instrument, and can also find a more in-depth modeling of the instrument in the thesis of Chabassier (2012).

2.3.1 A Brief History of Piano Craftsmanship

Keyboard instruments traditionally condense a physical sounding phenomenon into a single ready-to-be-actioned key, button or lever. Several of these keys are available, with the same mechanism tuned differently for each. As such, keyboard instruments are defined by the way they are played, instead of how the sound is produced. The first keyboard instrument can be tracked back in Ancient Greek times with the hydraulis, an organ

where the pipes are excited by air pressurized by water pumps (Apel, 1997). Since then, other keyboard instruments have emerged, leveraging different sound sources: aerophones use air flows (e.g. organs and accordions), idiophones set physical object in vibration (celesta, xylophone, glasschord or the carillon), chordophones set strings in vibration (harpischord, clavichord or hurdy-gurdy) and electrophones use electrical waves (mellotron, Ondes Martenot).

The inception of the piano mechanism is commonly associated with the creation of the pianoforte by Italian manufacturer Bartolomeo Cristofori around 1700. Based on the harpischord structure, the pianoforte employ buckskin-covered hammers to strike the strings instead of plectrums plucking them. The proposed action allowed for the hammers to be "launched" at varying velocities according to the strength of the key pressing, without the hammers sticking to the strings nor bouncing back at them afterward. Such mechanism allowed for nuances in dynamics during playing, which was unprecedented with the harpischord.

From that point on, the evolution of the instrument was intertwined with the evolution of musical stylistic movements. Close collaborations between instrument makers and renowned composers/musicians were common, as the latter could provide valuable feedback on the instrument while also being inspired to new composition processes thanks to technological improvements in instrument making (Grasser, 1995). For example, German builder Gottfried Silbermann improved on the key hardness and lack of power of higher notes following the criticisms of Bach. Johann Andreas Stein proposed a precursor system to the damping pedal and also improved over the original piano action through an innovative escapement mechanism that allows to better control the key velocities. He founded what would be later known as the Viennese school of piano makers, which were adopted by composers of the classical eras such as Haydn, Mozart and early Beethoven and Schubert.

Later on, the English craftmanship further pushed the boundaries by incorporating metal parts into the wooden structure, which could then withstand increased tension of the strings and make a more powerful instrument overall. Such design would suit the energetic and dramatic writing style of Beethoven for instance. The further enlargement of the piano dynamic range was a keystone of the romantic era, notably through the design of a full iron frame by Alpheus Babcock and the wrapped and tempered steel strings by Henri Pape in the 1820s. In 1821, Sébastien Érard also developed the double escapement design, or repetition lever, that allows the fast repetition of notes even if the action is not fully reset. This was rapidly adopted by Franz Liszt with its 8 Variations piece in 1823, demonstrating the new possibilities offered by such a system. There again, Liszt and Erard would later share professional and friendly collborations, as would Chopin with Camille Pleyel, heir to the French manufacturing firm.

2.3.2 A Monophonic String Model

The string is the main vibrating element making up the sound of the piano. For a single note, the steel string is attached at both ends and is excited by a single strike of the hammer, launched upon key pressing at **note onset** time, thanks to the double

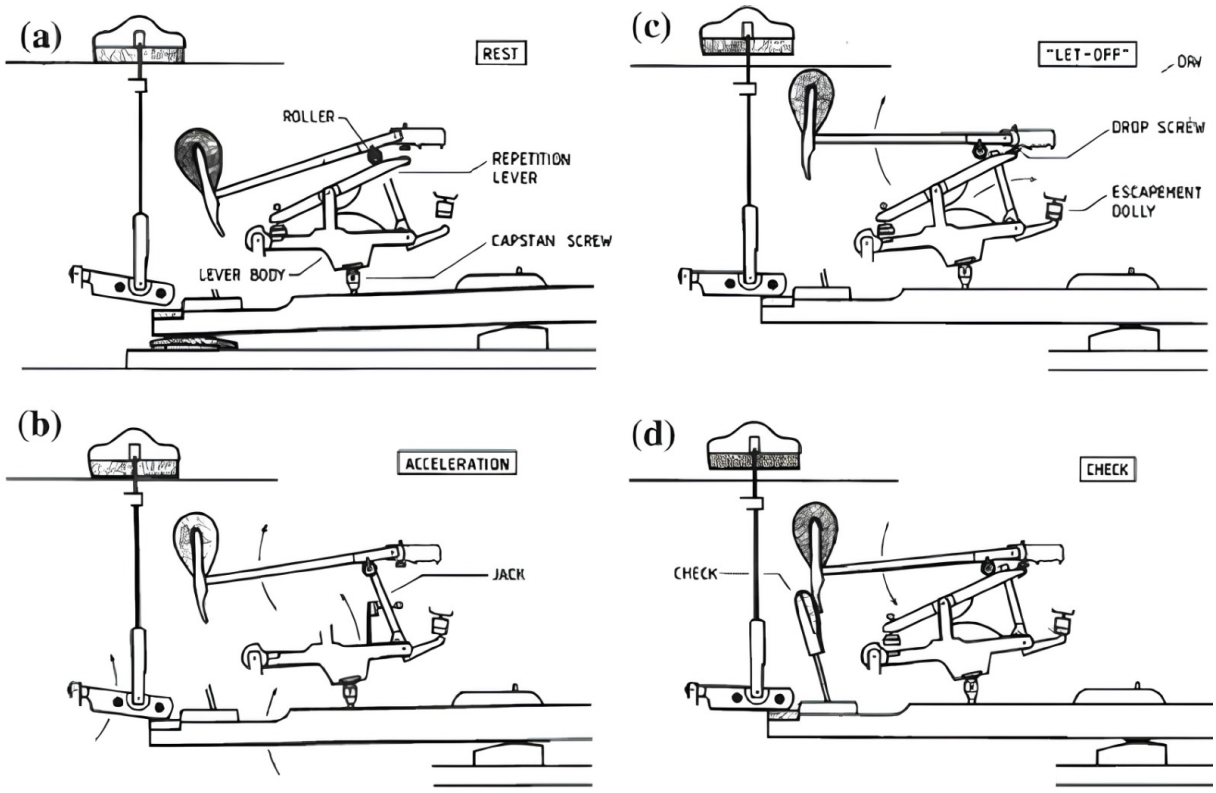


Figure 2.6: Diagram of the successive stages (a, b, c, d) of the double escapement mechanism, taken from Askenfelt and Jansson (1990).

escapement mechanism illustrated in Figure 2.6. Upon key release at **note offset** time, the damping mechanism returns to its resting state and attenuates the string vibration.

In theory, the transversal motion of an ideally flexible, vibrating string consists of harmonics featuring frequencies that are integer multiples of its fundamental frequency. However, piano strings are made of tempered steel that induces a non-negligible stiffness: that stiffness tends to bring the bent string back to its resting state, even without tension applying, which violates some hypothesis in the ideally flexible string settings. It has been demonstrated by Fletcher (1964) that the partials $\{f_k\}_{k\in\mathbb{N}^*}$ of the piano string satisfy instead the relation $\forall k \in \mathbb{N}^*$:

$$f_k = k f_0 \sqrt{1 + Bk^2}, \tag{2.12}$$

with f_0 the fundamental frequency and B the **inharmonicity coefficient**, which depends on the string properties (tension, length, diameter and Young modulus). As illustrated in Figure 2.7, the inharmonicity slightly increases the partial frequencies, with more pronounced shifts for higher partials. This particularity greatly contributes to the recognition of the piano tone.

To ensure uniformity of sound across the pitch range, despite large differences in length and diameter, higher strings can be doubled or even tripled in number to compensate for their low output. Strings of such duets or triplets are usually slightly detuned from each other, leading to the appearance of beatings between neighboring frequencies: for a given note, this is manifested as amplitude modulations of its partials. According to Weinreich

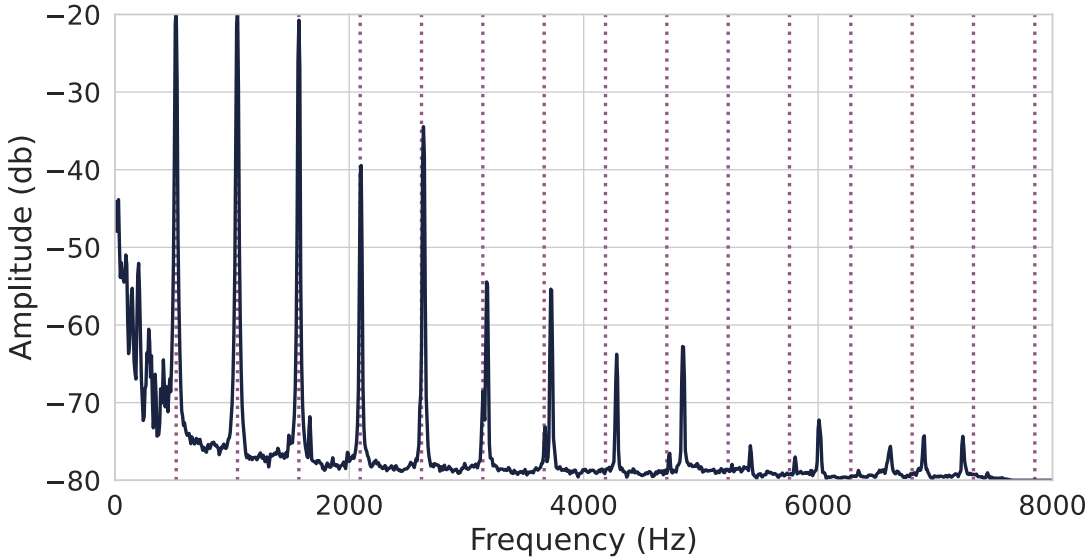


Figure 2.7: Spectral content of the ENSTDkCl C5 note from MIDI Aligned Piano Sounds (MAPS) (Emiya et al., 2010): the inharmonicity shifts the partials higher than pure harmonic frequencies (in dotted vertical lines).

(1977), the sub-strings being detuned while coupled at the bridge are also responsible of the distinct "piano aftersound". Indeed, most physical systems excited by an impulse have their motion decreases exponentially along a constant decay rate in time. However, piano notes exhibit a *double exponential decay* phenomenon: after a certain amount of time of regular exponentially decaying amplitudes, the sub-strings mutually maintain their motion through their coupling at the bridge, giving rise to a second, slower, exponential decay rate.

Additional phenomena occurring within piano string vibration involve:

- the presence of a fast longitudinal wave that can be heard at the very beginning of the piano tone (Knoblaugh, 1944; Askenfelt and Jansson, 1993).

- the multiple polarizations of the string motion. Even if the hammer strikes vertically, the string has been observed to move both vertically and horizontally, which can also explain the double amplitude decay (Weinreich, 1977).

- the resonance of the string portions between the bridge and the attachment point. As they are not necessarily damped, they can also resonate. In high-end pianos, they can be tuned with one of the strings harmonics (Askenfelt and Jansson, 1993).

- the presence of the so-called *phantom partials* in the spectrum, located at certain sums and differences of the main partials frequencies (e.g. $f_1 + f_2$, $f_2 + f_5 + f_6$, $f_{14} - f_{16}$,...). They result from the combination of the string stiffness, soundboard coupling and other non-linearities, such as non-uniform tension (Conklin, 1999).

The piano action is supplemented with pedals that provide further control options. Pianists manipulate these pedals with their feet, organized from right to left:

- the **sustain** pedal - also called *damper* pedal - can lift and disable the damping mechanism, which allows the string(s) to still vibrate even after note offset. When pressed, all dampers for all notes are lifted simultaneously.
- the **sostenuto** pedal (for grand pianos) can be viewed as a selective sustain pedal: when the pedal is pressed down while a piano key is being played, the associated damper is lifted and remains disabled as long as the sostenuto pedal in pressed. Its usage is rarer than the sustain pedal since the note pressed down pre-requisit makes it more difficult to compose with, but it allows for an overall "cleaner" sound as the other unwanted notes are still damped.
- the middle pedal for upright piano lowers a muffler between the hammer and the string, which highly softens the sound and changes the timbre. Its usage is mostly practical as a pianist can rehearse while less disturbing its surroundings since the sound is heavily soften.
- the **una corda** pedal aims to soften and change the note timbre. For grand pianos, it slightly shifts the whole keyboard so that the hammer only strike one string among the duets/triplets, reducing the partial beatings notably. For upright pianos, it instead brings the hammer closer to the string: as the hammer is launched from a closer point, it reaches the string with an lessen velocity, but does not change the timbre in the same way as all strings in duets/triplets as still excited.

2.3.3 Soundboard

Because of their small surface area, the piano strings by themselves do not radiate enough energy to be heard from a reasonable distance. Like other string instruments such as guitars, violins, and harps, which rely on resonance chambers, the piano strings need to be coupled with a more radiating element. To this end, the bridge transmits the string vibration to the soundboard, from which the sound radiates in the air. Coupling with the strings at the bridge involves a trade-off: a heavy coupling enhances transmission to the soundboard but increases the decay rate, wheres light coupling allows for longer string vibration at the expense of a reduced transmission to the soundboard, and thus a reduced sound output (Conklin, 1996).

As depicted in Figure 2.8, the soundboard exhibits modes, the 2D equivalent of a string partials. While transmitting the string vibration, these modes highlight certain frequencies: the soundboard thus acts as a filter on the transmitted string sound. Additionally, the soundboard acts as a sound-emitting element, as it is excited with the whole piano structure by the key and hammer strikes. Due to its distinctive shape and the presence of ribs (Boutillon and Ege, 2013), the frequency content of the soundboard's percussive sound is dependent of the location of excitation: it tends to be more bass-heavy when excited towards the low notes range, while showcasing higher frequency content when excited in the treble range.

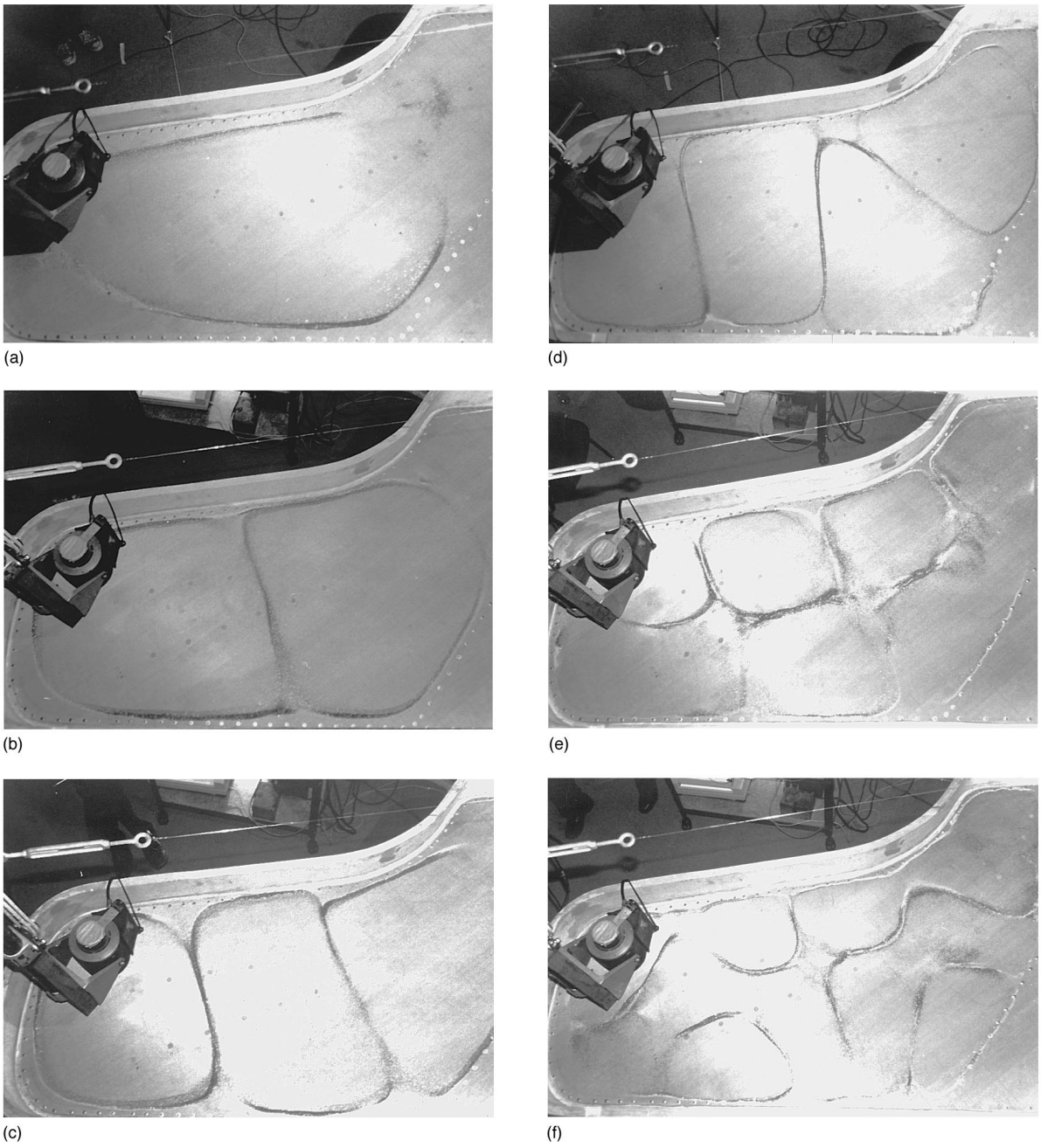
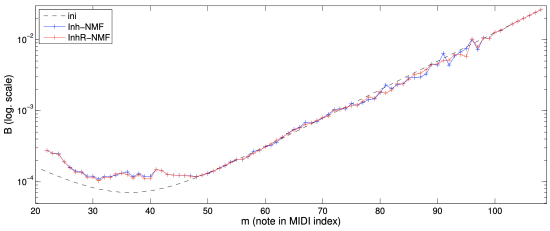


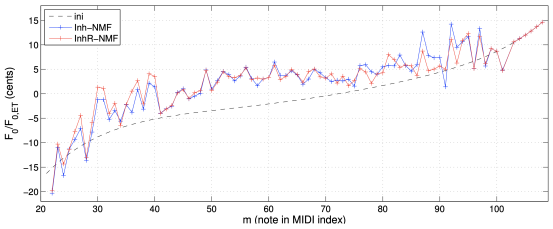
Figure 2.8: Visualization of 2D stationary modes (also called Chladni patterns) of a 2.74m grand piano soundboard excited by a vibration generator, as observed by Conklin (1996). (a) mode 1 (49 Hz); (b) mode 2 (66.7 Hz); (c) mode 3 (89.4 Hz); (d) mode 4 (112.8 Hz); (e) (184 Hz); (f) (306 Hz).

2.3.4 Polyphony

One of the key features of the piano is the ability to play multiple notes simultaneously, or **polyphony**. As the notes range from A0 to C8 - corresponding to the 21 to 108



(a) Default inharmonicity curve and estimated inharmonicity coefficients.



(b) Default detuning curve and estimated detuning coefficients.

Figure 2.9: Default parametric inharmonicity and detuning models over the pitch range proposed by Rigaud et al. (2011) (in black dashed line). Individual notes inharmonicity and detuning coeffcients estimated on the Iowa piano database (https://theremin.music.uiowa.edu/MISpiano.html) are estimated with two methods based on non-negative matrix factorization proposed by Rigaud (2013) (red and blue dots). Both plots are taken from the appendix of Rigaud (2013).

MIDI range -, piano makers need to establish a "string plan", specifying the dimensions and placements of each string, such that the overall tension can still be compensated by the iron frame and the soundboard, while maintaining a global timbre coherence. This results in large differences of length and diameter between the lowest and highest notes, and consequently the need to adapt individual actions accordingly. The lowest notes have a single dedicated string with wrapping to further increase their density while keeping the inharmonicity coefficient to reasonable values. Then, to compensate for the reduced output level of individual strings, a few low notes have string duets, then all remaining mid-to-high notes have string triplets. The highest notes are even deprived of a damping mechanism as they naturally fade quickly.

On top of the manufacturing challenge, the tuning of each note is also non-trivial in the polyphonic context. Indeed, due to the non-negligible strings inharmonicity, adhering strictly to the true temperament system (where the frequency ratio between two semitones remains constant) would result in dissonances, as displaced partials would conflict with the fundamental frequencies of higher notes. To mitigate this issue, piano tuners would align the second partial $f_2(x^{\text{pitch}})$ of a note $x^{\text{pitch}} \in [\![21, 108]\!]$ with the fundamental frequency $f_1(x^{\text{pitch}} + 12)$ of the note an octave above. For low notes with many audible partials and high inharmonicity coefficients, the tuner may prefer to match a higher partial with a higher matching note, for example matching $f_3(x^{\text{pitch}})$ with $f_1(x^{\text{pitch}} + 18)$, or $f_4(x^{\text{pitch}})$ with $f_1(x^{\text{pitch}} + 24)$. The *octave type* designates the rank of the partial selected for tuning. As depicted in Figure 2.9b, lower notes are tuned slightly lower than true temperament frequencies, while higher notes are tuned slightly higher. This curve is commonly known as the Railsback curve (Railsback, 1938).

Rigaud et al. (2011) proposed a parametric model for modeling both the inharmoncity and detuning coefficients along the whole piano pitch range, or **tessitura**. The inharmonicity curve is observed and approximated as the sum of two linear asymptotes in the exponential scale. For a note with MIDI pitch $x^{\text{pitch}} \in [\![0, 128[\![$, the associated inharmonicity coefficient B is given by:

$$B(x^{\text{pitch}}) = \exp\left(\alpha_T x^{\text{pitch}} + \beta_T\right) + \exp\left(\alpha_B x^{\text{pitch}} + \beta_B\right), \tag{2.13}$$

with $\{\alpha_T, \beta_T\}$ (respectively $\{\alpha_B, \beta_B\}$) the parameters of the linear asymptote in the treble (respectively bass) range.

Additionally, one can define the detuning of a note x^{pitch} as the ratio between the fundamental frequency and the corresponding equal temperament frequency $\delta f_0(x^{\text{pitch}}) := f_0(x^{\text{pitch}})/f_{0,\text{ET}}(x^{\text{pitch}})$. By taking a **reference note** $x_{\text{ref}}^{\text{pitch}}$ unchanged from the equal temperament tuning ($\delta f_0(x_{\text{ref}}^{\text{pitch}}) = 1$) and applying the octave-based tuning strategy described above, the detuning model proposed by Rigaud et al. (2011) gives:

$$\delta f_0(x^{\text{pitch}}) = \sqrt{\frac{1 + B(x_{\text{ref}}^{\text{pitch}}) \times \left(\frac{f_{0,\text{ET}}(x^{\text{pitch}})}{f_{0,\text{ET}}(x_{\text{ref}}^{\text{pitch}})}\rho(x^{\text{pitch}})\right)^2}{1 + B(x^{\text{pitch}}) \times \rho(x^{\text{pitch}})^2}}, \tag{2.14}$$

with B the inharmonicity model from Equation 2.13 and ρ an octave type model along the tessitura. ρ is parameterized by the reference note $x_{\text{ref}}^{\text{pitch}}$, a bass asymptote value β and a decrease slope α:

$$\rho(x^{\text{pitch}}) = 1 + \beta \tanh^{\dagger -}\left(\frac{x^{\text{pitch}} - x_{\text{ref}}^{\text{pitch}}}{\alpha}\right), \tag{2.15}$$

with $\tanh^{\dagger -} : x \in \mathbb{R} \to (1 - \tanh(x))/2$ the hyperbolic tangent function reversed and scaled to the $[0, 1]$ interval. The modeled octave type is not necessarily an integer for all notes, but it can be interpreted as a compromise between two octave types: for instance, $\rho(x^{\text{pitch}}) = 1.6$ implies that the note x^{pitch} has been tuned by trying to match both its partials f_1 and f_2 with their respective octave notes, with a slight emphasis towards f_2. Note that the asymptotic value in the treble range is set to 1 as only a few partials are audible.

Finally, during piano performances, simultaneous notes can mutually influence each other as their neighboring partials interact. This interaction may induce amplitude modulation of shared partials, caused by slight detuning or phase discrepancies between them. Additionally, undamped notes, that are either being played, freed by the sostenuto/sustain pedal or deprived of dampers (for the highest notes), can have certain partials excited by the vibration of other notes sharing such partials: these are commonly referred as *sympathetic resonances*.

Section summary - Piano mechanisms

At its core, the piano relies on the simple phenomenon of string vibration for producing harmonious sounds. However, as the instrument and its practice evolved with joint craftsmanship and technical progresses, it is the seat of numerous complex phenomenons. For its harmonic component, steel strings are subject to inharmonicity that displaces their harmonic frequencies. Coupled through the bridge, the soundboard further shapes and conveys the string sound while also adding a percussive element. At last, the polyphony of the instrument allows mutual interaction between notes, but also requires a dedicated tuning for each of them in order to form a cohesive whole.

2.4 Deep Learning for Music Processing

This section briefly covers the theoretical framework underlying the methods employed during this thesis. Deep learning has emerged in the last decade as one of the most powerful optimization framework for handling a large variety of tasks, spanning from classification and regression to complex generation and synthesis. Deep learning-based models have proven countless times their ability to adapt to various domains and tasks, while surpassing by an order of magnitude previous methods that were more reliant on prior knowledge. Hence, numerous research and applications were developed for leveraging the capability of these models for a large array of domains and modalities, such as texts with NLP, images with CV, videos, audio, finance, biology, healthcare, robotics, materials science, etc...

The flexibility of deep neural methods comes from their faculty to extract relevant information and complex relationships from the data themselves, instead of relying on hand-crafted features or rules. Methods learning from data are more generally said to be **data-driven** and belong to the broader research area of machine learning, which is presented in the following. Then, the main operations usually employed for building deep neural networks are introduced and combined in an example of advanced architecture with Generative Adversarial Network (GAN). Admittedly, even if these models can be domain-ignorant, several strategies can be used for incorporating domain knowledge into the framework in order to improve the learning and potentially reach better performances.

2.4.1 Data-driven Optimization with Gradient Descent

Data-driven optimization is a family of optimization techniques aiming to approximate a probability distributions $\mathcal{P}$ that yields pairs $(x, y) \in \mathcal{X} \times \mathcal{Y}$ of *features* and *labels*, or *inputs* and *outputs*, or *conditioning* and *outputs*. For illustrations, $\mathcal{P}$ can be the set of all realistic animal pictures $x \in \mathcal{X}_{\text{images}}$ with their labeling $y \in \{\mathsf{cat}, \mathsf{nocat}\}$ of whether they contain a cat or not: this is a binary classification task. Or, if $\mathcal{P}$ yields all spoken English utterances $x \in \mathcal{X}_{\text{sounds}}$ with their written transcript $y \in \mathcal{Y}_{\text{text}}$ in Latin alphabet, we would have a transcription task to tackle.

To this end, a differentiable parametric function, or **model**, $\mathcal{F}_W$ with trainable parameters/weights $W \in \mathcal{W}$ is optimized, or *trained*, to ideally fit the unknown, underlying function $\mathcal{F} : \mathcal{X} \to \mathcal{Y}$ that explains the distribution $\mathcal{P}$, i.e. $\forall (x, y) \sim \mathcal{P}, \mathcal{F}(x) = y$.

Loss function

In this setting, quantifying the discrepancy between the ideal function $\mathcal{F}$ and the model $\mathcal{F}_W$ is made using the **risk** function of the problem (also called *loss*, *cost*, *error* or *objective*) $\mathcal{L}$ defined as:

$$\mathcal{L} : \begin{cases} \mathcal{W} \to \mathbb{R}^+ \\ W \mapsto \mathbb{E}_{x,y\sim\mathcal{P}}\, \mathbb{D}(y, \mathcal{F}_W(x)) + \lambda_{\text{reg}} \Phi(W) \end{cases} \tag{2.16}$$

with $\mathbb{E}$ the expectancy, $\mathbb{D} : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$ a differentiable distance-like function of the output space [1] and $\Phi : \mathcal{W} \to \mathbb{R}^+$ an optional *regularization* term weighted by $\lambda_{\text{reg}} \in \mathbb{R}$.

The left term of the risk function output in Equation 2.16 is the **empirical risk** that can be understood intuitively as the quantification of the "mistakes" made by the model: it measures the difference between the estimated outputs $\mathcal{F}_W(x)$ and the ground-truth examples $y = \mathcal{F}(x)$ for $(x, y) \sim \mathcal{P}$. The right term is the **structural risk** that penalizes unnecessary complexity in the model and it is mostly used to avoid overfitting, as it will be explained later on.

The optimization process aims to find the optimal parameters W^* in the parameter space $\mathcal{W}$ that minimize the risk $\mathcal{L}$:

$$W^* \coloneqq \arg\min_{W \in \mathcal{W}} \mathcal{L}(W). \tag{2.17}$$

In practice, deep learning practitioners mostly use the term **loss** to refer to the distance function $\mathbb{D}$ of the empirical risk. Outside of this chapter, for the sake of consistency with the field, "loss" will also designate the chosen distance function instead of the overall risk function of the problem, but the optimizers will still minimize both empirical and structural risks.

Differentiability

It was previously mentioned that the model $\mathcal{F}_W$ is **differentiable**, which is the key property that is used, in this context of gradient-based optimization[2], for adapting the model parameters to fit the model output to the distribution $\mathcal{P}$. This process is commonly denoted as the *training*, or *learning* phase.

In mathematics, a multivariate function is said to be differentiable with respect to a variable when it has finite derivatives at every point in the domain of said variable. Here, $\mathcal{F}_W$ being differentiable along its weights W means that $\forall x \in \mathcal{X}, \forall W \in \mathcal{W}, \forall w \in W, \partial \mathcal{F}_W(x)/\partial w$ exists and is finite. If the function output is scalar, the derivative vector along the multi-dimensional variable W is called **gradient** and notated $\nabla_W \mathcal{F}_W \coloneqq [\frac{\partial \mathcal{F}_W}{\partial w}]_{w \in W}$. By extension, the derivative of a multi-variate and multi-output function is the *jacobian* matrix $J_W \mathcal{F}_W \coloneqq [\frac{\partial \mathcal{F}_W^{(i)}}{\partial w}]_{w, i \in W \times [\![1, D_{\text{out}}]\!]}$, with D_{out} the output dimension and $\mathcal{F}_W^{(i)}$ the i-th component of $\mathcal{F}_W$.

Differentiability is stable by linear combination and by composition, and the *chain rule* allows to express the derivative of the composition of several differentiable operations with their respective derivatives. For example, let the model $\mathcal{F}_W$ be the composition of two differentiable functions/layers $\mathcal{F}^2_{w^{(2)}}$ and $\mathcal{F}^1_{w^{(1)}}$: $\mathcal{F}_W = \mathcal{F}^1_{w^{(1)}} \circ \mathcal{F}^2_{w^{(2)}}$. The derivative of $\mathcal{F}_W$ with respect to its weights $W = (w^{(1)}, w^{(2)})$ can be obtained through,

[1] It is not necessarily a distance function in the mathematical sense as for example, cross-entropy is commonly used for classification tasks but it is not symmetrical.

[2] Other optimization techniques, such as genetic algorithms, do not rely on differentiability in order to learn.

$\forall x \in \mathcal{X}, \forall (w^{(1)}, w^{(2)}) \in \mathcal{W}$:

$$\begin{aligned} \frac{\partial \mathcal{F}_W}{\partial w^{(1)}}(x) &= \frac{\partial \mathcal{F}^1_{w^{(1)}}}{\partial w^{(1)}}(\mathcal{F}^2_{w^{(2)}}(x)) \\ \frac{\partial \mathcal{F}_W}{\partial w^{(2)}}(x) &= \frac{\partial \mathcal{F}^1_{w^{(1)}}(y)}{\partial y}\Big|_{\left(y=\mathcal{F}^2_{w^{(2)}}(x)\right)} \times \frac{\partial \mathcal{F}^2_{w^{(2)}}}{\partial w^{(2)}}(x). \end{aligned} \tag{2.18}$$

Recursively, the chain rule can be applied to get the derivative along any weight in a differentiable model of arbitrary depth. This property is of particular interest in deep learning as neural models can stack multiple differentiable layers one after the other in order to increase the overall model complexity and capability.

It is worth noting that in the second equality of Equation 2.18, even tho $\mathcal{F}^1_{w^{(1)}}$ does not depend on $w^{(2)}$ and $\mathcal{F}^2_{w^{(2)}}$ is not located at the end of the model, it is still possible to provide a derivative value to the layer $\mathcal{F}^2_{w^{(2)}}$ by taking into account the modification of its output $\mathcal{F}^2_{w^{(2)}}(x)$ by the intermediate layer $\mathcal{F}^1_{w^{(1)}}$. This allows us to trace back the influence of each weight on the final output and it is the core of **backpropagation**. This efficient implementation of the chain rule builds a computational graph connecting the different layers of inputs and outputs and formalizes their influence on each other: in the previous example, it consists in getting the explicit formula of $\partial \left(\mathcal{F}^1_{w^{(1)}} \circ \mathcal{F}^2_{w^{(2)}}\right) / \partial w^{(i)}, \forall i \in [\![1, 2]\!]$. Then, during the optimization process that will be explained in the next section, the derivatives are computed starting from the final layer and progressively transmitted back to previous layers following the graph.

Parameters optimization

If the loss function provides a measurement of the "mistakes" made by the model, the "learning process" in itself is still missing in order to conceive a framework that allows the model to "learn from its mistakes".

As both the distance function $\mathbb{D}$ and the model $\mathcal{F}_W$ are differentiable, the risk function $\mathcal{L}$ is also differentiable. By applying Fermat's theorem in the context of convex differentiable functions optimization [3], the risk is minimal when its derivative with regards to the weights is null, i.e. $\nabla_W \mathcal{L}(W^*) = 0$.

To reach this minimal value, the idea is to "run away" from the risk increase by recursively changing the weights towards the opposite direction of the risk gradient $\nabla_W \mathcal{L}$. This process is conducted by what is commonly referred to as the **optimizer**, an iterative algorithm that encompasses a risk function $\mathcal{L}$ to minimize with a rule for updating the weights W accordingly. For example, equation 2.19 gives the simple **gradient descent** rule for updating the model weights at step $\kappa + 1$ of the optimization/training process:

$$W_{\kappa+1} \leftarrow W_\kappa - \mathrm{lr}_\kappa \times \nabla_W \mathcal{L}(W_\kappa), \tag{2.19}$$

with $\mathrm{lr}_\kappa \in \mathbb{R}^+$ the **learning rate**, eventually evolving with κ. Its value is decisive for the optimization success, as small values raise slower convergence and risks of being stuck in

[3] In practice, being differentiable *almost everywhere* is sufficient, by relying on sub-derivatives, local approximations or optimization tricks on non-differentiable points.

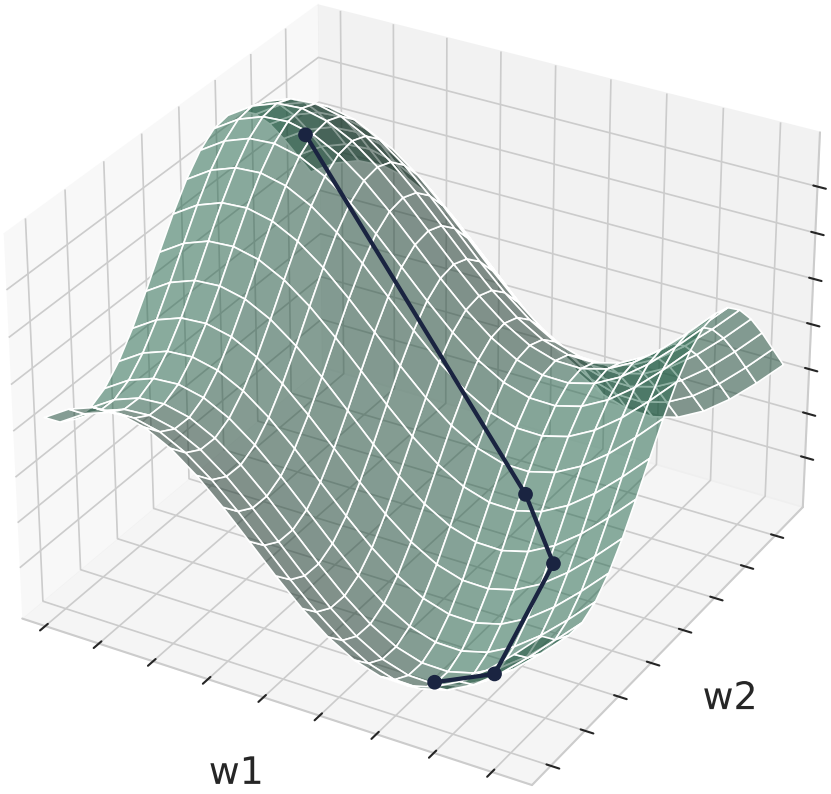


Figure 2.10: Optimization landscape (green surface) and optimization trajectory (dark blue line) of the gradient descent applied on a two-variable (w1,w2) risk function.

local minima, while large values might result in divergence of the optimization. Still in the case of convex optimization, if lr is set correctly and after iterating through a certain number of optimization steps, the weights should converge towards an approximation of the optimal values W^*. Plotting the risk with regards to the parameter space is the **optimization landscape**, which is illustrated in Figure 2.10 with an example of parameters optimization with gradient descent.

In practice, the optimization of real-world tasks is often non-convex, and more advanced optimizers are required to increase the success of the training. They are mostly based on **stochastic gradient descent**, which samples a batch of data $\{x_n, y_n\}_{n \leq N_{\text{batch}}} \sim \mathcal{P}$ at each step instead of covering the full distribution $\mathcal{P}$: this method noises the training by slightly changing the optimization landscape at each step and thus, avoids getting stuck in local minima. The Adam optimizer (Kingma and Ba, 2015) makes use of adaptive momentum in its update rule to make the training even more robust and has been widely adopted by the scientific community.

Datasets

In practice, it is usually impossible to provide the full distribution $\mathcal{P}$ as it is not exhaustively accessible. What is realistically done is the gathering of a finite set of **ground truth** data pairs $\{x_n, y_n\}_{n \leq N_{\text{dataset}}}$ through independent sampling of the distribution $\mathcal{P}$: this set is called the **dataset**. As the finite dataset is supposed to be representative of the distribution, all previous formulas are implemented using the mean operation over the dataset instead of the expectancy over the distribution. Training a model by parsing through the dataset once is commonly referred to as an **epoch**.

A discrepancy between the collected dataset and the target distribution often results in underwhelming performances of the trained model on real-world tasks. Such a discrepancy can emerge from:

- a **domain shift**: the features x that have been used are mismatched from what is available in real-world scenarios. The distribution underlying such a dataset is reduced and/or displaced compared to the target distribution. Models trained on synthetic data often present this issue.
- **noisy data**: the correspondence between (some) features x and the collected labels y is erroneous, creating a different implicit function $\mathcal{F}$ from the real-world.
- a **lack of data**: the quantity of collected data does not fully encompass the full complexity of the given task, resulting in sparsity in the distribution underlying the constituted dataset.

An ideal dataset should prevent the aforementioned issues by gathering as much quantity as possible of perfectly labeled real-world examples, with a representative diversity of both.

Generalization

An underlying characteristic of deep neural networks is their **capacity**, coming from the choice of layers, their parameterization, and their connections. Capacity is synonymous with expressiveness, as models with high capacities are able to fit more complex distributions.

The success in building a deep neural network depends on both: constituting a representative dataset of the given task and adapting the model capacity to the task complexity. Following the previously mentioned challenges in gathering the dataset, a mismatch between the model capacity and the task complexity is expressed by either the **underfitting** or **overfitting** of the model on the data. The former signifies that the neural network fails to model the distribution while the latter arises when the model fits the dataset "too well" and cannot generalize on distribution data outside of the sampled dataset.

Underfitting is relatively easy to identify as high loss values and (potential) evaluation metrics can indicate poor fitting of the model to the data. Overfitting on the other hand is usually monitored with **cross-validation** by splitting the dataset into multiple subsets, namely the **training** set, the **validation** set, and eventually the **test** set. In this setting, the model is only optimized with gradient descent techniques on the training set, while

its loss value and metrics on the validation set are kept track in parallel. During training, a similar decrease of both loss values on the training and validation sets indicates an adequate fitting to the data while maintaining generalization: as soon as the validation loss increases whereas the training loss still decreases, the model can be expected to start overfitting and it is recommended to stop the training procedure. This stopping criterion is commonly referred to as **validation-based early stopping**.

The test set represents a final set of data unseen during training to evaluate the trained model and it is used to prevent overfitting on the validation set through hyperparameter choices and to compare different methods trained on the same dataset.

2.4.2 Neural Networks

Artificial neural networks are commonly recognized as universal function approximators, implying that given sufficient layers and training data, they can approximate any function regardless of its complexity. Henceforth, deep learning is the sub-field of machine learning that uses networks comprised of multiple layers to model data distributions. Such layers are designed to be parallelizable, allowing for efficient utilization of hardware accelerators such as Graphical Processor Units (GPUs). With the continuous advancements in computational power facilitated by these accelerators, more complex tasks can be tackled by even deeper neural networks, or **neural models** as they will be referred to for the rest of the manuscript. The fundamental neural layers used for building the models presented in this thesis are explained in the following.

Dense layer

The **dense** layer, also called *fully-connected* layer or *single-layer perceptron*, is a linear matrix multiplication with an optional bias and a non-linear activation function at its output. Formally, it takes an input vector $x \in \mathbb{R}^{D_{\text{in}}}$ and outputs a vector $\mathcal{F}_{\text{Dense}}(x) \in \mathbb{R}^{D_{\text{out}}}$ such that:

$$\mathcal{F}_{\text{Dense}}(x) = \sigma(\mathbf{W}_A x + \mathbf{W}_B), \tag{2.20}$$

with $\mathbf{W}_A \in \mathbb{R}^{D_{\text{in}} \times D_{\text{out}}}$ the kernel weights, $\mathbf{W}_B \in \mathbb{R}^{D_{\text{out}}}$ the bias and σ the non-linear activation function. $\mathbf{W}_A$ and $\mathbf{W}_B$ are trainable with backpropagation as explained previously and the output dimension D_{out} is set by the practitioner when designing the overall neural model.

The activation function is the element that allows neural models to learn non-linear behaviors and relationships in the data. A wide variety of activation functions has been proposed in the deep learning literature: those mostly used in this thesis are listed in Table 2.1.

One can notice that the gradient value through Rectifier Linear Unit (ReLU) for negative input is zero, which can cause some weights to be "dead". To accommodate for this issue, Leaky ReLU is a variation of ReLU with a small non-zero slope $\alpha \in]0, 1[$ for negative values, which enables gradient information to always be transmitted during backpropagation.

Name	Definition	$\lim_{x\to-\infty}$	$\lim_{x\to+\infty}$
ReLU	$\max(0, x)$	0	$+\infty$
Leaky ReLU	$\max(\alpha x, x)$	$-\infty$	$+\infty$
Sigmoid	$1/(1+e^{-x})$	0	1
Hyperbolic tangent (tanh)	$(e^{x}-e^{-x})/(e^{x}+e^{-x})$	-1	1
Scaled Hyperbolic tangent ($\tanh^{\dagger+}$)	$(\tanh(x)+1)/2$	0	1

Table 2.1: Usual neural networks activation functions notably used in this manuscript.

Recurrent Neural Network (RNN) layers

Time-distributing dense layers allows processing sequential data by applying equation 2.20 sample-wise: $\mathcal{F}_{\text{Dense}}[x](t) \coloneqq \mathcal{F}_{\text{Dense}}(x(t))$ for $x \in \mathbb{R}^{T\times D_{\text{in}}}$. However, sequential data, such as sentences, signals, and videos, contain relationship information between the samples that time-distributed dense layers cannot model. To address this issue, RNNs reuse the previously computed output as a side input for computing a new output, which introduces a time dependency between the sequential elements. The simple RNN extends the dense layer parameterization with a recurrent kernel $\mathbf{W}_R \in \mathbb{R}^{D_{\text{out}}\times D_{\text{out}}}$. Given a sequence input $x \in \mathbb{R}^{T\times D_{\text{in}}}$, the RNN outputs $\forall t \in [\![0, T[\![$:

$$\mathcal{F}_{\text{RNN}}(x)(t) = \sigma(\mathbf{W}_A x(t) + \mathbf{W}_B + \mathbf{W}_R \mathcal{F}_{\text{RNN}}(x)(t-1)). \tag{2.21}$$

Computing the gradient for RNNs requires backpropagating the gradient through the layers and through the time axis: as the sequences can be quite long, vanishing or exploding gradient issues may arise (Bengio et al., 1993). The Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) layers employ different mechanisms (a forget gate and a gating mechanism respectively) to more effectively bypass irrelevant information from previous samples and allow for modeling longer sequences than the RNN layer. Their architectures involve introducing additional weights, with GRU having fewer total parameters than LSTM for the same input/output dimensions. Both have similar performances and are illustrated in comparison with the RNN layer in Figure 2.11.

Each previously introduced layer type can be **bidirectional** by concatenating the outputs of two recurrent layers, one processing the input sequence forward (from sequence index 0 to T) while the other processes it backward (from T to 0). This allows for a single output sample (at index t) to be computed using the information contained in both past (at indices $t' \leq t$) and future input samples (at indices $t' > t$).

Convolutional Neural Network (CNN) layer

Another way of processing multi-dimensional data, including sequences, can be done leveraging the discrete convolution operation presented in Section 2.2.3. Indeed, CNN layers (LeCun and Bengio, 1995) have proven to allow detecting structured patterns in the data by applying learned filters on neighboring elements, being chunks of signals, image crops, or video excerpts among others.

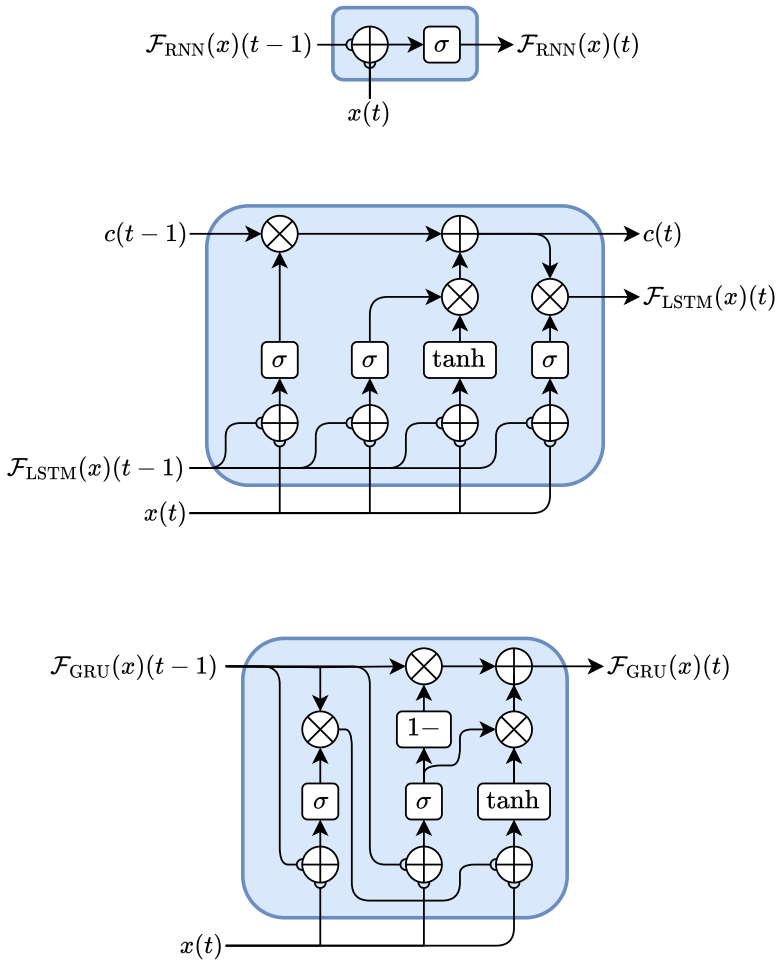


Figure 2.11: RNN, LSTM and GRU cells architectures. The cells process the input sample $x(t)$ using the previous output sample and by eventually updating their hidden state c. Filled arrows indicate the transmission of vectors as they are, while round connectors indicate an affine transformation of the vector by learnable weights at the operation entry point.

The 1D-convolution layer can be seen as a time-invariant FIR filter applied to a sequence of multidimensional features, with bias and non-linear activation. It has the same parameterization as the dense layer, with the exception of the kernel weights $\mathbf{W}_A \in \mathbb{R}^{Q \times D_{\text{in}} \times D_{\text{out}}}$ having an additional axis Q corresponding to the filter order, or *kernel size*. Given a sequence $x \in \mathbb{R}^{T \times D_{\text{in}}}$, the output of the CNN is:

$$\mathcal{F}_{\text{CNN}}(x)(t) = \sigma((\mathbf{W}_A * x)(t) + \mathbf{W}_B). \tag{2.22}$$

By adding other axes to the weights and applying the convolution operation along multiple axes, the CNN layer can be extended to data with higher ranks, such as images by convolving along the height and width, videos along the time, height and width axis, or 3D meshes along the height, width and depth axis.

Further parameterizations of the CNN layer can include the **stride** and **dilation rate**,

which are integer parameters controlling the way discrete convolution is applied. Striding downsamples the output by applying the CNN operation on frames $\{n \times T_{\text{stride}}\}_{n \in \mathbb{N}}$ of the input, while dilation applies the operation on the input downsampled/decimated by the chosen rate. Stacking multiple CNNs with striding or dilation allows increasing the receptive field of the model and learning larger structures in the data.

A common side operation accompanying CNN is **pooling**, which also downsamples the time/spatial dimensions output by the layer. It works by aggregating the CNN output features by only keeping the maximum (max-pooling) or mean value (mean-pooling) of a local neighborhood: for example, a mean-pooling of 3 along the time axis aggregates every chunk of 3 output samples into their mean value.

Regularization techniques

As mentioned previously, a discrepancy between the dataset size and the model capacity can result in a failure to correctly fit the data. To avoid overfitting issues, several regularization techniques and architecture designs have been proposed and have found success in mitigating the structural discrepancy between the deep neural models and the underlying ground-truth function, and have helped to stabilize the training process.

Weights regularization simply adds the magnitude of the model weights into the loss computation as a structural risk, encouraging the model to only use the necessary weights for the task and thus limiting its complexity. Computing the weights magnitudes is usually done with the L1 $\|.\|_1$, L2 $\|.\|_2$ or Frobenius norm.

Dropout (Srivastava et al., 2014) consists of randomly masking individual model weights during training, with a probability set by the practitioner. This technique encourages the neural model to learn more robust and generalizable features, by preventing it from being overly reliant on specific neurons or features.

A quick analysis of the formulas in 2.1 shows that the usual activation functions are centered around zero, meaning that their discriminating power is more efficient when the input also varies around zero. To ensure this property, normalization techniques aim to re-center and re-scale the layer outputs before applying the non-linear activation. The strategy for computing the scaling and centering factors is determined by the choice between batch normalization (Ioffe and Szegedy, 2015), layer normalization (Ba et al., 2016) and instance normalization (Ulyanov et al., 2016).

2.4.3 Generative Adversarial Networks

GAN is an advanced training method for generative deep neural networks that was introduced by Goodfellow et al. (2014). It has found success in various generative tasks as it has proven to be capable of producing high-quality examples compared to other regular training methods (Karras et al., 2021; Bond-Taylor et al., 2022).

Instead of designing a loss function that directly compares and measures the differences between the model outputs $\mathcal{F}_W(x)$ and the ground truth examples y, an auxiliary model $\mathcal{D}$ is jointly trained for this task. In this scheme, the generative model $\mathcal{F}_W$ is called **generator** while the auxiliary model $\mathcal{D}$ is the **discriminator**.

Both models are trained simultaneously with opposing objectives: the discriminator is fed with both outputs from $\mathcal{F}_W$ and ground-truth examples from $\mathcal{F}(X)$ and it is trained to discern artificially generated examples from real ones. On the other hand, the generator aims to maximize the errors made by the discriminator, by ideally producing examples close to those from $\mathcal{F}(x)$; or in simpler terms, as "real" as possible.

Formally, two losses $\mathcal{L}_{\mathcal{F}_W,\text{GAN}}$ and $\mathcal{L}_\mathcal{D}$ have to be designed for training each model. Different formulations of these losses have been proposed in the literature, one of which being the Least-Square (LS)-GAN variant (Mao et al., 2017):

$$\begin{aligned} \mathcal{L}_\mathcal{D} &= \mathop{\mathbb{E}}_{y\sim\mathcal{P}_\mathcal{Y}} \left[\|\mathcal{D}(y)-1\|_2\right] + \mathop{\mathbb{E}}_{x\sim\mathcal{P}_\mathcal{X}} \left[\|\mathcal{D}(\mathcal{F}_W(x))\|_2\right], \\ \mathcal{L}_{\mathcal{F}_W,\text{GAN}} &= \mathop{\mathbb{E}}_{x\sim\mathcal{P}_\mathcal{X}} \left[\|\mathcal{D}(\mathcal{F}_W(x))-1\|_2\right]. \end{aligned} \tag{2.23}$$

In this setting, one can notice that input and output data samples are never paired in the computation of each expectancy, which enables unsupervised training on unpaired input-output data, and even unlabeled data (by considering the input distribution as a Gaussian distribution for example). These losses can be interpreted as regression losses by labeling real examples as 1 and fake ones as 0: the discriminator is trained to output values close to 1 on provided ground truth examples, and close to 0 on synthesized examples. Concurrently, the generator reduces its loss value when the discriminator predicts values close to 1 on its generated outputs. According to Mao et al. (2017), formulating the GAN objectives as regression losses instead of classification losses helps to stabilize the training and improves the synthesis quality.

Indeed, the training instability is a well-known issue with GANs as the optimization landscape changes from one epoch to another. The models' capacities and their training dynamics have to be adjusted to avoid ill behaviors, such as **mode collapse**, where the generator can only output a poor diversity of samples compared to the full real data distribution. This phenomenon is an extreme consequence of the mode-seeking capability of GANs, in contrast with the mode-covering behavior of more traditional methods.

Nonetheless, focusing on modes in the data distribution promotes high-quality generation, and adversarial losses are now often used in conjunction with classical regular loss functions to refine the training objectives. From the generator perspective, the discriminator can be viewed as an adaptive loss that can pinpoint implicit data features that need to be matched but are not taken (enough) into account by the regular training objective. In this case, the LS-GAN formulation described in Equation 2.23 can be rewritten in a supervised learning setting:

$$\begin{aligned} \mathcal{L}_\mathcal{D} &= \mathop{\mathbb{E}}_{x,y\sim\mathcal{P}} \left[\|\mathcal{D}(y)-1\|_2 + \|\mathcal{D}(\mathcal{F}_W(x))\|_2\right], \\ \mathcal{L}_{\mathcal{F}_W,\text{GAN}} &= \mathop{\mathbb{E}}_{x,y\sim\mathcal{P}} \left[\|\mathcal{D}(\mathcal{F}_W(x))-1\|_2\right]. \end{aligned} \tag{2.24}$$

Exploiting this property, more complex adversarial training settings have been developed such as **multi-scaling** (Wang et al., 2018) that employs multiple discriminators, each focusing on a different scale of the data. **Feature matching** (Larsen et al., 2016) further helps to capture implicit data features by encouraging the synthesized data to have discriminator intermediate layers features similar to those from real data.

2.4.4 Domain-knowledge inclusion in Deep Neural Networks

The impressive modeling capabilities demonstrated by deep learning methods have fostered the development of dedicated hardware and infrastructures, as well as the gathering of even larger datasets. In turn, more sophisticated and higher-capacity neural models can be trained for handling complex tasks from raw data. However, building such large and complex models presents significant challenges concerning data accessibility and environmental impact during training, alongside ethical considerations in their application. Notably, training large models requires adequate infrastructures with high-energy consumption (Douwes, 2023), along with datasets of equivalent scales, which can be challenging to gather due to data scarcity and privacy concerns. Plus, naive deep learning models suffer from the "black-box" apprehension: as their high complexity make them difficult to understand and interpret their decision process, end-users can be reluctant in adopting them for sensitive tasks.

This raises the question of whether it is feasible to keep the high-quality modeling of deep learning methods while mitigating these concerns. An expansive research direction involves incorporating domain knowledge that was used by traditional methods. The term **hybrid deep learning** can be employed for designating this concept, which has helped to design more robust and interpretable models using limited amounts of data. Hybrid deep learning strategies can be introduced at various stages in the design of neural approaches (Dash et al., 2022; Shlezinger et al., 2023):

- with **data formatting**, known properties and relationships can be more easily exhibited, and thus extracted, through alternative encoding or slight preprocessing compared to raw data. Using the previous notations, this process reduces the complexity of the target function $\mathcal{F}$, making it simpler to match.

- efficient **data selection** with active learning (Ren et al., 2021a) and human-in-the-loop approaches (Wu et al., 2022a) can accelerate the learning process, by adjusting the data selection as the optimization progresses. This amounts to a knowledge-informed sampling of the distribution $\mathcal{P}$ to change the optimization landscape and steer the optimization trajectory.

- through **optimization constraints**, neural models are encouraged to comply to known behaviors, by designing loss functions $\mathcal{L}$ with knowledge-informed distances and/or differentiable constraints.

- motivated by their **structural capacity**, the practitioner can select specific neural layers to capitalize on their inherent advantages or properties, increasing the model capacity while keeping the same scale of weights number. For example, integrating attention mechanisms can aid in establishing connections between distant sequence samples.

- imposing **structural constraints** with pre-trained or highly structured layers, through transfer learning techniques and explicit implementation of phenomenons

respectively. These techniques reduce the output and the optimization spaces respectively, but bring the model closer to the target minimum W^*, which can alleviate the need for large data quantities and enable faster convergence.

Section Summary - Deep Learning for Music Processing

Thanks to the increase in computational power, deep learning has arisen as one of the most powerful optimization framework for modeling complex phenomenons and distributions $\mathcal{P}$, based solely on input-output observations (x, y). To minimize a task-specific loss function $\mathcal{L}$, the parameters W of a deep neural model $\mathcal{F}_W$ are optimized using gradient descent algorithms, such as Adam. Neural models can be designed with a wide variety of *differentiable* layers with non-linearities: the success of such an approach lies in the selection, parameterization, and arrangement of layers, along with problem formulation and motivated regularization to match the complexity of the task. Such choices can be further guided by domain knowledge techniques, favoring the development of more reliable, interpretable, and ultimately trustworthy models.

2.5 Technical Background for Music Processing, in short

Chapter summary - Technical background for music processing

This chapter introduced the technical and scientific background necessary for handling piano music synthesis tasks with deep neural networks. Music can be represented both in the symbolic modality, with notably the low-level MIDI protocol (as piano rolls or note sequences), and in the audio modality in the form of digital (discrete) signals. The latter encompasses theories and operations for analyzing and manipulating audio represented as either waveforms $y \in \mathbb{R}^{N_{\text{samples}}}$ or spectrograms $|\text{STFT}(y)|$: in particular, the spectral modeling paradigm aims to express a musical signal y as the sum of an additive signal composed of pure sinuses y^{additive} with a filtered noise y^{noise}. The piano instrument is built upon the principle of string vibration but exhibits specificities such as the displacement of partials $\{f_k\}_{k\in\mathbb{N}^*}$ and detuning δf_0 to account for the inharmonicity B of the strings in the polyphonic context. Regarding the methods employed in this thesis, deep neural networks show great capabilities and flexibility in learning from data observations. Parameterized by trainable weights $W \in \mathbb{R}^{N_{\text{weights}}}$ and built on differentiable layers, a neural model $\mathcal{F}_W$ can reproduce phenomenons by minimizing a loss function $\mathcal{L}$ with an optimizer. The neural model design and its training process can be supported with domain knowledge to increase its efficiency and interpretability while reducing the training duration.

Chapter 3

State-of-the-Art

In order to conceive new methods for tackling a research question, examining previous works is a fundamental step as they provide valuable knowledge and tools that can serve as foundations. This chapter aims to present the developed approaches for the tasks of interest of this thesis, being the polyphonic audio synthesis and the performance rendering tasks, in the specific case of piano music.

First, a condensed overview of the relevant datasets for either or both tasks is drawn in Section 3.1. Then, Sections 3.2 and 3.3 will report the musical audio synthesis and performance rendering tasks respectively. Each will start with a mathematical formalization of its task, introducing specific notations and completing those previously introduced in Chapter 2. In the following, existing methods for tackling them are unveiled and regrouped according to their methodology, followed by the proposed evaluation metrics for measuring their performances.

3.1 Piano Performance Datasets

The previous chapters hinted at the usage of data-driven methods in this thesis, and to say the least: as their name implies, they need data to be trained on. As explained in Section 2.4.1, a dataset is a sampling of a distribution that the data-driven methods try to model. Since it represents the phenomenon under examination, dataset collection is of utmost importance for the final performances of the models trained on it. An ideal dataset should be an unending pool of real-world examples, with perfect annotations by experts, or perfect input-output correspondences (depending on the task at hand). Benchmarking data-driven methods on a specific task is relevant when they are trained on the same set of data: publicly available datasets are thus preferred.

In practice, building a dataset often involves making trade-offs between the input data quality, the annotations/output data quality, and the quantity/diversity of them. For example, extracting labels from a large set of unlabeled data can be done with an automatic system to alleviate the need for human annotators, but the obtained pairs would be as reliable as the chosen system. In particular for music datasets, gathering and publicly releasing large quantities of data is complicated as copyright issues are all the

more prevalent.

The subject of this thesis is the synthesis of audio piano performances from symbolic compositions, by tackling both the performance rendering and piano sound synthesis tasks: we are particularly interested in publicly available datasets that provide piano music scores, piano performances as audio recordings or as MIDI sequences, or an aligned combination of them. An overview and comparison of these datasets are depicted in Table 3.1. In particular, they are grouped by the 3 methodologies for getting symbolic piano performances in MIDI:

- *Synthetic* MIDI performances are created artificially through manual edition or with an automatic non-human system, without playing any instrument.
- MIDI *Recorded* performances are acquired by letting pianists play on a device capable of recording their inputted controls.
- MIDI *Transcribed* performances are obtained by retrieving the instrument controls (by hand or with an automatic system) from audio performance recordings.

3.1.1 Synthetic MIDI Performance Datasets

A common ancestor of piano datasets is the Classical Piano MIDI (CPM) collection, also known the Piano-Midi.de[1] database. It provides synthetic MIDI performances of classical piano music pieces, with hand-made tempo and velocity curves made by Bernd Krueger from 1996 to 2018. A digital piano was used for synthesizing the MIDI performances into audio.

For constituting the MAPS dataset, Emiya et al. (2010) designed a set of MIDI files of individual notes at different velocities, usual and random combinations of chords, and the synthetic performances from the CPM collection. Audio rendition of these files was made with both recorded playbacks on Disklaviers and synthesizers from different commercial software. As seen in Table 3.1, the data quantity and the performance realism of the MAPS dataset have been surpassed by other datasets, but the instrument recordings quality and the targeted and documented individual notes and chords samples make it still relevant for piano related tasks.

Ycart and Benetos (2018) further enhanced this dataset with A-MAPS, by retrieving the original annotations of the CPM collection, such as tempo, time and key signatures, staffs, sustain pedal signal... This update allowed the MAPS dataset to be used for more Automatic Music Transcription (AMT)-related tasks, such as key and meter estimation, beat tracking and structure analysis.

3.1.2 MIDI Recorded Performance Datasets

Disklavier refers to the technology developed by the Yamaha piano brand that augments pianos with electro-mechanical sensors and solenoids. Modernizing the player pianos, or *self-playing pianos*, this technology allows for recording and playing back the keys and

[1] http://piano-midi.de/

Dataset	Sources	Composers	Performers	Unique Compositions	Symbolic Performances	Audio Performances	Annotations
CPM	-	26	1	?	S: 337 (23.2h)	S: 337 (23.2h)	
MAPS	CPM	20	1	30	S: 270 (18h)	S: 210 (14.5h) R: 60 (4.16h)	
A-MAPS	MAPS, CPM	20	1	30	S: 266 (17.7h)		Tempo, key and time signatures, quarter note length, staffs, pedal and text annotations
SMD	-	11	8	50	R: 50 (4.7h)	R: 50 (4.7h)	
Crest-MusePEDB 2	-	14	12	53*	R: 443 (?h)	R: 443 (?h)	Structure, performer intentions
MAESTRO	Piano e-competition	13	205	854	R: 1276 (198.7h)	R: 1276 (198.7h)	
ASAP	Piano e-competition, MuseScore	16	?	222*	R: 1586 (138.8h)	R: 520 (44.6h)	Beat, downbeat, key and time signatures
ACPAS	ASAP, A-MAPS, CPM	29	?	497 (222*)	S: 603 (40.1h) R: 1586 (138.8h)	S: 1611 (130.8h) R: 578 (49h)	Beat, downbeat, key and time signatures
GP	IMSLP, Youtube	2786	?	10855	T: 10855 (1237h)	R°: 10855 (1237h)	
GP-curated	GP	1787	?	7236	T: 7236 (875h)	R°: 7236 (875h)	
ATEPP	Spotify API, Youtube, ASAP, MuseScore	25	49	1580 (319*)	T: 11742 (1007h)	R°: 11742 (1007h)	
MazurkaBL	CHARM Project	1	135	44*	-	R: 2000 (110h)	Score-beat positions, loudness, dynamics and tempo markings
CIPI	Henle Verlag	29	0	652*	-	-	Difficulty level

Table 3.1: Overview of symbolic and audio piano music datasets. Compositions provided with MusicXML scores are notated with *. The symbolic statistics are sorted out between synthetic (S), recorded (R), and transcribed performances from audio (T). Audio statistics are also broken down between synthesized audio (S) and recorded audio (R), with ° indicating that the dataset consists of a collection of Youtube links.

pedal data on the instruments, enabling the recording of performances in both audio and MIDI formats.

The first realistic performance dataset leveraging this technology is the Saarland Music Data (SMD) dataset (Müller et al., 2011): it gathered 50 unique performances on Disklavier from piano students, amounting to around 5 hours of MIDI and audio-aligned classical piano music.

Hashida et al. (2018) created the second edition of the CrestMuse-PEDB dataset by recording more than 400 professional piano performances on Disklavier, with the majority of the 53 compositions being interpreted by at least 10 different pianists and with different playing styles. Supplementary annotations include the scores aligned at note-level with the performances and the performer intentions recovered from interviews.

The Minnesota International Piano-e-Competition[2], or commonly referred as the Yamaha Piano e-Competition, is an international piano competition where the performers play on Disklaviers and the jury evaluates their performances from another room with another playback Disklavier. All publicly available MIDI and audio performance pairs were gathered into the MIDI and Audio Edited for Synchronous TRacks and Organization (MAESTRO) dataset (Hawthorne et al., 2019), which amounts to around 200h of professional piano playing, spanning over $I = 10$ editions of the competition.

A similar endeavor is the Aligned Scores and Performances (ASAP) dataset (Foscarin et al., 2020) that gathered music scores (in MusicXML format) from the MuseScore Online library [3] and matched as much MIDI performances from the Yamaha Piano e-competition as possible. More than a thousand performances have been matched, with half of them intersecting the MAESTRO performances that have audio recordings. The scores are used to extract beats, downbeats, and time and key signature annotations. The (n)ASAP update (Peter et al., 2023) raised the score-to-performance alignment from beat-level to note-level.

The Aligned Clasical Piano Audio and Scores (ACPAS) dataset (Liu et al., 2021) compiled the ASAP, A-MAPS, and remaining CPM data into a 180h piano performances dataset. It has both synthetic (from CPM-based subsets) and real performances (from Yamaha Piano e-Competition) with either real audio recordings (from MAPS and MAESTRO) or rendered audio with a software synthesizer (completing the missing audio from ASAP). Scores and annotations previously available in ASAP and A-MAPS are kept.

3.1.3 MIDI Transcribed Performance Datasets

Thanks to the quantity and quality of the MIDI-recorded piano datasets, neural audio-to-MIDI piano performance transcription models (Hawthorne et al., 2018; Kong et al., 2021; Hawthorne et al., 2021) have achieved such impressive accuracy that they can be deployed for real-world tasks. New piano datasets are released by gathering large quantities of raw audio performances and applying such models to get their MIDI transcriptions.

For collecting the GiantMIDI-Piano (GP) dataset, Kong et al. (2022) crawled over the IMSLP website [4] to find the metadata of 10855 unique piano pieces. One audio performance per piece was retrieved from YouTube, to be then transcribed into MIDI using their AMT model (Kong et al., 2021) displaying a 96.72% onset F1-score. The

[2] https://piano-e-competition.com/
[3] https://musescore.com/sheetmusic
[4] https://imslp.org/

extracted metadata were noisy as some composers were assigned pieces they did not compose, the authors also designed a curated subset with a more rigorous matching rule, reducing to 7236 performances. The dataset provides the performances in their MIDI-transcription: to get the audio, users need to download the files themselves using the provided list of YouTube links.

The Automatically Transcribed Expressive Piano Performance (ATEPP) dataset (Zhang et al., 2022) follows a similar methodology, but puts an emphasis towards the performers instead of the pieces. By collecting the discography metadata of 49 renowned piano performers from the Spotify API, 11742 performances were downloaded from YouTube and then transcribed into MIDI. The transcription model is an improved version of (Kong et al., 2021) that also transcribes pedal data: elected through a listening test, it displays 92.1 % onset F1-Score. Finally, 319 out of the 1580 different pieces have their corresponding scores retrieved from ASAP and MuseScore. On the contrary of the GP dataset, multiple interpretations of the same pieces are available, which allows for modeling performer styles to an extent.

3.1.4 Other Piano Datasets

Symbolic performances are the main modality of this thesis scope, as it is both the output of performance rendering systems and the input to audio synthesizers. The following datasets unfortunately do not contain performances in the MIDI format, but they provide interesting data compilation that surpasses certain statistics from the previous datasets.

The MazurkaBL dataset (Kosta et al., 2018) gathered 2000 audio performances from the CHARM Chopin Mazurka Project and extracted the beat-level loudness and the beat positions in the performances. The original scores are the 44 Mazurkas of Chopin, each having on average around 40 performances in the dataset. MazurkaBL is highly interesting for studying variations in performer styles, with however limited diversity of music pieces and composers. Also, the beat/loudness annotations are not sufficient for reconstructing the MIDI performances from the scores as they do not encode the way individual notes in chords are broken down during the interpretation.

Concurrently, there has been a growing interest in the task of piano score difficulty assessment. In order to train models for this task, Ramoneda et al. (2024) have constituted the "Can I Play It? (CIPI)" dataset, which contains 652 piano scores in MusicXML format (744 when splitting the movements) from different public sources (Mutopia, MuseScore, Craig Sapp) matched with the catalog of a renowned publisher. The difficulty labels are also recovered from the catalog, with manual validation by an expert annotator. It represents, as of writing, the largest dataset of classical piano scores.

Section summary - Piano Performance Datasets

Datasets are crucial for the training and comparison of data-driven methods: performances of the model are linked to the quality and diversity of the data they have been trained on. For piano audio synthesis, datasets collected using the Disklavier technology will be preferred as they provide real piano recordings paired with aligned MIDI inputs. MAPS and MAESTRO will be the go-to datasets for supplying recordings of individual notes, as well as recordings of polyphonic performances, respectively. Traditionally, performance rendering models would need aligned scores and symbolic performances for training, leaving (n)ASAP as the best candidate. However, this thesis aims to tackle performance rendering without alignments between the scores, MIDI performances, and audio performances. For each modality, the largest datasets while being as realistic as possible would be CIPI for piano scores, ASAP and MAESTRO for recorded MIDI performances, and ATEPP or non-curated GP for audio performances.

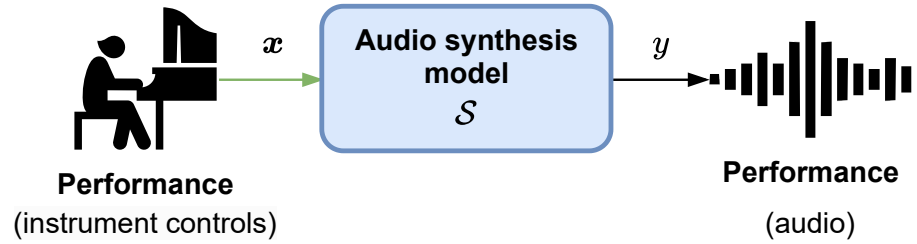


Figure 3.1: The conditional audio synthesis task. For the case of piano, the input instrument controls are usually encoded in MIDI.

3.2 Polyphonic Instrument Audio Synthesis

Figure 3.1 depicts the musical instrument sound synthesis task. That task objective is to design a digital synthesizer $\mathcal{S}$ that, given an input sequence x of notes/controls, generates an audio signal $\hat{y} \in \mathbb{R}^{N_{\text{samples}}}$ that mimics the sound produced by a real musical instrument:

$$\hat{y} = \mathcal{S}(x). \tag{3.1}$$

The information and the encoding of the input sequence depends on the chosen instrument to be modeled, and on the design and methodology underlying the synthesis model. For the case of piano audio synthesis, the usual controls are the sequence of eventually overlapping notes (parameterized by their onset time, their duration, and their velocity) and the pedal actions, which all can be encoded in MIDI.

As for the synthesis strategies, numerous methods have been proposed, each with varying complexity, needs for data, and overall quality. This section will briefly present them, following the taxonomy of Hayes et al. (2024) shown in Figure 3.2. Methods are grouped by whether the modeling relies more on finding parameters for a tailor-made model of the instrument (*parametric models*), or more on fitting a generic sound synthesizer to a set of audio examples of the targeted instrument (*data-driven models*). These two methodologies are not incompatible and several techniques can blend them: in particular, a closer look will be given to the Differentiable Digital Signal Processing (DDSP) framework. Finally, a summary of the proposed metrics for evaluating sound synthesis models will be given.

3.2.1 Parametric models

Parametric models are synthesis models that incorporate prior knowledge into their mathematical formulation. The use of such prior information is expected to have two effects: on one hand, the number of parameters that have to be adapted to fit a particular instrument is reduced, and on the other hand, the models are specialized for a particular type of instrument, and therefore can not be used to represent instruments that do not satisfy the included prior information. Physical models and signal models both belong into the category of parametric models, differing with respect to the amount of knowledge that is engaged to build them, and also with respect to their generality.

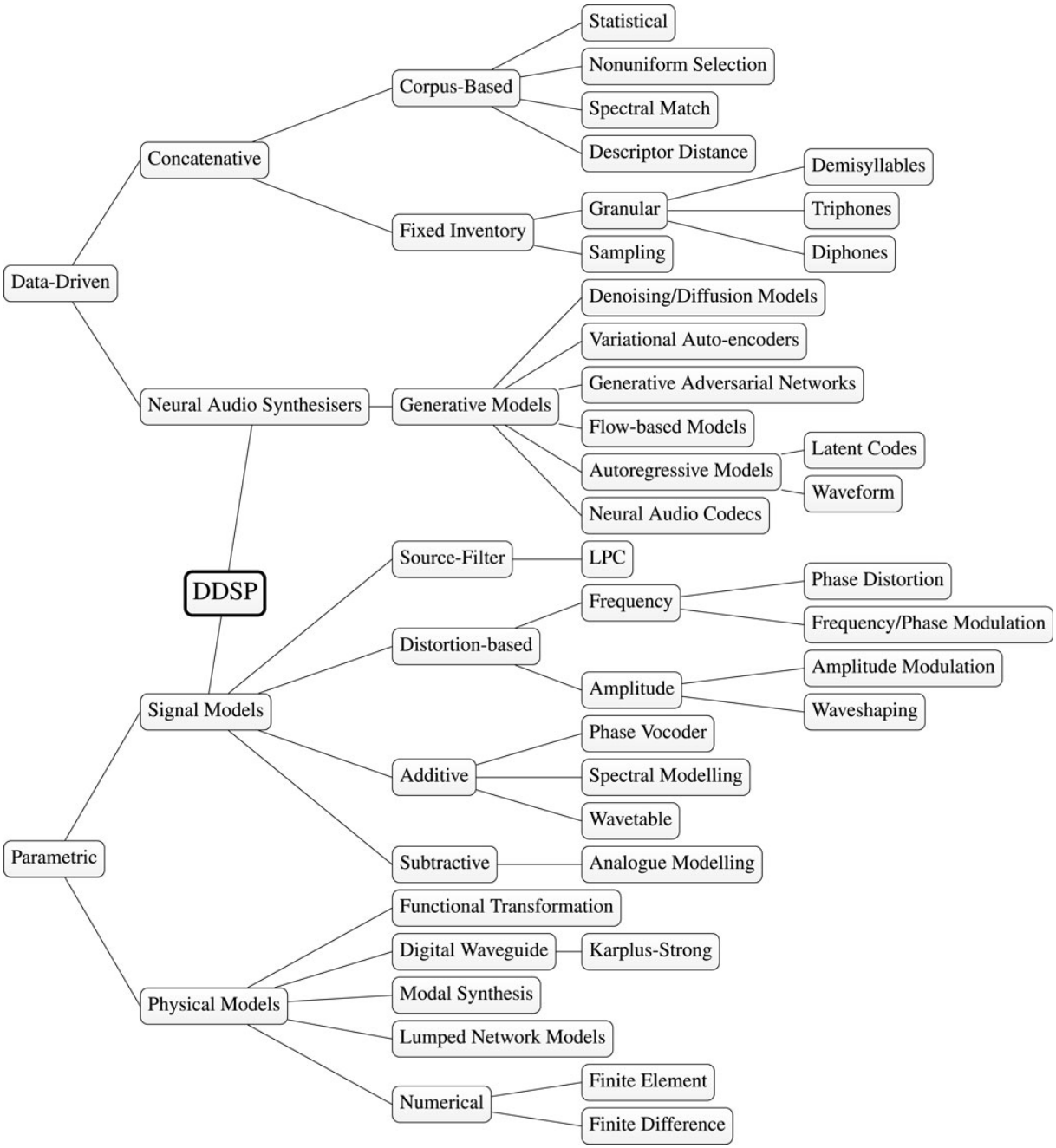


Figure 3.2: High-level taxonomy for popular sound synthesis techniques, updated by Hayes et al. (2024) from Schwarz (2006) and Bilbao (2009). It illustrates high-level relationships between the techniques and does not intend to make an exhaustive listing of them.

Physical Modeling

Physical models aim to accurately imitate the physical phenomena happening in a musical instrument. They require to explicitly formulate and solve, to an extent, some equations governing the behavior of the instrument being played, such as the displacement and motion of physical elements, the propagation of the energy, their transmission and radiation

as sound, etc... The solving or simulation of these equations can be done through modal analysis and synthesis, digital waveguides, finite difference schemes, Hamiltonian port systems, mass-spring networks,... An extensive review of these methods can be found in Bilbao (2009).

The trade-off in most physical modeling systems is twofold. First, the modeling accuracy improves as the equations are more extensive, taking more physical details into account and limiting the mathematical approximations. For example, the physical modeling of the piano done in Chabassier (2012) requires explicitly formulating the behavior and the numerous couplings of all elements in the instrument (from the string to the soundboard through the hammers), plus the retrieval of all physical properties, such as the soundboard dimensions and rigidity, the string tensions and dimensions, the crushing coefficients of the hammers, etc... This endeavor requires a consequent amount of work from the practitioner that is not easily scalable. Nonetheless, if done correctly, the models can achieve high synthesis quality with extensive controllability and interpretability.

The other trade-off lies in the numerical solving of the established equations. Similar to audio sampling concerns mentioned in Section 2.2.1, discretization of the wave propagation equations in time and space discards high-frequency information that can be relevant to the overall quality. Increasing the respective sampling rates can increase the simulation accuracy but at the cost of a longer duration of the process, and it is prone to approximation error accumulation. For practical usages, the optimization can be run as a development step to get the parameters of sound synthesizers that can be less grounded to the physical reality but are faster and more robust during inference, such as digital waveguides (Rauhala et al., 2008) and additive synthesis (Bank and Chabassier, 2019).

Signal-based Modeling

Signal-based methods leverage the signal processing operations, partially presented in Section 2.2, to synthesize audio. As shown in Figure 3.2, a large array of synthesis operations have been developed, ranging from linear additive and subtractive synthesis to complex and non-predictable modulation operations. While mostly used as engines in analog or digital synthesizers for producing sounds from scratch, they can also be employed for reproducing target sounds and, by extension, replicating a musical instrument. In the latter case, the choice of synthesis method can be motivated by knowledge of the instrument to be modeled: for instance, source-filter models have been prevalent for speech synthesis (Dudley, 1939) as they imitate the filtering of the glottal pulses by the vocal tract.

More relevant for this thesis, spectral modeling (Serra and Smith, 1990), as presented in Section 2.2.4, fits the modal behavior of pitched instruments and has thus been widely investigated for instrument sound synthesis. The underlying sines-plus-noise synthesis is still highly expressive and flexible, and the challenge lies in finding the correct parameterization of the synthesizer such that it sounds like the target instrument. Insufficient or too simplistic parameters often yield a "synthetic" sound that lacks realism. As mentioned previously, physical models are constructed using an understanding of the physical interactions that are taking place in the instrument. For spectral modeling, the parameters for a given sound can be obtained through analysis of a few audio recordings of the instru-

ment with signal processing tools: this is commonly referred to as the *analysis/synthesis* technique.

The time-varying parameters for a sines-plus-noise model can be retrieved from a time-frequency representation of the sounds (Keiler and Marchand, 2002), as done with PARSHL (Smith and Serra, 1987), least-squares methods (Stylianou, 1996) or derivatives methods (Xue and Sandler, 2009). Temporal connection between the estimated partials can be reinforced using partial tracking methods (Serra, 1990; Lagrange et al., 2007), while (quasi-)harmonic constraints can be applied for the estimated frequencies (Rigaud et al., 2011; Hahn and Roebel, 2013). Other parameter extraction methods that do not rely on time-frequency representations leverage, instead, subspace exploration (Roy and Kailath, 1989; Badeau, 2005) or atomic prototypes (Mallat and Zhang, 1993).

3.2.2 Data-driven models

While exploited to some extent by signal-based models, data-driven models put a larger emphasis on reproducing the musical instrument outputs by manipulating a larger set of audio recordings. *Concatenative synthesis* and *neural synthesis* both require such data quantity for operating, with the benefits of not needing much a priori knowledge and can thus be highly adaptable to different instruments.

Sample-based Synthesis

Among the different concatenative synthesis techniques, *sampling-based* synthesis is widely used in the industry, especially for digital piano. As detailed in the high-level taxonomy of Hayes et al. (2024) Figure 3.2, the core principle of concatenative synthesis is to organize an ensemble of audio recordings and to play them back, entirely or in chunks, according to the user inputs (Schwarz, 2006). The algorithm for sample selection can be a simple mapping from one button to a stored sample (as it can be found in drum machines), but it can also integrate more advanced rules based on the sample characteristics (for automatic granular synthesis notably).

For musical instrument sound synthesis from MIDI, one should gather individual note recordings across various pitches, velocities and, eventually, playing styles (pertinent for string instruments for instance). The recordings are segmented and played according to the sound regimes: the attack part is played during MIDI note onsets, the sustained part is looped while the note is playing, and the release part is played when the note concludes. An interpolation algorithm between samples enables the synthesis of pitch-velocity-style configurations that are not present in the dataset.

The main benefit of sample-based synthesis is its good sound quality, which is due to the fact that real recordings are played directly. Increasing the number of recordings evens out the realism across the wide ranges of pitches, velocities, and playing styles. Notably, swapping between multiple samples for the same pitch-velocity configuration helps to simulate the stochastic behavior of a real instrument, avoiding the unrealistic "copy-paste" sound. However, these techniques also entail a proportional increase in memory usage, which can become significant: professional sound libraries, or *sound banks*, often accumulate several gigabytes of audio recordings. Also, in polyphonic contexts, the

playback of simultaneous individual notes often fails to incorporate mutual interactions, such as the sympathetic resonances for piano.

Neural Audio Synthesis

Following their success in modeling and generating images (Van Den Oord et al., 2016) and texts (Jozefowicz et al., 2016), deep neural networks have been adapted for also synthesizing audio signals. The breakthrough for Neural Audio Synthesis (NAS) was the WaveNet model (Oord et al., 2016) that can generate speech and music signals with unprecedented quality. The main principle behind WaveNet is to generate a quantized waveform sample autoregressively (i.e. by using previous outputs as side inputs for predicting the next sample) using stacks of dilated convolutional layers (see Section 2.4.2). This model paved the way for audio synthesis using deep neural networks, mainly through the design of new approaches that could speed up the training and generation processes, reduce the necessary amount of model parameters and training data, and/or increase the model controllability; all while maintaining, or even surpassing, the synthesis quality.

Notable neural models synthesizing raw audio as waveform unconditionally, from linguistic features or from musical attributes, include ParallelWaveNet (Oord et al., 2018), SampleRNN (Mehri et al., 2017), the WaveNet auto-encoder (Engel et al., 2017), SING (Défossez et al., 2018) and WaveGAN (Donahue et al., 2019). More recently, the RAVE model (Caillon and Esling, 2021) addresses the multiple scales of audio modeling by employing a multi-band decomposition.

Indeed, modeling audio signals directly as raw waveform is a tedious endeavor because of their very high dimensionality (at least 8000 samples per second for low-quality audio). To circumvent such complexity, a common approach consists of modeling audio through a compressed representation: a model is trained for outputting audio in such representation for a high-level modeling task (such as Text-to-Speech (TTS), MIDI synthesis or timbre transfer), while a low-level algorithm/model is responsible for decompressing this representation into a hearable waveform. Widely used representations are time-frequency representations, such as the amplitude Mel-spectrogram: as seen in Section 2.2.2, they are easily extractable from waveforms while preserving the most perceptually relevant features of sounds and reducing the sequence length. As a side benefit, time-frequency representations can be considered as images, thus facilitating the adaptation of CV models for audio[5].

Remarkable works in the modeling of high and mid-level audio scales through time-frequency representations include GANSynth (Engel et al., 2019), the perceptually-motivated variational auto-encoder from Esling et al. (2018) and TTS acoustic models such as the Tacotron models (Wang et al., 2017; Shen et al., 2018; Elias et al., 2021), FastSpeech (Ren et al., 2021b) and Transformer-TTS (Li et al., 2019). In almost all these works[6], audio is manipulated through the magnitude of a time-frequency representation, discarding the phase information: therefore, a supplementary vocoder/waveform model is needed

[5]Images and amplitude spectrograms are similar in being 2D data, but their information distribution are different: spectrograms time and frequency axis have distinct meanings on the contrary of the two spatial axes of images.

[6]With the exception of GANSynth that model both spectrum magnitude and a derivative of the phase.

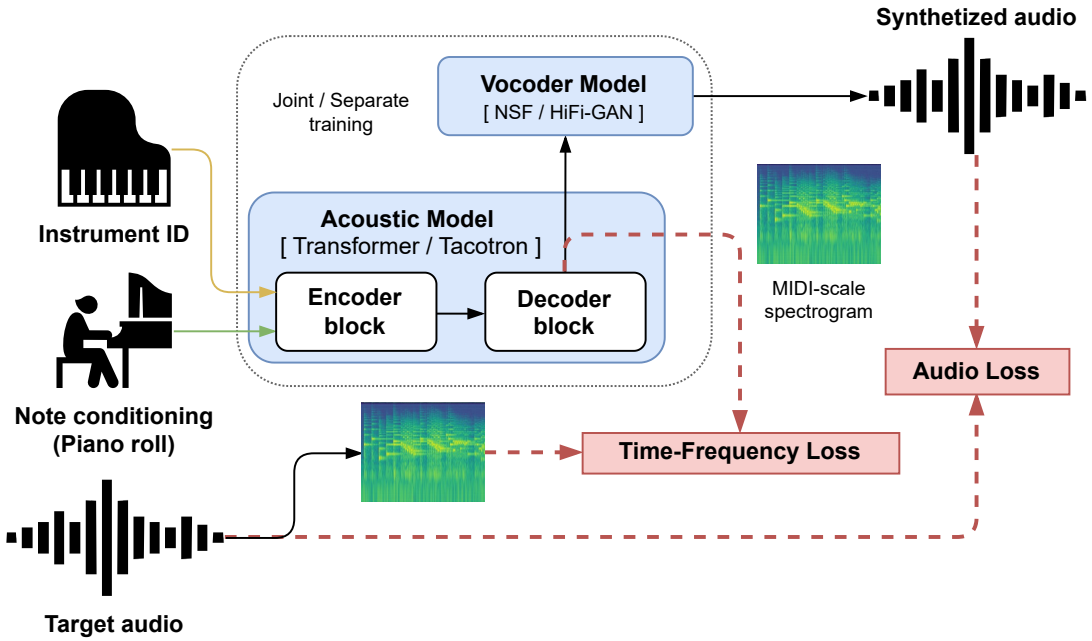


Figure 3.3: Architecture of the Piano-TTS baseline models from Cooper et al. (2021) and Shi et al. (2023): the first version employs a separate training of a Tacotron and a NSF model, while the update employs a joint training of a Transformer-TTS and HiFi-GAN model. MIDI-filter-bank-based spectra are used as an intermediate representation of audio between the acoustic and vocoder models.

to reconstruct realistic audio signals from the sole magnitude information. The traditional benchmark is the Griffin-Lim algorithm (Griffin and Lim, 1984), which delivers underwhelming sound quality. Neural vocoders include MelGAN (Kumar et al., 2019) and Parallel WaveGAN (Yamamoto et al., 2020), WaveGlow (Prenger et al., 2019), HiFi-GAN (Kong et al., 2020), WaveGrad (Chen et al., 2021) and source-filter inspired models, such as Neural Source Filter (NSF) (Wang et al., 2019) and Multi-band Excited WaveNet (Roebel and Bous, 2022).

Ultimately, neural models themselves have been employed for learning custom representations of audio that are more compressive and easily revertible. The forefront of modern neural codecs are SoundStream (Zeghidour et al., 2021) and EnCodec (Défossez et al., 2023), which are notably exploited in recent text-to-music generation models (Copet et al., 2023; Agostinelli et al., 2023).

For the specific task of instrument audio synthesis from MIDI, one can use neural models trained for single note synthesis, such as SING and GANSynth. Other works have directly tackled the task in the polyphonic context by adapting TTS models, the first being Hawthorne et al. (2019) that trained a WaveNet model on the MAESTRO dataset. The combination of an acoustic model followed by a waveform vocoder has been studied in PerformanceNet (Wang and Yang, 2019), Mel2Mel (Kim et al., 2019), Deep Performer (Dong et al., 2022), the Variational Auto-Encoder (VAE) of Tan et al. (2020) and the multi-instrument diffusion model by Hawthorne et al. (2022).

In particular, Cooper et al. (2021), then Shi et al. (2023), adapted many configurations of TTS models for piano audio synthesis from MIDI inputs represented as piano rolls. They notably proposed a modified time-frequency representation of audio by using filter banks centered around the MIDI note frequencies instead of the Mel frequencies. Tested acoustic model architectures were Tacotron-2 (Shen et al., 2018), PerformanceNet (Wang and Yang, 2019) and Transformer-TTS (Li et al., 2019), while the tested vocoder model architectures were NSF (Wang et al., 2019), HiFi-GAN (Kong et al., 2020), and a combination of both. According to the evaluations, the best configuration in the first study Cooper et al. (2021) was the modified Tacotron-2 acoustic model followed by a simplified NSF vocoder. In Shi et al. (2023), the best results were achieved using joint training of the Transformer acoustic model and the HiFi-GAN vocoder. These specific models will be referred to as **Piano-TTS v1** and **Piano-TTS v2** and are depicted in Figure 3.3.

3.2.3 Differentiable Digital Signal Processing

Differentiable Digital Signal Processing (DDSP) is a recent approach to neural audio modeling that integrates traditional signal processing operations into deep neural networks. The name Differentiable Digital Signal Processing (DDSP) originates from the work of Engel et al. (2020a) where additive synthesis, subtractive synthesis (with LTV filters), and reverberation (with a LTI filter) were implemented in a DL framework (Tensorflow) with automatic differentiation, making these operations differentiable with respect to their control and signal inputs. As seen in Section 2.4.1, the differentiability allows for gradient backpropagation from the loss function through the audio output back to the controls and signal inputs. Hence, a neural network can be trained for audio synthesis by predicting synthesizer controls, instead of raw waveforms, time-frequency representations, or discrete tokens to be decoded by neural codecs.

The original work by Engel et al. (2020a) illustrates the possibilities enabled by the approach that can be understood as a modern variation of the spectral modeling paradigm (presented in Section 2.2.4) applied to the task of timbre transfer, i.e. changing the instrument in a recording while preserving the musical content (namely the notes). As shown in Figure 3.4, fundamental frequency and loudness features are extracted from the input audio and the decoder model (re-)applies an instrument timbre by outputting parameters for the differentiable sines-plus-noise synthesizer. Notably, three key elements within the model set the tone for the following DDSP-based models:

- The neural decoder outputs are scaled into positive-value synthesizer controls by means of a modified sigmoid function. With the original sigmoid function defined in Table 2.1, this variant is here notated sigmoid† and defined as:

$$\text{sigmoid}^\dagger : x \in \mathbb{R} \mapsto 2 \times \text{sigmoid}(x)^{\log 10} + 10^{-7} \tag{3.2}$$

- The model is optimized to reconstruct the audio signal through the Multi-Scale Spectral (MSS) loss $\mathcal{L}_{\text{MSS}}$, defined in Equation 3.3. Since its introduction by Wang et al. (2019) following the proposition of Défossez et al. (2018), it has been a popular

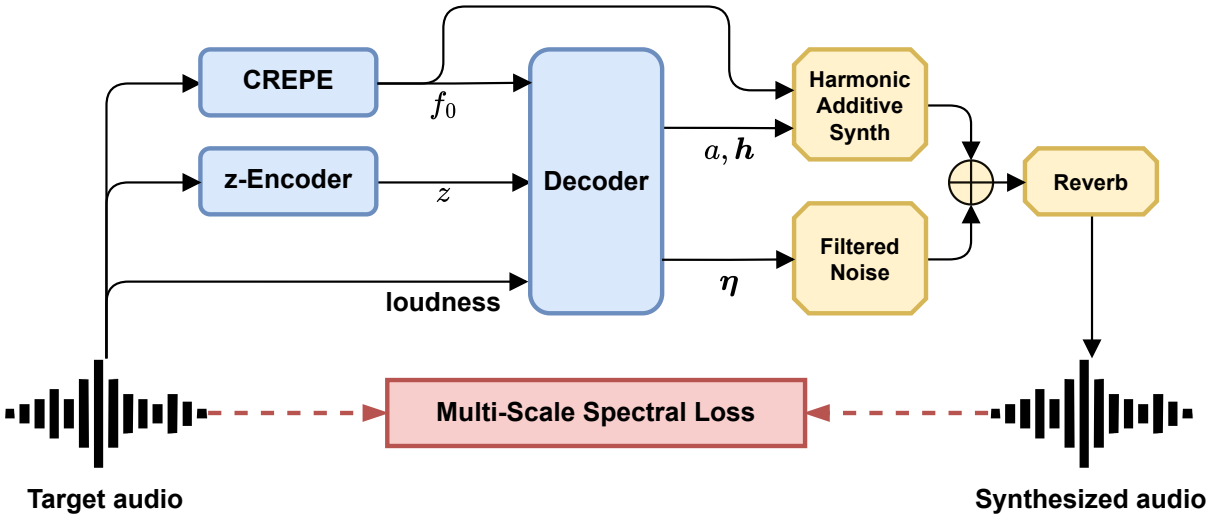


Figure 3.4: The original Differentiable Digital Signal Processing (DDSP) auto-encoder proposed by Engel et al. (2020a) for the task of timbre transfer. Fundamental frequency f_0, embedding z, and loudness controls are extracted from the target signal. The neural decoder processes them to output harmonic amplitudes $a, \boldsymbol{h}$, and filter coefficients $\boldsymbol{\eta}$ controls for the differentiable synthesizers (in yellow). Re-adapted from Engel et al. (2020a).

loss function for training neural models outputting audio waveform. It compares the linear and log magnitude spectrograms of the target signal y and the synthesized signal $\hat{y}$ at multiple DFT sizes $m \in \{2048, 1024, 512, 256, 128, 64\}$, as defined in Equation 2.5. Combining multiple DFT resolutions mitigates the time-frequency trade-off of STFT, previously mentioned in Section 2.2.2.

$$\begin{aligned}\mathcal{L}_{\text{MSS}}(y, \hat{y}) = \sum_m &\| |\text{STFT}_m(y)| - |\text{STFT}_m(\hat{y})| \|_1 \\ &+ \| \log |\text{STFT}_m(y)| - \log |\text{STFT}_m(\hat{y})| \|_1\end{aligned} \tag{3.3}$$

- Partials are constrained to be harmonics of the fundamental frequency, which is suitable for the modeled instruments (violin, saxophone, and flute) and alleviates the need for estimating their frequencies.

DDSP is located at the crossroads of several research directions. With regards to the audio synthesis task, it can be viewed as an enhancement of signal-based modeling (Section 3.2.1) with the neural optimization framework (Section 3.2.2). In particular, it delegates the parameter estimation task to neural networks that are more expressive and able to learn complex and non-linear behaviors. Deep learning also broadens the scope of applications of signal-based modeling to unprecedented tasks that would otherwise require hand-crafted features, for instance. From the pure deep learning optimization point of view, following Section 2.4.4, DDSP can be seen as domain-knowledge inclusion through structural constraints: the DDSP components are designed to exhibit and to

take advantage of known properties of audio signals, such as periodicity and harmonicity. As these strong priors on the sound structure are introduced, the amount of training data and model parameters can be significantly reduced and the synthesis process is more interpretable. DDSP-based models can also be more easily integrated into real-world applications by leveraging the existing efficient implementations of signal-based modeling[7].

The following will give an overview of the applications derived from this principle of integrating signal processing knowledge as differentiable operations. Note that constraining the neural model structures also creates new challenges, notably for their optimization. Both are discussed more extensively in the literature review of Hayes et al. (2024).

Short review of DDSP-based models

Following the release of DDSP, numerous works have been published that revisit traditional signal-based operations and methods and integrate them as differentiable layers in neural models.

Early adaptation of the framework expanded the synthesis of harmonic sounds with methods other than the sines-plus-noise model, namely with frequency modulation (Caspe et al., 2022), wavetables (Shan et al., 2022) and waveshapers (Hayes et al., 2021). Non-harmonic and polyphonic instruments were also modeled, such as percussive instruments (Diaz et al., 2023; Shier et al., 2023), guitars (Wiggins and Kim, 2023; Jonason et al., 2023) and sound effects (Barahona-Ríos and Collins, 2024; Liu et al., 2023). Early works employing an oscillator as the source and neural networks as filters (Wang et al., 2019; Michelashvili and Wolf, 2020) can also be considered as DDSP methods a posteriori.

Classical audio effects were also revamped by implementing usual signal processing operations as differentiable functions. Linear filters with IIR filters (Kuznetsov et al., 2020) can be done by leveraging their similarity (Equation 2.8) with the RNN formulation (Equation 2.21). Comparing Equations 2.7 and 2.22, FIR filters are also clearly differentiable since CNNs are based on them: notably, FIR approximation of IIR filters can be done through the frequency sampling method (Nercessian, 2020). Longer filters also enable the differentiability of artificial reverberation models (Lee et al., 2022; Santo et al., 2023). Modulation effects can be modeled after the estimation of their LFO control signal (Mitcheltree et al., 2023; Carson et al., 2023). As for non-linear effects, differentiable distortions (Esqueda et al., 2021; Colonel et al., 2022) and the recording through tapes (Mikkonen et al., 2023) have also been developed.

Beyond audio synthesis and effect modeling, DDSP-based methods can also be employed for more complex tasks previously addressed by pure neural models. Several works made DDSP controllable by MIDI while aiming for a more expressive synthesis (Jonason et al., 2020; Castellon et al., 2020; Wu et al., 2022b). Furthermore, since DDSP components already exhibit typical behaviors found in real audio, they can be used to gather a rather unlimited amount of control/audio data pairs for self-supervised sound-matching tasks: re-implemented as differentiable synthesizers/effects, it has been done with spectral modeling (Engel et al., 2020b), synthesizers (Masuda and Saito, 2023; Uzrad et al., 2024),

[7] https://github.com/magenta/ddsp-vst

physical-based synthesis (Han et al., 2023) and full signal effect chains (Steinmetz et al., 2022; Colonel and Reiss, 2021). As for Music Information Retrieval (MIR) tasks, differentiable components were used for tackling bandwidth extension (Grumiaux and Lagrange, 2023) and source-separation (Kawamura et al., 2022; Schulze-Forster et al., 2023; Richard et al., 2024).

In the context of instrument sound synthesis, our work (Renault et al., 2022) was the first to adapt DDSP for the piano and has served as a basis for Simionato et al. (2024) and Berendes et al. (2023).

Frequency Estimation in DDSP-based Synthesis

Simply implementing signal processing operations in a framework with auto-differentiation is not always sufficient for successfully integrating them into the neural optimization scheme. For example, optimizing IIR filters directly can be slow and even unstable as the gradient computation has to be computed through time. Truncated backpropagation (Kuznetsov et al., 2020) and frequency sampling (Nercessian, 2020) help dealing with both issues.

Arguably the most challenging issue with the hybrid approach, which is still unsolved at the time of writing, is the estimation of frequencies by gradient descent. As shown experimentally by Turian and Henry (2020), the optimization of a differentiable oscillator is non-convex with regards to its frequency parameter: the gradient of usual spectral-based losses, such as the MSS $\mathcal{L}_{\text{MSS}}$, does not provide useful information for retrieving a target frequency.

This issue has been circumvented by relying on external frequency estimators in multi-pitch (Schulze-Forster et al., 2023; Richard et al., 2024) and monophonic settings (Engel et al., 2020b), eventually with the harmonic constraint and by only predicting the fundamental frequency with CREPE (Kim et al., 2018), as in the original work of Engel et al. (2020a). Caspe et al. (2022) have also manually selected pre-configuration schemes to set the frequencies closer to the target ones.

Recently, several works have explored different paths in order to conceive a robust optimization configuration that can estimate the frequencies through an unconstrained additive synthesizer. Hayes et al. (2023) proposed to re-introduce convexity in the optimization landscape by adding an additional parameter to the oscillator: an exponential damping factor. In single and multiple sinuses cases, they report better frequency estimation than with the default undamped configuration but also acknowledge the difficulty in estimating low amplitude sinuses. Schwär and Müller (2023) made an extensive exploration of the different MSS parametrization for the task, namely the choice of window type and size, the magnitude compression method (linear/log/others), and the distance norm order. They reveal a sensitivity of the optimization convergence with regards to these parameters: mainly, choosing a window function that increases the overlapping between the main frequency lobe and reduces the spectral leakage in side lobes is beneficial. Finally, Torres et al. (2024) leveraged optimal transport operations (Cazelles et al., 2021; Latorre et al., 2023) in the spectral domain. By measuring the cost for displacing in frequency the magnitude spectrum of the estimated audio to the target spectrum, they would obtain a gradient indicating the frequency shift necessary for matching both spec-

tra. In conjunction with the MSS loss, they have better optimization stability than with the MSS alone, but they still report difficulties when dealing with sounds that also exhibit non-sinuses components, such as noise, transient, etc...

3.2.4 Evaluating Sound Synthesis

Evaluation metrics are essential in order to compare the results of different methods. For instrument modeling, the perfect model would be able to reproduce the audio generated by the target instrument given the same controls: data-driven approaches conveniently offer sets of recordings to compare to. Concurrently, assessing the audio quality of a synthesis model is inherently linked to human perception and thus, to subjectivity (Hayes et al., 2024). Hence, both objective metrics in the form of audio reconstruction measures, and subjective metrics through listening tests, have been employed for evaluating sound synthesis results.

Objective Evaluation Metrics

Objective evaluation computes systematic metrics on outputs of a given model, compared eventually to a reference model or ground truth outputs. It can be applied model-wise, and results can be comparable from one study to another, assuming the evaluation conditions (hardware, test sets,...) are preserved.

The most intuitive metric for assessing instrument audio synthesis is to measure the reconstruction of target audio recordings, given the same inputs. To this end, ground-truth control and audio pairs have to be gathered into a test set, unseen during training, which can be extracted beforehand from a dataset for data-driven methods. Several audio similarity metrics have been proposed throughout the literature. There is no clear consensus regarding the distance to be used, with the exception of disregarding waveform similarity measures that penalize phase differences that are not perceptible.

Spectral-based distances are commonly used for objective evaluation, such as the MSS distance (also used as a loss function, defined in Equation 3.3) (Défossez et al., 2018; Barahona-Ríos and Collins, 2024; Han et al., 2023; Jonason et al., 2023), Joint Time-Frequency Scattering (JTFS) (Han et al., 2023) or for a single DFT size, log-spectral distance (Hayes et al., 2023; Torres et al., 2024), mel-spectral distance (Caillon and Esling, 2021), mel-cepstrum distortion (Masuda and Saito, 2023) and chroma features distance (Shi et al., 2023). **Pitch reconstruction** metrics, such as f_0 and multi-pitch transcription scores, evaluate the ability of the models to preserve the pitch content by comparing the outputs of a pre-trained AMT model on real and synthesized recordings (Cooper et al., 2021; Jonason et al., 2023; Hawthorne et al., 2022).

On the other hand, reference-free audio metrics evaluate audio quality without comparing individual pairs of synthesized and target audio. Instead, they are compared as a whole by considering all synthesized outputs and all reference outputs as two distributions to compare. Hayes et al. (2021); Caspe et al. (2022); Hawthorne et al. (2022); Barahona-Ríos and Collins (2024) all used **Fréchet Audio Distance (FAD)** for instance. By leveraging a VGGish model trained for audio classification (Hershey et al., 2017), they extract embeddings on all synthesized and real test samples: two multivariate Gaussians

are fitted to the synthesized embeddings and real embeddings respectively, to be then compared using the Fréchet distance. Several works have reported a correlation between FAD and perceptual similarity (Hayes et al., 2021; Hawthorne et al., 2022), implying the use of this metric as an objective proxy of listening tests.

Speed is also an important factor for the usability of an algorithm by end-users, especially in the context of music creation. Slow algorithms can hinder the creative process for music production, and can even be unusable in the context of live performance. The **real-time factor** is commonly used for quantifying the speed of an algorithm (whether causal or not): it is the ratio between the time taken to process a certain amount of data and the corresponding duration of the data. An algorithm is said to be *real-time*, or *capable of real-time*, when its real-time factor is 1 or less, and *non-real-time* otherwise. For online processing, it can be understood intuitively as the ability of the model to "keep up" with controls being continuously inputted. This metric is highly dependent on the hardware running the algorithm. For neural-based models, there is usually a clear difference of the real-time factor when the models run on a Central Procesing Unit (CPU) or on a GPU. Neural network layers have efficient implementations for training and running on GPUs (using parallel processing and dedicated hardware architectures), which makes the real-time factor on GPU almost always better than the real-time factor on CPU. However, performances on CPU are more valuable as CPUs are always available in end-user computers, while there is often a discrepancy between the GPU of an end-user and those used by researchers for model training. In recent years, GPU-like processors, which can execute neural models are becoming more accessible for the end-users (Apple M series, Google Tensor chips, gaming PC), which makes the GPU real-time factor worthy of consideration to some extent.

Lastly, model compactness is also sought after in musical applications, as one can try to embed the synthesis models into low-memory processors, or reduce their Random Access Memory (RAM) footprint in DAWs. Traditionally, sample-based synthesizers are measured based on the size of their sound bank. As for neural-based models, the number of model parameters is generally used for comparison.

Subjective Evaluation Metrics

Subjective evaluation is usually considered to provide a more insightful assessment of audio quality, but they are more time-consuming to gather. Listening tests need to collect a pool of listeners that evaluate different models according to a specific criterion. Careful selection of inference samples is crucial to isolate the specific quality being assessed. In instrument audio synthesis, all evaluated models should generate their synthesized outputs from a common set of input controls, while ensuring uniform loudness levels to mitigate any loudness-related biases. Each evaluation score is valid only for a single work, as a model is evaluated in the context of other similar methods. The full evaluation process on all methods has to be conducted when adding or removing a specific model, and evaluators' ratings may exhibit variability.

For audio synthesis tasks, the most commonly used measure gathered through listening tests is the Mean Opinion Score (MOS). Listeners are asked to rate each sample individually on a 5-point Likert scale according to a property. For example, TTS evalu-

ations usually ask to independently rate the naturalness and intelligibility properties of speech excerpts.

Another listening test format is the MUltiple Stimuli with Hidden Reference and Anchor (MUSHRA). Originally developed for evaluating audio encodings, the listeners are asked to rate the similarity of the inference samples with the real sample. Ratings are given with a finer-grained scale than the 5-point Likert scale. Hidden anchors, being the real sample itself and a purposely degraded version of the real sample, enable filtering out evaluators with sub-optimal listening conditions.

Section summary - Instrument Audio Synthesis State-of-the-Art

Audio synthesis of musical instruments has a long history of proposed systems with various methodologies, needs for knowledge and data, and ultimately, various realism qualities. Given an input sequence x of notes/controls, the designed synthesis model $\mathcal{S}$ aims to generate an audio signal $\hat{y}$ that imitates the sound that would be produced by the target instrument. In particular, synthesizing piano sounds from MIDI inputs can be done through physical modeling, signal-based modeling, concatenative synthesis, or neural audio synthesis. The first two methodologies require leveraging knowledge of the instrument in order to conceive a tailor-made model but require extensive parameter retrieval in order to sound realistic. The last two categories of techniques can mimic an instrument with minimal knowledge required but demand large storage or model sizes to capture the full complexity of the instrument. Recently, Differentiable Digital Signal Processing (DDSP) offers a hybrid approach by incorporating signal-based modeling knowledge into neural audio synthesis. It leverages the advantages of both approaches, mainly the interpretability of signal-based synthesis and the expressivity of neural-based synthesis. Although it also raises new challenges in the optimization process, most notably for frequency estimation. Audio reconstruction in time-frequency representations is usually used to objectively evaluate audio synthesis models, but subjective evaluation through listening tests better reflects the subjectivity of audio quality perception.

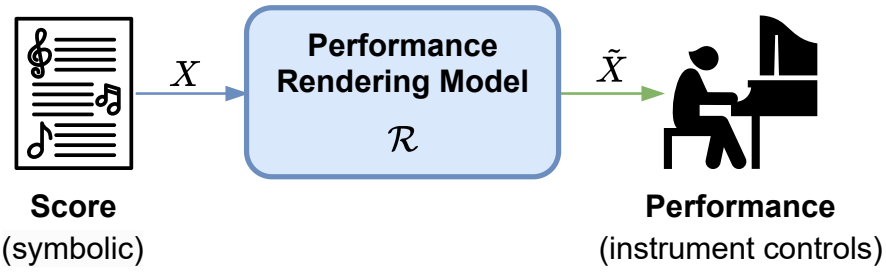


Figure 3.5: The expressive performance rendering task. For the case of piano music, the output instrument controls are usually encoded in MIDI.

3.3 Performance Rendering

As mentioned in Section 2.1.1, the music score is a compact, symbolic representation of a musical piece made by a composer. Then, during a performance, musicians play their instruments, following the score in order to make an audio rendition of the piece. Similarly, as for the one-to-many relation between text and speech, there is not one, but a plethora of valid interpretations that follow the score indications. Those usually produced by musicians are *expressive*, in an attempt to bring out effective and emotional qualities that can resonate with the listeners (Cancino-Chacón et al., 2018). On the contrary, the simplest and most straightforward rendition of a score is usually perceived as unnatural and emotionless.

As illustrated in Figure 3.5, the performance rendering task aims to imitate musicians that imbue a music score with expressive features, through the control of their instrument that is not entirely described by the written score. Formally, given a composition as a sequence of N notes with score features $X = \{x_n\}_{n \leq N}$, the performance rendering model $\mathcal{R}$ aims to produce an expressive rendition $\tilde{X} = \{\tilde{x}_n\}_{n \leq N}$ by modification or enhancement of the composition with performance features:

$$\tilde{X} = \mathcal{R}(X). \tag{3.4}$$

These expressive features can be organized into four dimensions:

- *timing* features include instantaneous tempo deviation, micro-displacement of note onsets, and change of note duration (articulation).
- the *loudness* features denote the local deviations of dynamics from the global score annotation. They encompass the individual note loudness in a chord and the evolving nuance of held notes.
- the *pitch* controls reflect micro-tonal deviation effects, such as vibrato and slides between notes.
- one of several *timbres* offered by the instrument, elected through some playing techniques (palm muting, sustain pedal,...)

It can be seen that the expressive features are entailed in the available controls offered by the playing instrument. For instance, on the contrary of violists, piano players cannot modify note nuances beyond the onset velocity, nor modulate note pitches. Thus, a piano performance can be more easily encoded through the MIDI protocol, than violin performances. According to the instrument, the rendered performance can be represented in the symbolic domain, but some works have also attempted to directly render the performance audio signal. The following will briefly cover the different propositions for tackling the piano performance rendering task. For continuously controlled monophonic instruments, one can refer to the work of Wu et al. (2022b) that conceived a hierarchical modeling approach to complete the insufficient MIDI inputs, through DDSP controls, and render audio performances.

3.3.1 Traditional Approaches to Performance Rendering

Before the rise of deep neural networks, expressive performance rendering was tackled using rule-based and less complex data-driven systems. They were notably fostered by the RenCon competition (Hashida et al., 2008) that evaluated piano performance rendering models. These *computational models* are discussed more extensively in the review of Cancino-Chacón et al. (2018).

Rule-based approaches rely on manually designed performance rules, motivated by musical hypotheses. Such systems include the KTH model (Friberg et al., 2006) that derived rules from both experts and conducted listening tests.

Modeling the complex relationship between a score notation and its actual realization has been more thoroughly explored from actual observations of score-performance pairs and learned with data-driven probabilistic models. Such systems can leverage Hidden Markov Models (Grindlay and Helmbold, 2006), Bayesian networks (Flossmann et al., 2013) and switching Kalman filters (Gu and Raphael, 2012). Maximum entropy models were also used by Kim et al. (2013).

3.3.2 Neural Approaches to Performance Rendering

The first modern work integrating deep neural networks for piano performance rendering is the Basis Mixer model of Chacón and Grachten (2016) that combined RNNs with a Gaussian Mixture Model. For loudness prediction, Malik and Ek (2017) used genre-specific RNNs to predict MIDI velocities from score piano rolls. For more expressive renditions, Wang and Yang (2019) conceived a model that converts piano rolls directly into performance spectrograms. While this model can generate more fine-grained loudness features, they are entangled with the notes and the instrument timbre in the output spectrograms, which greatly limits the a posteriori manipulation of the rendered performance. The Deep Performer model (Dong et al., 2022) also proposed a full audio performance rendering pipeline by concatenating a transformer-based score-to-performance alignment system with a TTS-inspired neural synthesizer. Each sub-model is trained separately, the alignment model in particular changes the onset, duration, and velocity properties of the notes, based on the score tempo and the pitch, onset, and duration of the input notes.

The alignment model is solely trained on violin data but is used on piano pieces during inference, making the underlying assumption that the interpretation process is instrument ignorant.

As stated previously, performance rendering is a one-to-many task, as different valid interpretations can be made from the same music score. Such variability can be observed between musicians and for different performances of the same musician. Variational techniques have proven to be relevant for learning a latent space of realistic playing styles when conditioned on score features (Maezawa et al., 2019; Jeong et al., 2019a,b; Rhyu et al., 2022; Borovik and Viro, 2023). The main difference between these works is how they maintain performance coherence across the whole music: Maezawa et al. (2019) autoregressively synthesize performance features using an internal hidden state, while VirtuosoNet (Jeong et al., 2019a,b) and ScorePerformer (Borovik and Viro, 2023) employ hierarchical approaches by encoding the score (and performances in ScorePerformer) at the note, beat and measure levels. Rhyu et al. (2022) stands out by putting an emphasis towards the musicians' personal intention instead of solely relying on the guidance provided by the score: stylistic transfer of performer intent from one piece to another is achievable in their work.

The performance features outputted by all previous approaches are defined as the difference in timing, articulation, and velocity of the played notes, compared to the exact rendition of the score (Jeong et al., 2019c). The score features may vary, but they mainly include note-wise pitch, tempo, positions in the measure, dynamic and articulation annotations, and eventually high-level score markings such as slurs, positions in a chord, and key and time signatures. Obtaining such features requires the collection of MIDI performances with their associated digital scores and aligning them at note-level (Nakamura et al., 2017; Foscarin et al., 2022; Peter et al., 2023). These requirements limit the amount of available training data and call for scores in MusicXML format since MIDI does not include high-level score markings (see Section 2.1).

Recent works have started addressing the task without these prerequisites. Tang et al. (2023) circumvents the reliance on gathering matching scores and performances, by using the performance-to-score transcription model of Liu et al. (2022) to get estimated scores from the MIDI performances (also transcribed) of the ATEPP dataset (Zhang et al., 2022). They then use a Transformer encoder to predict the performance features from those estimated scores. Zhang and Dixon (2023) completely discards score data by employing a Vector Quantized (VQ)-VAE with mutual information minimization for disentangling the score content from the performing style in ATEPP performances. Still, they require multiple performances of the same pieces in order to extract the different performing styles from the common score content. Also, since they did not include raw scores as a style in itself, they have not shown how the model behaves for making an expressive rendition out of a style-deprived score.

3.3.3 Evaluating Expressivity

Evaluating performance rendering models is still an open question as it requires quantifying musical and emotional qualities, which are highly subjective. Given a test score, a

naive approach to objective evaluation would be to measure the difference between rendered performance features from the real ones, with mean absolute error, mean squared error, or Pearson correlation. However, this requires selecting a single performance per score, which does not take the diversity of valid performances into account. Thus, this metric is only relevant as a reconstruction metric for models encoding and regenerating performances (with optional score input), such as the auto-encoder models from Jeong et al. (2019a,b) and Rhyu et al. (2022).

To compare against a set of real performances of the same musical track, Jeong et al. (2019b) proposed to measure the inter-set correlation between a rendered performance and human performances, but only on pieces that have a strong intra-set correlation between their test performances. Zhang and Dixon (2023) and Tang et al. (2023) both developed proxy metrics for evaluating the reconstruction of performer styles, through a performer identification model and performer-wise velocity distributions respectively.

In the end, the most widespread evaluation of performance rendering models is through listening tests. To prevent any audio quality bias, performances rendered by different methods have to be synthesized with the same audio synthesis model, which preferably needs to be of sufficient quality. However, the rating criterion that is asked to be used by the listeners has to be carefully chosen, as it can be subject to personal interpretation (Cancino-Chacón et al., 2018).

- Listeners can be asked to rate the "naturalness" and "expressiveness" of performances on a Likert-scale in order to get MOS, as done by Maezawa et al. (2019) and Jeong et al. (2019a).
- A MUSHRA test can also be done by asking the listeners to quantify the "expressive differences" of different methods compared to a human performance (Tang et al., 2023).
- Pairwise evaluations have been done in Jeong et al. (2019b); Rhyu et al. (2022) where two renditions of the same piece are presented and listeners have to elect the one with "better musical quality". Evaluated methods can be present in each excerpt of a pair (Jeong et al., 2019b), or always presented against the raw rendition of the score (Rhyu et al., 2022).

Section summary - State-of-the-art in Expressive Performance Rendering

The performance rendering task aims to produce an expressive rendition of a given composition as if a musician injected an interpretation to communicate expressive qualities. It translates through instrument-specific controls that are not entirely described in the music score. For piano music, performers can inflect small variations of note onset timings, durations, and velocities. These performance features are usually modeled by data-driven approaches, conditioned on features extracted from the music score. Stochastic models, such as VAEs, have been popular because they can reflect the diversity of valid performances for the same input score. Traditionally, the task has always been handled in a supervised fashion by providing matching pairs of scores and performances, but very recent works have sparked an interest in removing this prerequisite. Nonetheless, evaluating performance rendering models is non-trivial as expressive qualities are highly subjective: listening tests are thus preferred for this end.

3.4 State-of-the-Art, in short

Chapter summary - State-of-the-art

This chapter reported on the available data and methods developed by the scientific community for handling the tasks of musical audio synthesis and expressive performance rendering. For piano music, large and high-quality datasets of audio and symbolic performances are accessible, thanks to the Disklavier/piano player technology. They enable the training of deep neural models $\mathcal{S}$ for the synthesis of piano audio $\hat{y} \in \mathbb{R}^{N_{\text{samples}}}$ from MIDI input performances x, implicitly modeling the instrument from these input-output observation pairs. However, piano modeling was tackled long before the democratization of deep learning, by leveraging physical modeling to derive and predict characteristic behaviors of the instrument, at the expense of laborious explicit modeling and/or parameter estimation difficulties. DDSP democratized the integration of signal processing tools (that rely on such audio modeling knowledge) into the deep neural optimization framework, leading to the revitalization of signal processing and synthesis methods in the era of deep learning, while alleviating the modeling difficulty for neural networks. As for performance rendering, a model $\mathcal{R}$ should transform a music score X into an expressive interpretation $\tilde{X}$ to communicate emotional features. The interpretation manifests itself through specific instrument controls chosen by the performer. For piano music, data-driven approaches have been popular for learning and reproducing performance features from provided pairs of scores and performances. Still, the variety of plausible performances for the same score raises challenges in both the modeling and evaluation stages.

Chapter 4

DDSP-Piano: a Neural Piano Synthesizer informed by Instrument Knowledge

This chapter presents the main contribution of this thesis: a neural piano synthesizer based on DDSP, with an architecture motivated by knowledge inherited from physical and signal-based modeling of the instrument. The model is MIDI-controllable and extends the original DDSP approach for handling polyphonic inputs and further incorporating instrument modeling knowledge that was notably introduced in the previous chapters.

First, the model and its related experiments are presented in Section 4.1, as in the published papers (Renault et al., 2022, 2023a). Then, in Section 4.2, several improvements of DDSP-Piano are proposed in order to address the shortcomings revealed through the analysis of the first version. Based on the presented work, Section 4.3 will extract some guidelines for designing hybrid neural models and discuss future improvements of the model. The chapter will be finally summarized in Section 4.4.

Completing this chapter, examples of audio synthesis[1] and the source code[2] for model training and inference can be found online.

4.1 First Iteration of DDSP-Piano

4.1.1 Model architecture

The proposed synthesis model is a sines-plus-noise synthesizer (Section 2.2.4) with polyphonic controls and outputs. It separately generates the inharmonic and noisy components $y^{\text{additive}} \in \mathbb{R}^{N_{\text{samples}}}$ and $y^{\text{noise}} \in \mathbb{R}^{N_{\text{samples}}}$ of up to P simultaneous notes. The synthesized audio $\hat{y} \in \mathbb{R}^{N_{\text{samples}}}$ is produced by summing all monophonic signals and by applying the estimated LTI response $\text{IR}_i \in \mathbb{R}^{N_{\text{reverb}}}$ of the recording environment i:

$$\hat{y}(t) = (\text{IR}_i * \sum_{p=1}^{P} (y_p^{\text{additive}} + y_p^{\text{noise}}))(t). \tag{4.1}$$

[1] http://renault.gitlab-pages.ircam.fr/thesis-support/chap_4-1
[2] https://github.com/lrenault/ddsp-piano

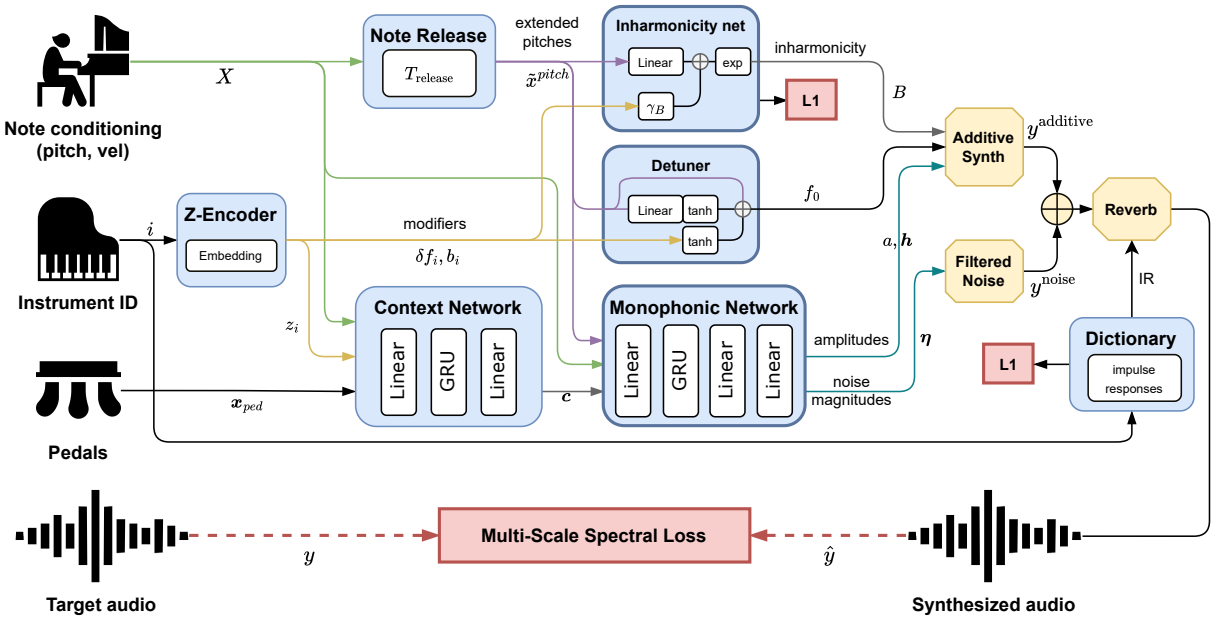


Figure 4.1: Full architecture of the proposed piano sound synthesizer. The blue boxes represent the trained modules for the control of the synthesis. Those with a thickened border are applied along each monophonic voice. The synthesis modules from DDSP are represented by yellow boxes (*Additive, Filtered Noise*, and *Reverberation*). Finally, the *Multi-Scale Spectral Loss* compares the *target signal* (bottom left) and the output *synthesized sound* (bottom right).

The following sections detail the sub-modules composing the full model architecture, which is illustrated in Figure 4.1.

Input conditioning

DDSP-Piano is conditioned on all the controls pianists have over their instrument: the sequence of notes being played (including their velocities), the action of the pedals, and the recording environment.

The monophonic conditioning of a DDSP model (Section 3.2.3) is comprised of f_0 and loudness control signals, which provide the instantaneous fundamental frequency and intensity of a note sequence at a constant frame rate F_{frame}. In the proposed model, for compatibility with MIDI inputs (see Section 2.1.2), the f_0 control is substituted by an active pitch control signal $x^{\text{pitch}}(\tau)$ that indicates the MIDI pitch of the note at frame τ, taking the sustain pedal effect into account for its duration. Note that $x^{\text{pitch}}(\tau) = 0$ means that no note is currently being played at frame τ. Likewise, the loudness control is replaced by an onset velocity control $x^{\text{vel}}(\tau)$ that specifies the note velocity (scaled to the range $[0, 1]$) only at onset time. This encoding allows to disentangling of sustained notes from repeated notes within an active sustain pedal, as explained in Section 2.1.3. For an input MIDI control sequence spanning over T frames, the polyphonic conditioning $X \in \mathbb{R}^{T \times P \times 2}$ is obtained by allocating all input notes into P sequences of non-overlapping

notes, or *voices*, and by stacking these monophonic conditionings, as in Kawamura et al. (2022):

$$X(\tau) = \{x_p^{\text{pitch}}(\tau), x_p^{\text{vel}}(\tau)\}_{p\in[\![1,P]\!]}. \tag{4.2}$$

The pedal input controls $\boldsymbol{x}^{\text{ped}}(\tau)$, including most notably the sustain pedal control, are extracted from the MIDI data at the same frame rate as the conditioning input $X(\tau)$.

Finally, the piano model, the room reverberation, and the microphone choice and placement are all entangled independently of the piano performance: each recording environment is provided as a one-hot encoding $i \in [\![1, I]\!]$, for I different recording environments in the dataset.

Global model

Because the size and the tuning of physical pianos vary considerably, related parameters like the timbre, the inharmonicity and the detuning profiles over the pitch range, described in Section 2.3.4, have to be adapted accordingly. Thus, a **Z-Encoder** is used to store an embedding vector $\boldsymbol{z}_i \in \mathbb{R}^Z$, an inharmonicity modifier $b_i \in \mathbb{R}$, and an instrument-specific detuning $\delta f_i \in \mathbb{R}$ for each recording environment i.

Also, as mentioned in Sections 2.3.2 and 2.3.4, the pedals and the mutual interaction between simultaneous notes can change the timbre of an individual note. This effect is modeled by a C-dimensional context signal control $\boldsymbol{c} \in \mathbb{R}^{T\times C}$ computed by the RNN-based **context network** $\mathcal{C}$, from the piano embedding $\boldsymbol{z}_i$, the pedal controls $\boldsymbol{x}^{\text{ped}}(t)$ and the conditioning $X(t)$:

$$\boldsymbol{c}(\tau) = \mathcal{C}(\{X(\tau'), \boldsymbol{x}^{\text{ped}}(\tau'), \boldsymbol{z}_i\}_{\tau'\leq\tau}). \tag{4.3}$$

This context signal is duplicated across all monophonic voices, which gives the subsequent monophonic layers (to be presented in Section 4.1.1) access to global and polyphonic information and thus adjusts the computation of monophonic note properties accordingly.

Monophonic string model

Following Section 2.3.2, the attenuation of the string vibration by the damper is not instantaneous, and higher notes do not even have dampers. Hence, in practice, the piano strings still vibrate for a certain amount of time after the note offset. Inspired by the *release* parameter of digital synthesizers, a **Note Release** module is implemented to generate an extended pitch signal $\tilde{x}^{\text{pitch}} \in \mathbb{R}^{T\times 1}$ by prolonging the active pitch component of the conditioning signal x^{pitch} by a learned duration $T_{\text{release}} \in \mathbb{R}$. Note that the extended pitch conditioning signal $\tilde{x}^{\text{pitch}}$ does not replace the original pitch conditioning x^{pitch}, as we would lose the note offset information.

Furthermore, the explicit *inharmonicity model* adapts Equation 2.13 the get the inharmonicity coefficient $B_p \in \mathbb{R}^{T\times 1}$ of the p-th voice from the extended pitch $\tilde{x}_p^{\text{pitch}}$ and the instrument-specific modifier b_i:

$$\begin{aligned} B_p(\tau) = &\exp\left(\alpha_T \tilde{x}_p^{\text{pitch}}(\tau) + \beta_T\right) \\ &+ \exp\left(\alpha_B \tilde{x}_p^{\text{pitch}}(\tau) + \beta_B + \gamma_B b_i\right), \end{aligned} \tag{4.4}$$

with $\{\alpha_T, \beta_T\}$ (resp. $\{\alpha_B, \beta_B\}$) the parameters of the linear asymptote in the treble (respectively bass) range. According to Young (1952), the treble asymptotes are very similar across all pianos, so b_i only influences the bass asymptote, weighted by the parameter $\gamma_B \in \mathbb{R}$.

The partial beating produced by string duets and triplets is approximated by means of constructing a monophonic note as the sum of n_{string} sub-strings, each detuned by a detuning factor δf. The **detuner** sub-model gathers the per-string deviations predicted by a time-distributed linear layer g_δ from the pitch, and a global instrument-specific detuning δf_i:

$$\boldsymbol{\delta f}_p(\tau) = \tanh(g_\delta(\tilde{x}_p^{\text{pitch}}(\tau)) + \tanh(\delta f_i). \tag{4.5}$$

Each command contributing to the detuning is limited to a semitone range $[-1, 1]$, as in Wu et al. (2022b), using the tanh activation function.

Finally, the spectral envelopes of notes and their evolution are predicted by the **monophonic network** $\mathcal{M}$. It is implemented as a causal RNN that computes the remaining synthesizer controls from the extended pitch $\tilde{x}^{\text{pitch}}$, the conditioning vector X, and the context vector $\boldsymbol{c}$. This recurrent network is applied along each voice $p \in [\![1, P]\!]$, in order to learn a monophonic string model and to predict the monophonic notes amplitude $a_p \in \mathbb{R}^{T\times 1}$, the energy distribution $\boldsymbol{h} \in \mathbb{R}^{T\times K}$ for K partials and noise filter magnitudes $\boldsymbol{\eta}_p \in \mathbb{R}^{T\times Q}$, with Q the number of frequency filter bands:

$$a_p(\tau), \boldsymbol{h}_p(\tau), \boldsymbol{\eta}_p(\tau) = \mathcal{M}(\{X_p(\tau'), \tilde{x}_p^{\text{pitch}}(\tau'), \boldsymbol{c}(\tau')\}_{\tau'\leq\tau}). \tag{4.6}$$

Differentiable Synthesizers

The outputs of the neural network are used to control the differentiable synthesizers, which generate and process audio signals following the spectral modeling paradigm. As in the original DDSP auto-encoder (Section 3.2.3), controls are upsampled from the controls frame rate F_{frame} to the audio sampling rate F_{audio}. Amplitude, energy distribution, and noise filter magnitude controls are also scaled with the exponential sigmoid function sigmoid† (defined in Equation 3.2) in order to be non-negative.

Along a monophonic voice $p \in [\![1, P]\!]$, the *additive synthesizer* generates the inharmonic audio component $y_p^{\text{additive}} \in \mathbb{R}^{N_{\text{samples}}}$ of the piano notes. It sums multiple sinuses at frequencies computed from the extended pitch $\tilde{x}_p^{pitch}$, inharmonicity B_p and detuning δf_p controls, and with amplitudes provided by the global amplitude a_p and harmonic distribution $\boldsymbol{h}_p$:

$$y_p^{additive}(t) = \frac{a_p(t)}{n_{\text{strings}}} \sum_{n=1}^{n_{\text{strings}}} \sum_{k=1}^{K} \boldsymbol{h}_{p,k}(t) \sin(\Phi_{p,n,k}(t)), \tag{4.7}$$

with $\Phi_{p,n,k}(t)$ the instantaneous phase of the k-th partial at timestep t:

$$\Phi_{p,n,k}(t) = 2\pi \sum_{t'=0}^{t} f_{p,n,k}(t'). \tag{4.8}$$

The inharmonic frequencies $\{f_{p,n,k}\}_{k\in[\![1,K]\!]}$ are deduced from the fundamental frequency $f_{p,n,0}$ and the inharmonicity coefficient B_p with Equation 2.12. $f_{p,n,0}$ are obtained by converting the detuned pitch $\tilde{x}_p^{pitch}+\delta f_n$ into frequencies with the MIDI note-to-frequency formula defined in Equation 2.2.

The *subtractive synthesizer* generates the residual noises that happen during a performance, mainly the hammer and key noise upon note onsets, the pedal noises, and even the recording background noise. As in Engel et al. (2020a), a white noise STFT spectrum $\boldsymbol{u} \in \mathbb{R}^{T\times Q}$ is filtered in the frequency domain with the LTV noise filter magnitudes $\boldsymbol{\eta} \in \mathbb{R}^{T\times Q}$ computed by the model, before being inverted in the audio domain:

$$y_p^{\text{noise}}(t) = \text{DFT}^{-1}\Big[\boldsymbol{\eta}_p(\lfloor t\frac{F_{\text{frame}}}{F_{\text{audio}}}\rfloor)\boldsymbol{u}\Big](t). \tag{4.9}$$

The room response in the piano recordings is modeled by a differentiable convolutional reverb. A FIR IR_i is learned for each recording environment $i \in [\![1, I]\!]$ and it is applied to the sum of audio signals output by the bank of additive and subtractive synthesizers (Equation 4.1).

4.1.2 Model Training

DDSP-Piano is trained and evaluated with MIDI-audio performances from the MAESTRO dataset, which spans over $I = 10$ editions of the Piano e-competition (see Section 3.1). The ground-truth audio performances are downsampled to $F_{\text{audio}} = 16$kHz and downmixed to mono. The *una corda*, *sostenuto* and *sustain* pedal controls are available in the MIDI recordings through the 64, 66, and 67 `CC` messages, which corresponds to the three pedals of most grand pianos (Section 2.3.2). Conditioning and pedal controls are extracted at a frame rate of $F_{\text{frame}} = 250$Hz, which is also the control rate in the original DDSP paper (Engel et al., 2020a).

Tracks are split into 3-second long segments, with a 50% overlap between two consecutive segments. The input and output lengths are $T = 750$ and $N_{\text{samples}} = 48000$. Segments in which the maximum number of simultaneous notes is greater than the model polyphonic capacity P are removed from the training set.

In the tradition of neural-based synthesis, the model is trained to minimize the MSS loss, as defined in Equation 3.3, between the target audio y and the synthesized audio $\hat{y}$.

In preliminary experiments, it has been observed that the reverb module tried to model the sustain and release behavior of the notes, which did result in abnormal reverberations and unrealistic raw piano signals that were generated as input for the reverb module. A L1 regularization loss $\|.\|_1$ is applied to the learned impulse responses to reduce the reverb complexity thus discouraging the module from learning characteristics of the piano tones (such as note sustains and releases) that can be modeled by the other unregularized modules.

Furthermore, the correct placement of partials in frequency is decisive for training stability, especially during early training. As partial frequencies in our system are deduced from explicit sub-modules, a two-phase training procedure is proposed for separately optimizing the pure DNN components and the explicit sub-models:

1. During the first training phase, weights responsible for computing the partial frequencies are frozen to their pre-defined initial values, which should be close to the optimal ones. The concerned weights are those from the *detuner*, the *inharmonicity model* and the model-specific detuning and inharmonicity modifiers of the *Z-encoder*. The other modules can thus learn to reproduce the spectral envelopes of the notes, the residual noises, and the reverberation without displacing the note partials. The model is optimized using the Adam optimizer (see Section 2.4.1) with a learning rate of 10^{-3} and a batch size of 6, with regard to the first phase loss function $\mathcal{L}_1$:

$$\mathcal{L}_1 = \mathcal{L}_{\text{MSS}}(y, \hat{y}) + \lambda_{\text{IR}} \sum_{i=1}^{I} \|\text{IR}_i\|_1, \tag{4.10}$$

   with λ_{IR} the balancing weight for the reverb regularization loss with regard to the spectral loss, here set to 0.01.

2. During the second training phase, the trainability of the model weights is reversed compared to the first training phase. In such a manner, the system is expected to match the learned partial frequencies and beating for each piano specifically. For stabilizing the training, an L1 regularization loss is applied to the deviation of *inharmonicity model* parameters from their initial values. The total loss associated with this second training phase can be expressed as:

$$\mathcal{L}_2 = \mathcal{L}_{\text{MSS}}(y, \hat{y}) + \lambda_{\text{B}} \sum_{\theta \in \{\alpha_B, \beta_B, \alpha_T, \beta_T\}} |\theta - \theta^0|, \tag{4.11}$$

   with $\lambda_B = 0.1$ the weight on the *inharmonicity model* regularization loss with regard to the spectral loss. The parameters of the *detuner*, the *inharmonicity model*, and the *Z-Encoder* model-specific modifiers are fine-tuned by Adam with a learning rate of 10^{-5} and a batching size of 3.

The whole system is optimized and fine-tuned by successively alternating between these two training phases. In our experiments, the system is trained with the first phase formula for 2 full epochs on the 160h of training data, until note partials are correctly generated by the additive synthesizer. It is then fine-tuned for 1 full epoch on the training data with the second training phase. Finally, the model is further fine-tuned using the first training phase again for 3 epochs, until the minimal validation loss value is reached. The full model training takes about 340k steps, which corresponds to around 8 days of training on a single Nvidia GeForce GTX 1080 Ti GPU.

4.1.3 Evaluation

Both objective and subjective evaluations have been conducted on DDSP-Piano. It is compared with other sound synthesis methods and some ablated variants.

F_{frame}	F_{audio}	N_{samples}	N_{reverb}	T	I	P	Z	C	n_{string}	K	Q
250	16000	48000	24000	750	10	16	16	32	2	96	64

Table 4.1: Hyper-parameters of the *Default* model configuration.

Baselines

The proposed DDSP-Piano model is evaluated against the piano sound synthesis methods presented in Section 3.2. All samples synthesized with the following systems are also downsampled to 16kHz and converted to mono.

The commercial software `Pianoteq 7`[3] with the default preset `NY Steinway D Classical` is used as the physical-modeling-based baseline. Results from the physical modeling of the instrument are synthesized in real-time using modal synthesis (Bank and Chabassier, 2019).

For the wavetable synthesizer benchmark, performances are obtained by stitching isolated note recordings from the `YDP Grand Piano` [4] soundfont, using the open-source software `Fluidsynth` [5].

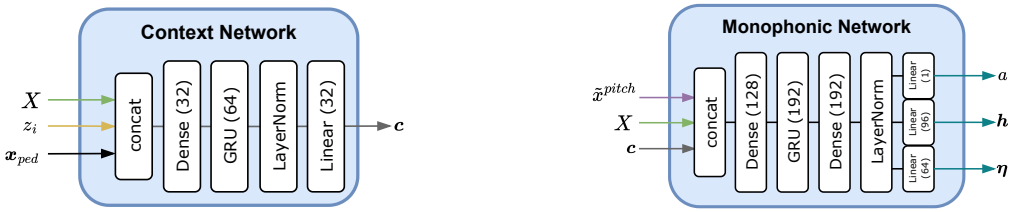
Finally, **Piano-TTS v1**, the TTS-inspired model from Cooper et al. (2021) presented in Section 3.2.2, is elected as the pure neural audio synthesis benchmark. Also trained on MAESTRO, this model is a modified Tacotron-2 acoustic model followed by a simplified NSF vocoder model. MIDI-filter-bank-based spectra are used as the intermediate representation between the two sub-models, which has the advantage of being aligned with the input piano rolls in the frequency/pitch axis.

Default and Ablated models

The **Default** configuration of DDSP-Piano is given in Table 4.1, with the architectures of the context $\mathcal{C}$ and monophonic models $\mathcal{M}$ illustrated in Figures 4.2a and 4.2b respectively.

As for the initial values of the explicit layers, the release time of the **Note Release** module is set to $T_{\text{release}} = 1\,\text{sec}$, which is longer than the observed attenuation time of piano notes after key release (Lehtonen et al., 2009). The **inharmonic model** is initialized with the parameters estimated in Rigaud et al. (2011): $\alpha_B^0 = -0.0847$, $\beta_B^0 = -5.82$, $\alpha_T^0 = 0.0926$ and $\beta_T^0 = -13.64$. Initial detunings are set to zero in the linear model g_δ of the **detuner**, like the model-specific inharmonicity and detuning modifiers of the **Z-encoder**, in order to first learn a generic piano model during early training. Finally, the different reverb impulse responses are 1.5 seconds long at 16kHz (24k parameters for each recording environment), with the same random initialization as in Hayes et al. (2021) and the inference decay function from Wu et al. (2022b). The total number of training parameters in this **Default** configuration is given in Table 4.2, against **Piano-TTS v1**, the neural-based benchmark.

[3] https://www.modartt.com/pianoteq
[4] https://freepats.zenvoid.org/Piano/acoustic-grand-piano.html
[5] https://www.fluidsynth.org/



(a) Context Network $\mathcal{C}$.

(b) Monophonic network $\mathcal{M}$.

Figure 4.2: Architectures of the Context and Monophonic networks. The Context network computes a context signal $c \in \mathbb{R}^{T \times C}$ from the input conditioning $X(t) \in \mathbb{R}^{T \times P \times 2}$, the pedals input signal $x^{\text{ped}} \in \mathbb{R}^{T \times 3}$ and the instrument embedding $z_i \in \mathbb{R}^{Z}$. The Monophonic Network takes the context signal $c \in \mathbb{R}^{T \times C}$, a voice conditioning input $X_p \in \mathbb{R}^{T \times 2}$ and the extended pitch control $\tilde{x}_p^{\text{pitch}} \in \mathbb{R}^{T \times 1}$ to predict the monophonic DDSP synthesizer controls, being the global amplitude $a_p \in \mathbb{R}^{T \times 1}$, the energy distribution of the partials $h_p \in \mathbb{R}^{T \times K}$ and the residual noise filter coefficients $\eta_p \in \mathbb{R}^{T \times N_{\text{noise}}}$. Numerical shapes reflect the implementation of the *Default* version of DDSP-Piano.

Model	Parameters
Piano-TTS v1	31.4M
- Tacotron-2	30.6M
- NSF	736.3k
DDSP-Piano	512.5k
- Sub-models	281.5k
- Reverb	240k

Table 4.2: Approximate number of parameters of the evaluated neural-based models and their sub-models.

The relevance of the system sub-modules is evaluated by training ablated versions of DDSP-Piano. All following variants are trained with the same procedure and hyper-parameters (losses balance, number of training steps, learning rates, batch sizes) exposed in Section 4.1.2:

- The **Deep Inharmonicity** variant replaces the explicit inharmonicity model from Rigaud et al. (2011) with a DNN. Ideally, this DNN should reproduce equation 2.13: we use a Multi-Layer Perceptron (MLP) with sinusoidal activation functions as in Hayes et al. (2021). This model takes the same inputs as the explicit inharmonicity model and it is composed of 3 dense layers with sinusoidal activation and a hidden size of 8, followed by a linear layer with ReLU activation. The final output is scaled in order to keep the estimated inharmonicity factor within a realistic range $B \in [0, 0.02]$.

- The **Reduced-context** variant imitates sampling-based synthesis by removing the conditioning input X from the context control computation. Since the synthesizer

controls are computed on all monophonic channels independently, a monophonic note control would not have information on which other notes are also played, thus preventing mutual interaction between notes.

- The **No Fine-tuning** variant is the Default configuration where the inharmonicity model and the detuner sub-modules are reverted back to their initial values before training. This variant is similar to a model trained solely with the first training phase of Section 4.1.2.

- As stated in the original DDSP paper Engel et al. (2020a), the sound structure priors inherent to the DDSP synthesizers enable full model training with a reduced amount of data. In the same manner, we test DDSP-Piano on the simpler but also less resourceful task of **single** piano modeling. The **2009-only** variant is trained by only keeping performances from the year 2009 in the MAESTRO dataset, which amounts to about 20 hours of training data.

Objective Evaluation

Among the **audio reconstruction** metrics available that were presented in Section 3.2.4, the MSS distance between the real and synthesized performances is selected. Figure 4.3 shows the reconstruction quality of the model variants on each recording environment in the MAESTRO test set. One can notice significant spectral loss differences among the different recording environments, implying that certain years were easier to model than others.

- In all recording environments, the **Deep Inharmonicity** variant has higher loss values than variants using the explicit *inharmonicity model*. This explicit sub-module proves to be valuable for the overall reconstruction quality of the model, as it allows it to control the additive synthesizer with better estimated spectral envelopes.

- **2009-only** seems to have slightly better reconstruction quality on the 2009 subset compared to the **Default** configuration, although not significantly. If the difference is confirmed perceptually, that would suggest that the piano model embedding in the multi-instrument setting is not sufficient for achieving the same quality as for single piano modeling. Nonetheless, if one wants to profile a single piano, training on the full MAESTRO dataset is not necessary and smaller aligned datasets could be used instead, such as MAPS (Emiya et al., 2010).

- Reducing the global context appears to have different consequences, as the **Reduced Context** variant can have similar, better, or worse reconstruction quality than the **Default** configuration depending on the piano model. While mutual notes interaction should be present in all performances, independently of the recording year, the perceived effect may not be exhibited in the interpreted piano pieces. If modeling such interaction is unnecessary, the **Reduced Context** variant, with reduced complexity, can converge faster than the **Default** configuration and achieve

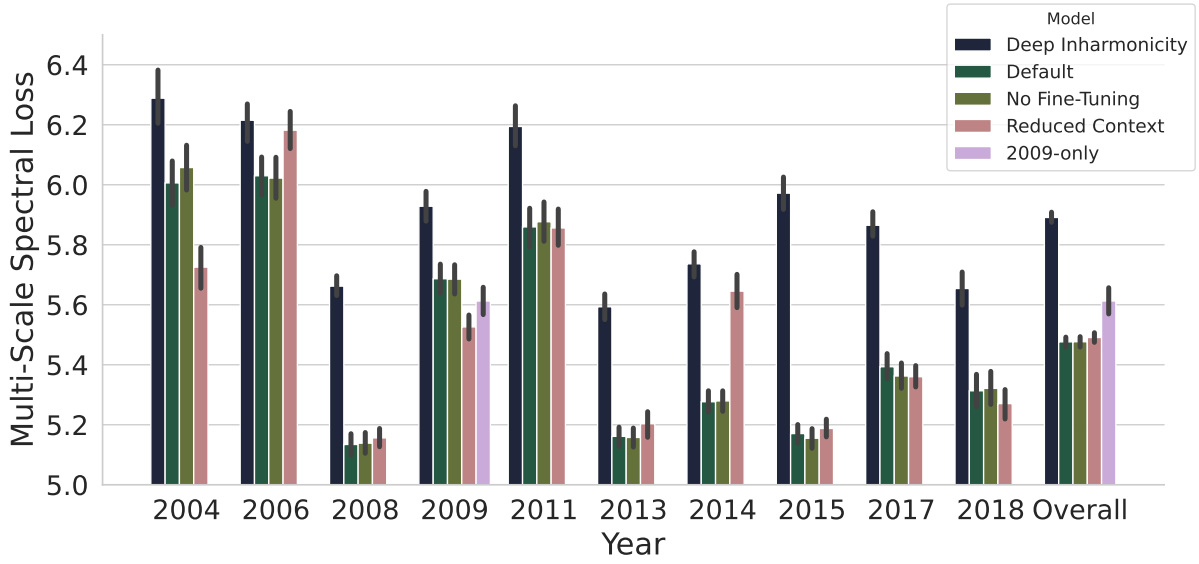


Figure 4.3: Systems evaluation on the MAESTRO test set, broken down by recording environments. Measured by mean and standard deviation values of the MSS difference (Equation 3.3) with the original recordings. Lower loss values indicate better reconstruction quality.

better reconstruction quality for the same number of training epochs: this would be the case for pieces of the years 2004 and 2009. On the contrary, if such effects are significantly present in the training pieces, the ablated variant cannot reproduce it and thus achieves worse reconstruction quality, which concerns the years 2006 and 2014. Mutual interaction between notes (sympathetic resonances) is more exploited in contemporary music for instance: using such examples may help the system to systematically learn this specificity.

- DDSP-Piano performs similarly with and without applying the fine-tuned inharmonicity and detuning parameters. This indicates that the second training phase proposed in Section 4.1.2 could not successfully fine-tune the inharmonicity and detuning parameters to match the target pianos. Otherwise, the note partials would have been exactly matched in frequency and the model could better reproduce the associated amplitude through the spectral loss, during the third training phase.

From the overall results, it seems that DDSP-Piano can be trained for single piano modeling and only with the first training phase from Section 4.1.2, without significant loss of reconstruction quality. Reducing the context may accelerate the training if mutual note interactions are ignored in the modeling. Nevertheless, the explicit inharmonicity model is crucial for reaching good reconstruction quality.

The **real-time factor** of the **Default** model was measured on the test data to assess if the provided Tensorflow implementation [6] is real-time or not. For all 3-second long

[6] https://github.com/lrenault/ddsp-piano

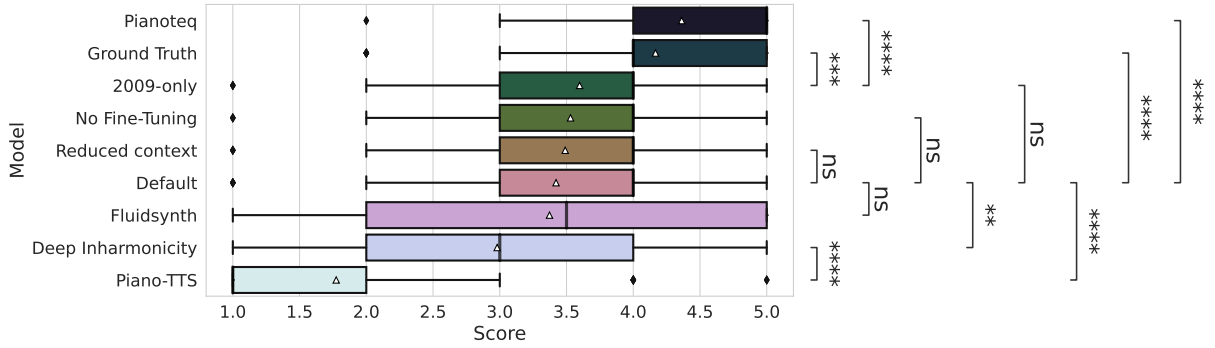


Figure 4.4: Box plots of MOS for each system. The thickened bars indicate the median values while the white triangles indicate the mean values. Two-sided Mann-Whitney U tests with Holm-Bonferroni correction were conducted on relevant systems pairs at $\alpha = 0.05$. p-value annotation legend: `ns` for $p > 0.05$; * for $p \leq 0.05$; ** for $p \leq 0.01$; *** for $p \leq 10^{-3}$ and **** for $p \leq 10^{-4}$.

segments of the test data, the average synthesis time is measured on the same hardware as for GPU training (Nvidia GeForce GTX 1080 Ti), and on a 2.6GHz Intel Xeon E5-2623 v4 CPU processor. We report real-time factors of 0.6 ± 0.1 on the GPU and 1.9 ± 0.1 on the CPU. This invalidates the usage of the current implementation for real-time applications without relying on a GPU. However, model architecture design choices were made in order to reduce the structural latency of the model: the GRU layers do not rely on future samples as they are only causal, their hidden sizes are compatible for CPU computation in real-time (Wright et al., 2019) and the model does not rely on non-causal convolution operations that raise real-time applications challenges (Caillon and Esling, 2022). Therefore, by exporting the learned model for inference into efficient frameworks dedicated to real-time neural audio processing (Wright et al., 2019; Chowdhury, 2021; Stefani et al., 2022), the real-time factor of DDSP-Piano on CPU can be improved.

Subjective Evaluation

A listening test was conducted for gathering MOS on all systems under evaluation. Eleven performances were hand-picked from the test data, covering all recording environments and with a diversity of composers, registers, and note densities. The first 9 seconds of the performances were synthesized with all systems[7], which, with the real recordings, gives 99 audio samples to evaluate. Listeners were asked to rate their overall quality on a 5-point Likert scale, from "very annoying" to "real recording". In each trial, 2 samples from each of the 8 systems and 2 real recordings were randomly presented to the listener for rating. We gathered 52 participants that are musicians or audio professionals: 14 among them have notions of piano playing and 29 have been playing the instrument for several years. Box-plot and mean values of the MOS ratings are reported in Figure 4.4, with statistical tests following the evaluation procedure of Cooper et al. (2021).

[7]We would like to thank Erica Cooper, among the authors from Cooper et al. (2021), for kindly sharing the test samples of their model for the listening test.

Comparing the ratings of the model against its ablations:

- The quality difference between the **Deep Inharmonicity** variant and the models including the explicit inharmonicity model is confirmed perceptually. Only the **Default**-against-**Deep Inharmonicity** hypothesis is not statistically significant, but the median and quartile values still suggest a slight advantage in favor of the Default configuration.

- Ratings also confirm that the second training phase does not improve the perceived quality, suggesting that the natural beating between simultaneous notes in harmony may be sufficient for achieving realistic-sounding partial beatings during polyphonic performances.

- Reducing the context also does not significantly hinder the perceived quality of the DDSP-Piano model. It can be deduced that other components of the approach can be improved before the lack of note interaction limits the perceived quality.

- Single piano modeling is still perceived as good sounding as variants trained on several pianos simultaneously, which raises the question of the minimum amount of training data required for achieving such quality. Note that previous neural-based synthesis works did not report the model quality when trained on a single environment of MAESTRO.

As for comparisons with the other piano synthesis methods:

- All variants of DDSP-Piano have a significant difference over the neural-based Piano-TTS benchmark. Although this baseline is more versatile since it was developed for speech synthesis at first, our approach is better suited for piano sound synthesis, achieving better sound quality with significantly fewer training parameters, as shown in Table 4.2.

- Only the physical-modeling-based method achieves sound quality comparable to the real recordings (even slightly better, although not significantly, as also found by Cooper et al. (2021)). Various unwanted noises and the recording quality of the real samples may have been perceived as slightly annoying compared to the clean sounds synthesized by the `Pianoteq` software. In this training setting, the data represents an upper bound limit to the quality of the neural-based synthesizer: the model can thus benefit from cleaner audio recordings, and/or from pre-processing the data with a noise-filtering strategy as in Zhang et al. (2022).

- Nonetheless, there is still a significant gap in the perceived quality between the synthesis offered by the DDSP-Piano model and the real recordings.

- As it stands, all variants of our approach are not significantly different from the sampling-based synthesizer in terms of overall quality ratings, although with less variability. Among all evaluated systems, the ratings given to the synthesis from `Fluidsynth` are the most scattered: this may suggest that some listeners are more sensitive than others to an unrealistic feature in this synthesis algorithm.

4.1.4 Qualitative Results: Comparison with Known Behaviors

Since the synthesis relies on spectral modeling, it is possible to examine the behavior of the model and interpret what it has learned more easily than with pure deep learning models. To this end, the following section will relate different DDSP-synthesizer inputs and outputs with the results expected according to signal-based and acoustic understanding.

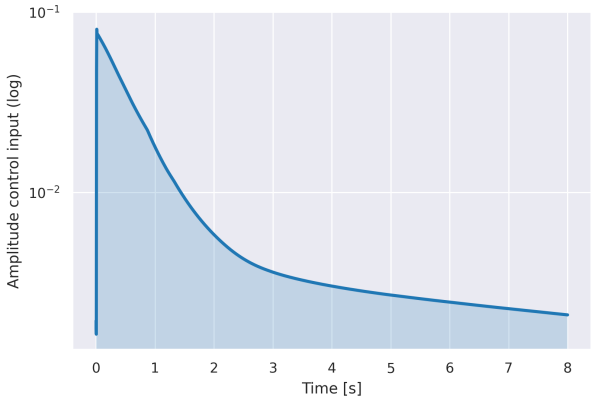


Figure 4.5: Amplitude control input (in log scale) of the additive synthesizer predicted by the Default model for a sustained A3 note with a MIDI velocity value of 100.

For a single input note, the note amplitude envelope $a_p(\tau)$ predicted by the Default configuration of DDSP-Piano is shown in Figure 4.5. In the logarithmic scale, one can see that the amplitude decays at two distinct and successive rates: the piano note decays faster right after the onset before settling down to an aftersound with a slower decay. This corresponds to the "double decay" phenomenon mentioned in Section 2.3.2. No prior knowledge of this effect was incorporated in the architecture of DDSP-Piano but thanks to the recurrent layers, it has successfully captured it directly from the audio data. Thus, explicitly modeling of the double decay is not necessary for improving the synthesis quality, but it can help reducing the number of training parameters nonetheless, as it is now done by Simionato et al. (2024).

The filtered noise output by the subtractive synthesizer from a real test performance input is shown in Figure 4.6. The audio is represented with the MIDI-filter-bank-based spectra proposed by Cooper et al. (2021), which has the advantage of being aligned both in time and frequency with the piano roll representation of the MIDI input. Noise is synthesized in all frequency bands during note onset times, which corresponds to the impacts of the key on the keyboard base and the hammer on the string. However, the noise spectrum also presents a correspondence between the played notes and the energy location in frequency, which can coincide with the piano soundboard modes, as explained in Section 2.3.3. The model successfully replicated this relationship between the soundboard modes and the input pitch by learning from the target audio data, without the need for explicit modeling of the soundboard modes, which can be quite challenging (Chabassier et al., 2013).

Several participants in the listening test have reported hearing a continuous background noise in the lower frequencies of the samples synthesized by the model. The

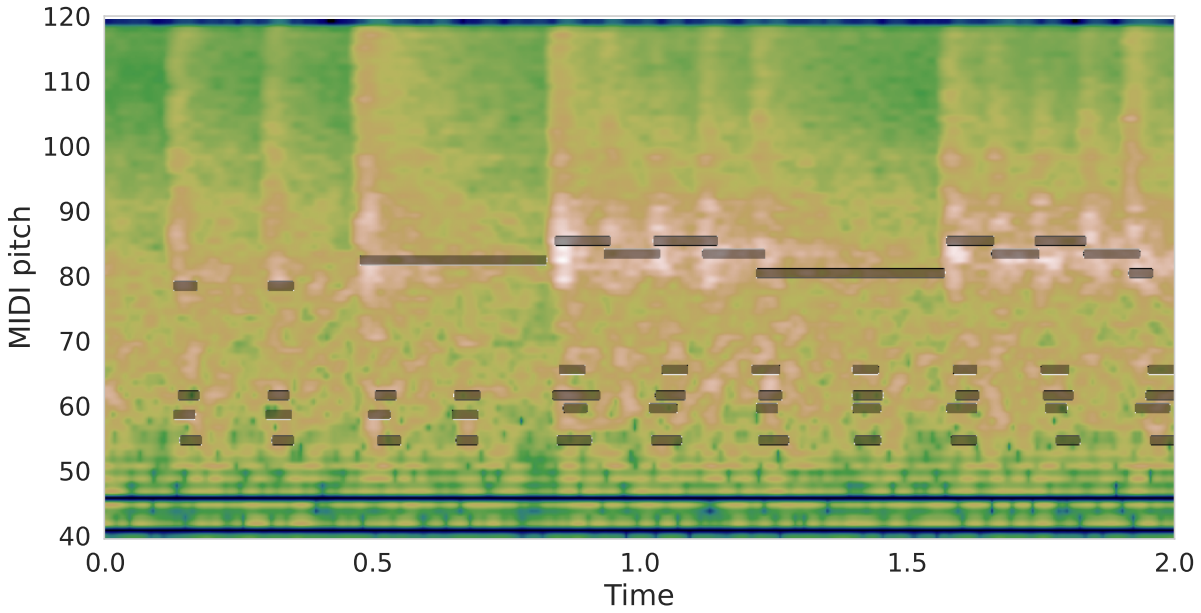


Figure 4.6: MIDI-filter-bank-based spectra (Cooper et al., 2021) of the subtractive synthesizer output. The active piano roll of the input performance is superposed over in black. Both are zoomed around the MIDI range [40,120].

authors have found, empirically, that this noise is generated by the *subtractive synthesizer* and it is different from one piano model to another. This behavior may reflect the varying quality of the recordings in the MAESTRO dataset. Other participants have also shared hearing an "initial impact" at the beginning of the samples synthesized by DDSP-Piano, as if the piano soundboard was excited even before inputting any control. We intuit this as an undesirable side-effect of segmenting the audio and MIDI recordings into chunks, without providing the amount of time segmented notes were already playing before the beginning of a chunk. Smooth transitions between consecutive segments have been addressed by Hawthorne et al. (2022) through providing the previous output segment as a context side-input. Similarly, encoding the previous input segment into the context computation may benefit DDSP-Piano.

Spectrograms of real room impulse responses have been analyzed and demonstrated to have high energy in all frequency bands during the early reflections, and only in the lower frequency bands for the late reverberation (Valimaki et al., 2012). As it can be seen in Figure 4.7, some impulse responses learned by the reverb module of DDSP-Piano do not fully align with these expectations, since one can distinguish modes in the higher frequencies in the late reverberation. When put into relation with the reconstruction quality presented in Figure 4.3, the least well-modeled pianos (most notably from the years 2006, 2004, and 2011) show such modes in their impulse responses estimated by the model. These modes may correspond to either prevalent note partials in the training data (as we observed before adding the L1 reverb regularization constraint) or soundboard modes, which can also be simulated with reverberation algorithms (Bank et al., 2010).

In either case, such features should have been generated by other DDSP components

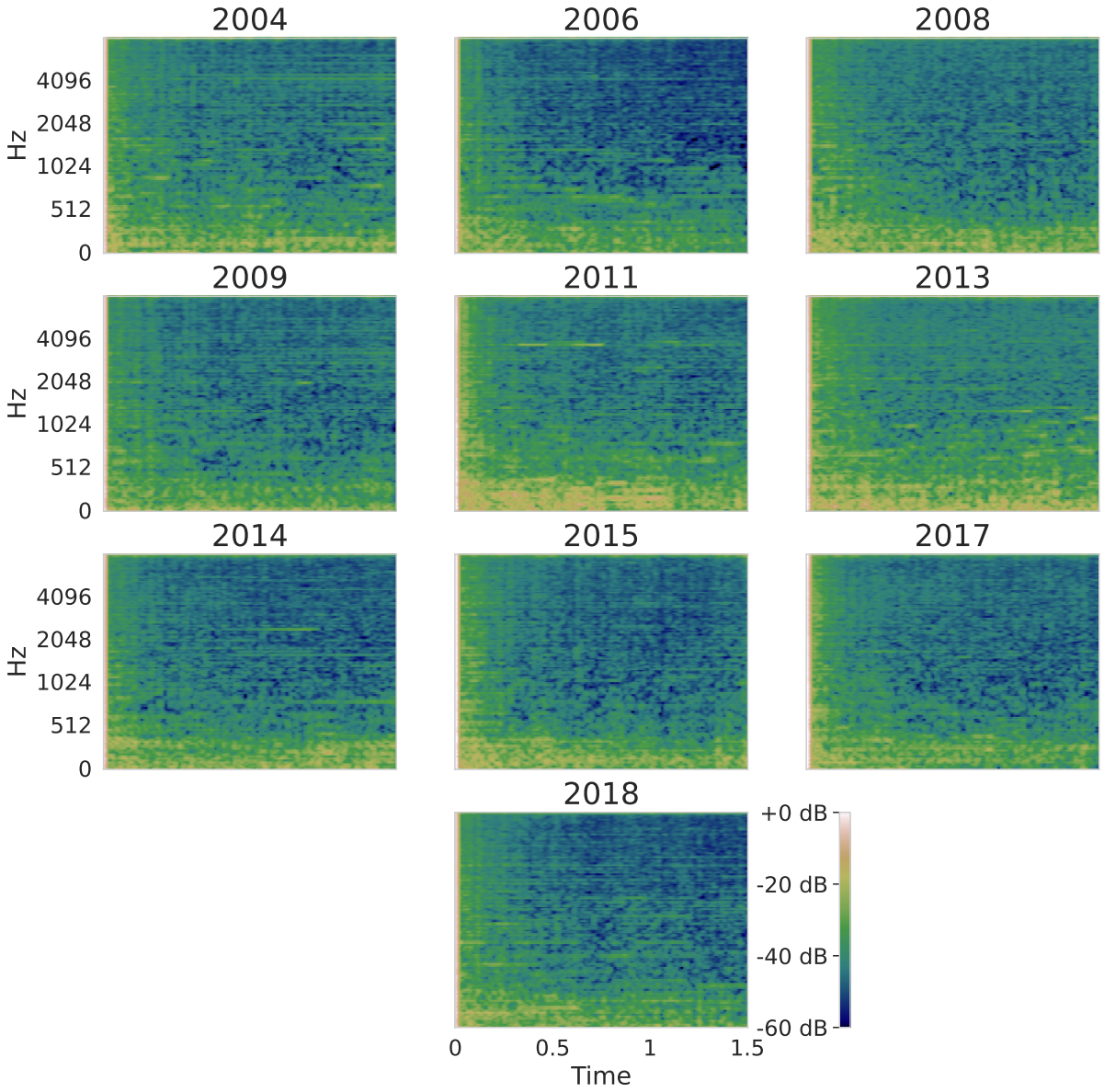


Figure 4.7: Mel-spectrograms of the reverb impulse responses $\{\text{IR}_i\}_{i \leq I}$ learned by the *Default* configuration for each recording year in the MAESTRO dataset.

instead. Despite the L1 regularization, the reverberation module remains too expressive and does not achieve a realistic-sounding reverberation, as it also tries to model behaviors not related to the recording environment.

Section summary - First Iteration of DDSP-Piano

The presented DDSP-Piano model extends the original DDSP work for the task of polyphonic piano audio synthesis from MIDI. It includes high-level modeling knowledge, inherited from physical and signal modeling, to explicitly handle specificities of the instrument, in conjunction with the expressivity of deep neural networks. Through a subjective evaluation, the hybrid model, with significantly fewer parameters, achieves better synthesis quality than a reference neural model. Yet, while the quality is on par with wavetable-based synthesis, it is less realistic than physical-based synthesis. Thanks to its interpretability, further analysis of the model behavior was conducted, revealing that some results found in acoustic modeling works were reproduced by the model from the data. It has also been revealed that the different sound components were not as well disentangled as intended when designing the model architecture, which opens up perspectives for further integration of acoustic modeling knowledge as constraints, and for adapting the training procedure with regard to such constraints.

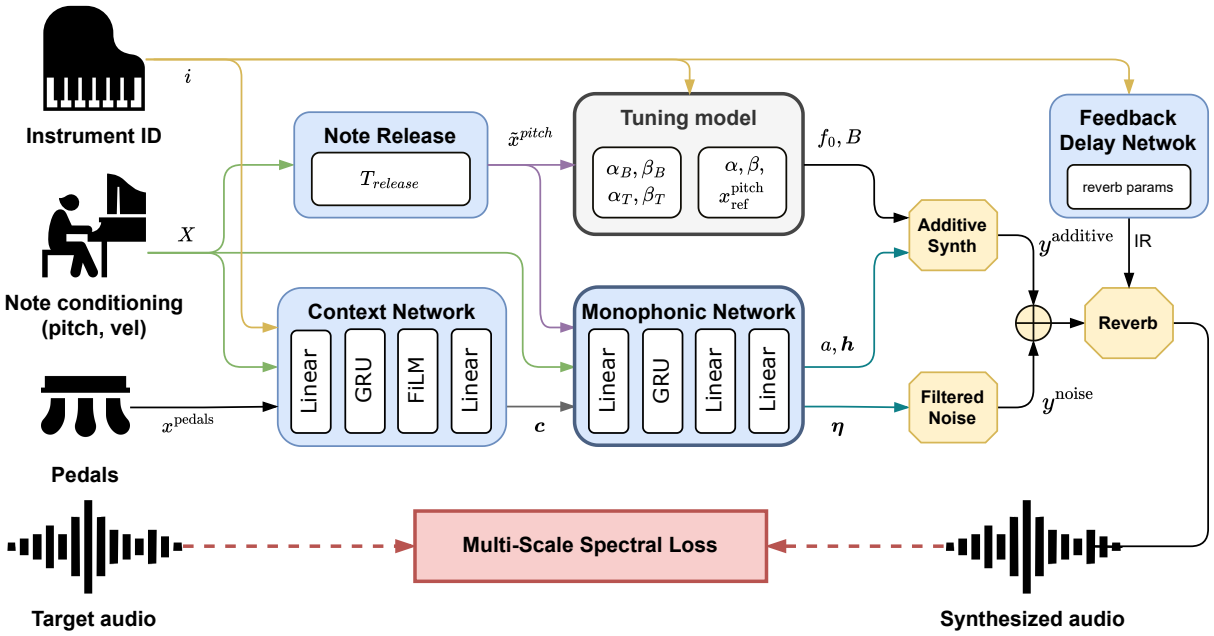


Figure 4.8: Full architecture of DDSP-Piano v2. The blue rounded boxes represent the trained modules for the control of the synthesis. The weights of the grey **Tuning Model** are optimized beforehand and are frozen during neural optimization. Modules with a thickened border are applied along each monophonic voice. Differentiable signal processing and synthesis layers are represented by yellow hexagons (*Additive*, *Filtered Noise*, and *Reverberation*). Finally, the MSS loss compares the input target signal (bottom left) and the output synthesized sound (bottom right).

4.2 Improving DDSP-Piano

The previous section presented the first proposition of a DDSP-based piano audio synthesizer from MIDI. It combines expressive neural network layers with explicit modules that embed modeling knowledge of the instrument: this modular approach allows for tackling specificities of the piano sound in a targeted manner. However, while the overall synthesis quality appears to be quite decent and surpasses a pure neural benchmark, some individual modules did not converge as expected. This section will go over a few proposed modifications to the model, addressing some of these concerns, along with early evaluation results.

4.2.1 Architectural Changes

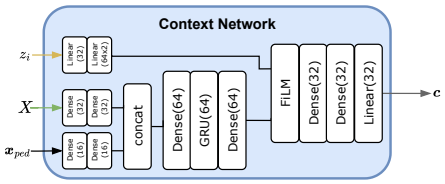
The full architecture of the updated DDSP-Piano is illustrated in Figure 4.8.

The most apparent issue with the first iteration of DDSP-Piano is its inability to fine-tune the frequencies tuning parameters to the target pianos. A few modifications are suggested for the **inharmonicity model** and the **detuner**, while their new optimization strategy will be presented in the next section. The detuner is replaced by the **parametric tuning model** of Rigaud et al. (2011), detailed in Equation 2.14 of Section 2.3.4, that

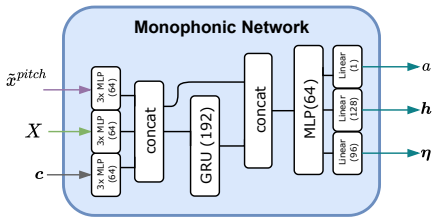
takes the explicit **inharmonicity model** into account for modeling the tuning deviations from the equal temperament. Added parameters are the per-piano reference notes $\{x^{\text{pitch}}_{\text{ref},i}\}_{i\leq I}$, bass asymptotes $\{\beta_i\}_{i\leq I}$, and decrease slopes $\{\alpha_i\}_{i\leq I}$ of the parametric octave type model. Similarly for the **inharmonicity model**, the instrument-specific modifiers $\{\delta_i, b_i\}_{i\leq I}$ are removed in favor of instrument-specific bass and treble linear asymptotes $\{\alpha_{B,i}\}_{i\leq I}$, $\{\beta_{B,i}\}_{i\leq I}$, $\{\alpha_{T,i}\}_{i\leq I}$ and $\{\beta_{T,i}\}_{i\leq I}$.

Moreover, the memory bottleneck of the model lies in synthesizing all $n_{\text{string}} \times K$ individual partials for all P voices. Since retrieving the correct partial beatings through note duplication failed, we set n_{string} to 1 in order to spare some memory space.

Since the instrument-specific modifiers are removed, the **Z-Encoder** is simply integrated into the **Context Network**. Its instrument embedding output is applied through a Feature-wise Linear Modulation (FiLM) layer (Perez et al., 2018), which has notably found usages for global conditioning (Hawthorne et al., 2022). Other modifications to architectures of the context $\mathcal{C}$ and monophonic networks $\mathcal{M}$ are shown in Figures 4.9a and 4.9b.



(a) Updated context Network $\mathcal{C}$.



(b) Updated monophonic network $\mathcal{M}$. MLP are stacks of 3 Dense layers.

Figure 4.9: Updated architectures of the Context and Monophonic networks. Their inputs and outputs remain unchanged from their first version illustrated in Figure 4.2a and 4.2b. Layer normalization is applied before applying the activation function of each Dense layer. Numerical shapes reflect the implementation of the presented DDSP-Piano v2 model.

Another shortcoming of the first DDSP-Piano is its reverb module that has learned unrealistic features for usual room reverberations. Despite an optimization constraint through an L1 regularization, the module remained too expressive. Therefore, the explicitly learned FIRs are replaced by a **differentiable Feedback Delay Network (FDN)-based reverb** module, with implementation and default parameters taken from Lee et al. (2022)[8]. In the same manner as spectral modeling for instrument sound synthesis, the FDN structure is motivated by modeling knowledge of natural reverberations, achieving realistic reverb FIRs with fewer parameters. One can refer to the works of Lee et al. (2022); Santo et al. (2023) for an in-depth explanation of the layer: notably, the early reflections are still modeled by a short FIR filter while the late reverberation is modeled by the FDN structure. The structural constraints inherent to the module should prevent

[8]The re-implementation of this module was done by Pierre-Hugo Vial in the context of the ANR project AQUA-RIUS (ANR-22-CE23-0022).

it from learning unrealistic features and help to achieve better behavior disentanglement between the DDSP components.

Here, the instrument-specific FDN parameters are jointly learned with the other layers parameters of DDSP-Piano. Differentiable reverbs are usually optimized by providing dry and wet audio signals, but we do not have access to the raw piano sound. However, the reverberation is fixed for each recording environment, independently of the notes being played: this signal chain is reproduced in the model architecture, and multiple recordings in the same environments are available in the dataset. Learning through these recordings should let the FDN module capture this invariant filtering of the environment.

Ultimately, to align with the Piano-TTS benchmark, the audio sampling rate is increased to $F_{\text{audio}} = 24\text{kHz}$, which also requires an increase in the number of partials and filter bands to $K = 128$ and $Q = 96$. Also, during inference, a 0.5 second warm-up is applied before synthesizing MIDI files. This warm-up aims to mitigate the "initial impact" issue reported earlier, by running the RNN layers for a few steps with all-zero controls inputted.

4.2.2 Revised Training Procedure

Compared to the initial training strategy presented in Section 4.1.2, we no longer alternate between two phases. Instead, the estimation of frequency-related parameters (from the parametric *tuning model*) is supposed to be completely done in a first stage, then the neural layers parameters are optimized afterward in a second stage. The tuning parameters estimation is detailed afterward. As for the neural optimization phase, since the reverb module was changed, the loss function is simply reduced to the MSS loss between the target and synthesized signals. Other optimization parameters remain unchanged (Adam learning rate, frame rate, segment duration, validation-based early stopping), with the exception of the increased output length due to the audio sampling rate upgrade.

On the Estimation of Tuning Parameters with Gradient Descent

As exposed in Section 3.2.3, DDSP-based synthesizers suffer from convergence issues when estimating partial frequencies from the MSS loss. DDSP-Piano is no exception to this symptom: both the *Deep Inharmonicity* and the *Default* models do not surpass the *No Fine-Tuning* ablated variant, implying that the optimization phase for tuning the inharmonicity and detuning controls failed.

A logical approach to tackle this issue is to investigate if the neural optimization can be improved with the recent propositions, mentioned in Section 3.2.3, which aim to provide informative gradient direction for matching frequencies. Namely, we explored and combined the **surrogate synthesis** method of Hayes et al. (2023), different **variations of the MSS** loss in the manner of Schwär and Müller (2023) and an early adaptation of an **optimal transport-inspired** loss (Latorre et al., 2023) in the spectral domain (similar to the work of Torres et al. (2024)). They have been applied for our specific case of piano note frequency estimation, where the partials can be matched by **unconstrained** oscillators or by **inharmonic-constrained** oscillators: in the first case, each partial frequency has to be estimated individually, while the latter case reduces the problem into

a $\{f_0, B\}$ parameters estimation. We also investigated a strategy inspired by **progressive growing** and Hahn and Roebel (2013), where partials are introduced one at a time, to iteratively refine the inharmonicity estimation. Furthermore, since the MIDI transcript of the piano note is available, we also explored **starting** the f_0 optimization **from the quantized MIDI frequency**: initializing the learning from a starting point close to the real target frequency should help the convergence. Finally, the configurations were tested on **synthetic** inharmonic signals and on **real** piano signals (where partial amplitudes have to be jointly estimated).

During the experiments, we noticed that the most robust configuration for matching the partials of individual real piano notes is using inharmonic-constrained oscillators with the first partial initialized at the quantized MIDI frequency. Other strategies (surrogate, optimal transport, different MSS configuration, and the progressive growing) do not seem to significantly improve the convergence and can be detrimental in some configurations.

However, while it seems feasible to use gradient descent techniques to fit the frequency parameters for a single piano note, the framework fails to generalize on multiple notes to estimate simultaneously. Indeed, given a trainable tuning model (either from Equation 4.4 or with a MLP), estimating the tuning and inharmonicity curves over the pitch range fails when optimizing in the audio modality through the differentiable additive synthesize and the MSS. We noticed that the convergence fails even on synthetic audio and when initializing on the MIDI quantized frequencies. This calls for a two-step optimization scheme, where detuning and inharmonicity coefficients have to be estimated on individual notes, then a tuning model should fit on these estimated coefficients to get a generalized model over the tessitura.

This subdivision of the detuning and inharmonicity curves estimation further reduces the relevancy of using the full DL optimization scheme. Usually, the main advantage of DNNs training is to model complex and implicit relationships in an end-to-end manner. Yet, an ideal full end-to-end training of DDSP-Piano would require estimating the partial frequencies of notes played simultaneously in the polyphonic recordings, which is an even harder setting than the previous fitting on individual notes. Therefore, we leave the neural optimization of the tuning parameters through audio on the full pitch range as future work.

Back to Signal-based extraction

From the attempts at strengthening the neural estimation of the frequency-related layers, we have concluded that the safest approach would be to first estimate the frequencies of individual piano notes, then fit the tuning and inharmonicity models on those estimations, rather than relying on matching through the audio modality. Even for individual notes, neural optimization through DDSP synthesis does not seem reliable enough and we instead employ a more traditional signal-based estimator designed for piano notes.

Thanks to the aligned MIDI data, we first extract all MAESTRO audio segments where only a single note is being played. Redundant notes with extreme velocities (above 100 and below 60) are discarded in order to avoid too much residual noise level that may hinder the peak selection step. Then, the method of Hahn and Roebel (2013) is used for the joint estimation of the notes f_0 and inharmonicity coefficient. The method has

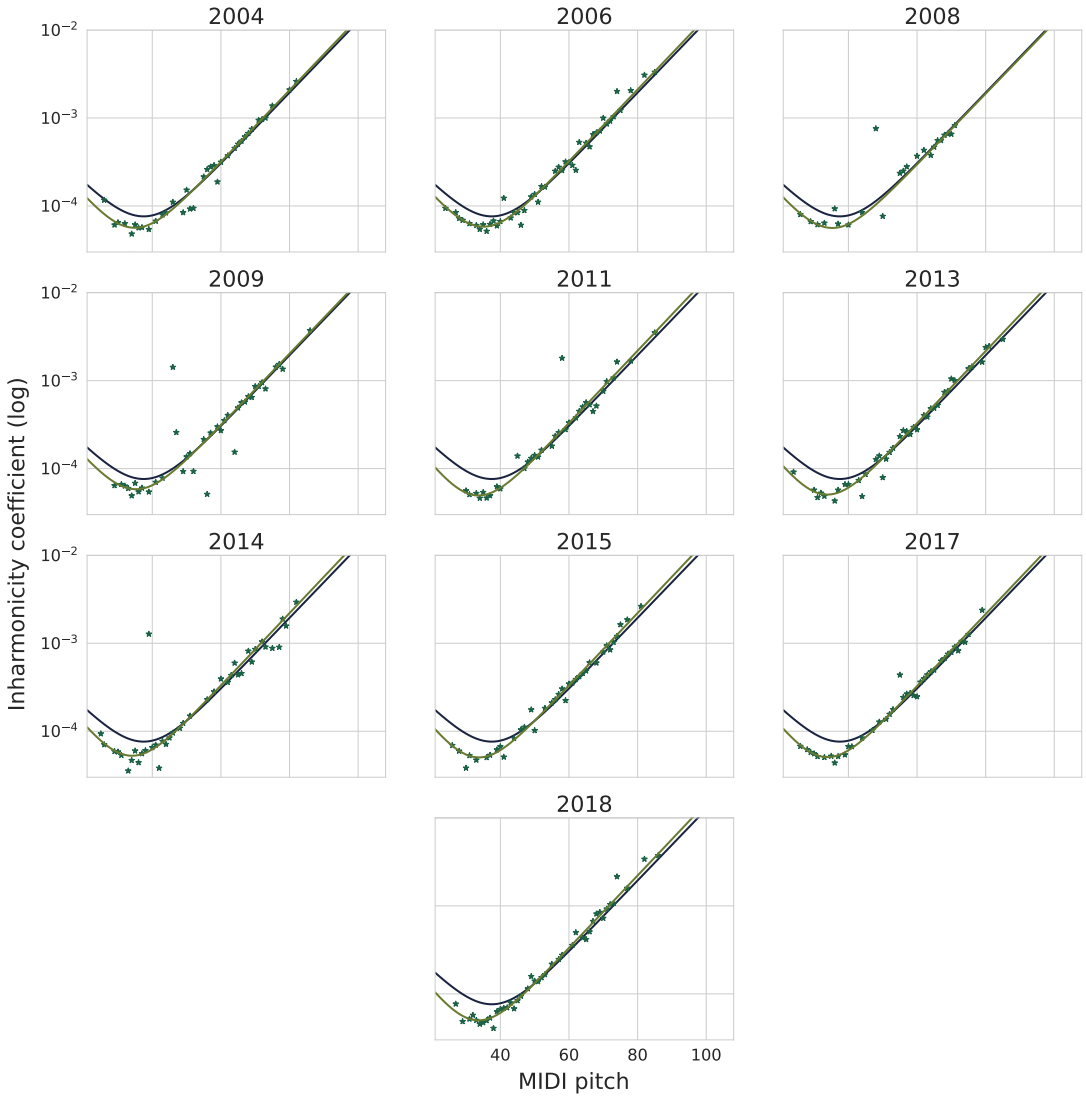


Figure 4.10: Estimated inharmonicity curves over the pitch range on all pianos from MAESTRO. Star points indicate the inharmonicity values extracted from certain notes, while the full green line is the parametric model fitted on those values. In comparison, the dark blue curve represents the inharmonicity curve with the initial parameters from Rigaud et al. (2011).

proven to be more efficient for such estimations than other contemporary approaches. When failing to estimate the inharmonicity coefficient, the method returns the value obtained from the inharmonicity model of Rigaud et al. (2011) fitted on another piano: we thus reject those default estimations. Subsequently, the parametric tuning models presented in Equations 2.13 and 2.14 are fitted piano-wise to the extracted f_0 and B of the notes, using the conjugate gradient descent method, as recommended by Hahn and Roebel (2013).

The estimated inharmonicity curves on the MAESTRO pianos are shown in Figure 4.10. Assuming that the f_0, B extraction is correct, the curves show that the parametric

model needed to be adjusted in the bass range when comparing with the values from Rigaud et al. (2011) used as initialization of the first iteration of DDSP-Piano. Also, they confirm that the inharmonicity values in the treble range are similar between pianos.

Figure 4.11 shows the estimated detuning curves piano-wise: the individual coefficients δf_0 do not seem to follow a smooth curve, yet most of them follow the expected behavior of tuning low notes slightly lower and high notes slightly higher. The noisy curves indicate either a bad estimation of the f_0 values or that the tuning of the MAESTRO pianos should be considered note-wise rather than fitting a model over the tessitura.

4.2.3 Evaluation

Baseline updates

The updated **DDSP-Piano v2** is compared against its first iteration **DDSP-Piano v1**, retrained with increased number of partials and noise filter bands to account for the updated sample rate at 24kHz. Since it does not improve the quality of DDSP-Piano v1, the frequency fine-tuning step is not applied.

A **Regularized** variant of DDSP-Piano v2 is also trained, by adding a constraint to the *early reflections* of the FDN reverb module, in the same manner as the regularization applied for DDSP-Piano v1 (Equation 4.10). Like with the first iteration of the reverb module, the early reflections are modeled through a learnable FIR filter: this ablation study investigates whether further regularization of the reverb (on top of the late reflections simulated by the FDN model) can improve the disentanglement of the piano sound components.

In the meantime, the Piano-TTS model of Cooper et al. (2021) was updated by Shi et al. (2023): this **Piano-TTS v2** model is retrieved from their public repository[9] and selected as the new neural benchmark.

Updated Model	Parameters	Model	Parameters
Piano-TTS v2	31.5M	Piano-TTS v1	31.4M
- Transformer-TTS	17.6M	- Tacotron-2	30.6M
- HiFi-GAN	13.9M	- NSF	736.3k
DDSP-Piano v2	344.5k	DDSP-Piano	512.5k
- Sub-models	341.5k	- Sub-models	281.5k
- Tuning Models	70	- Tuning Models	33
- FDN Reverb	2820	- Reverb	240k

Table 4.3: Approximate parameters count of the updated and previous models. Note that the discriminator of HiFi-GAN has been left out (since it is not used during inference), but it represents 70.7M additional parameters during training.

For assessing the reconstruction quality of the MAESTRO pianos, the Fluidsynth and Pianoteq benchmarks are discarded since they do not reproduce the same piano model,

[9] https://github.com/nii-yamagishilab/midi-to-audio

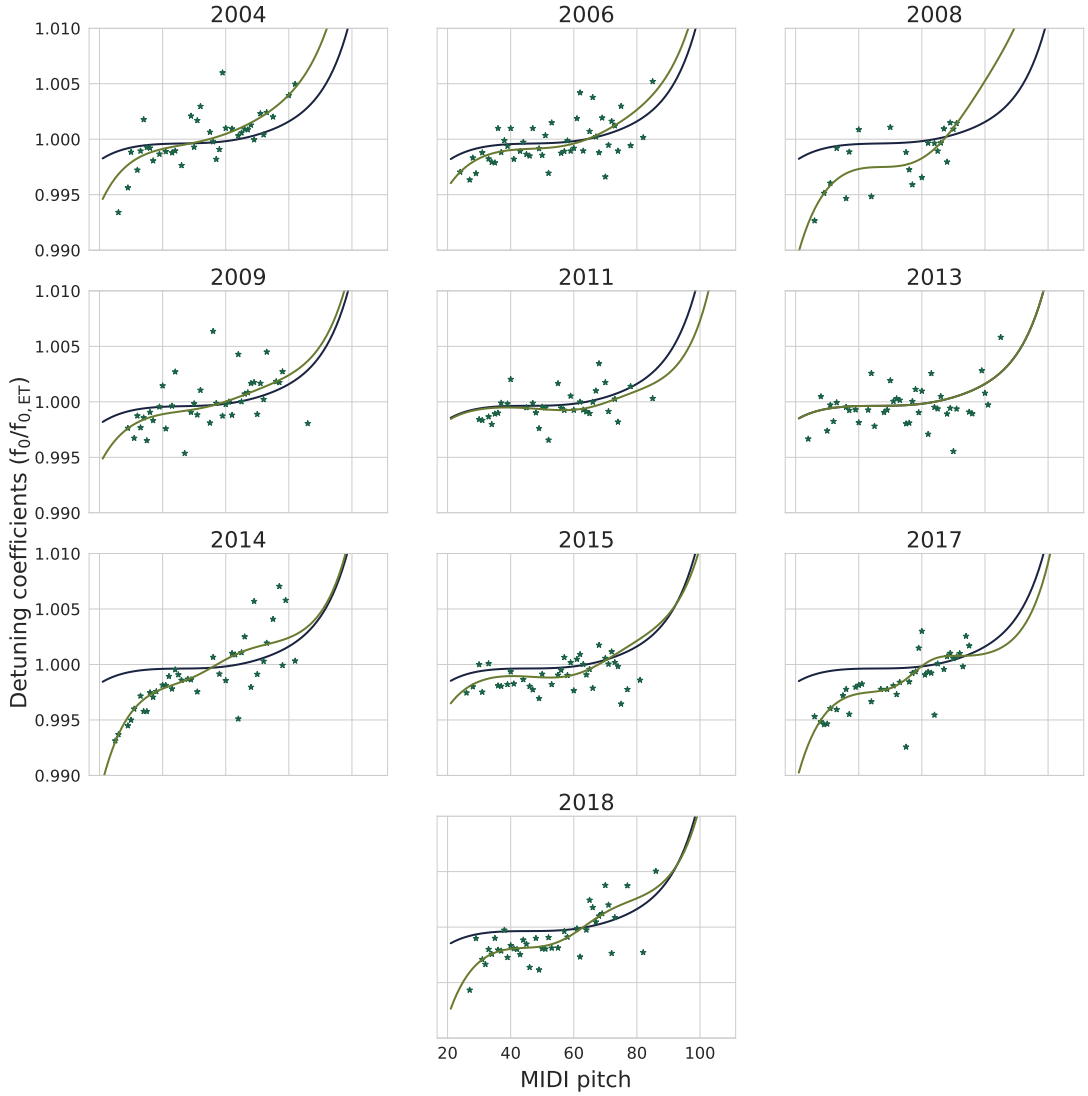


Figure 4.11: Estimated detuning curves over the pitch range on all pianos from MAESTRO. Star points indicate individual detuning values extracted note-wise, while the full green line is the parametric tuning model fitted on those values. In comparison, the dark blue curve represents the detuning curve with the initial parameters from Hahn and Roebel (2013).

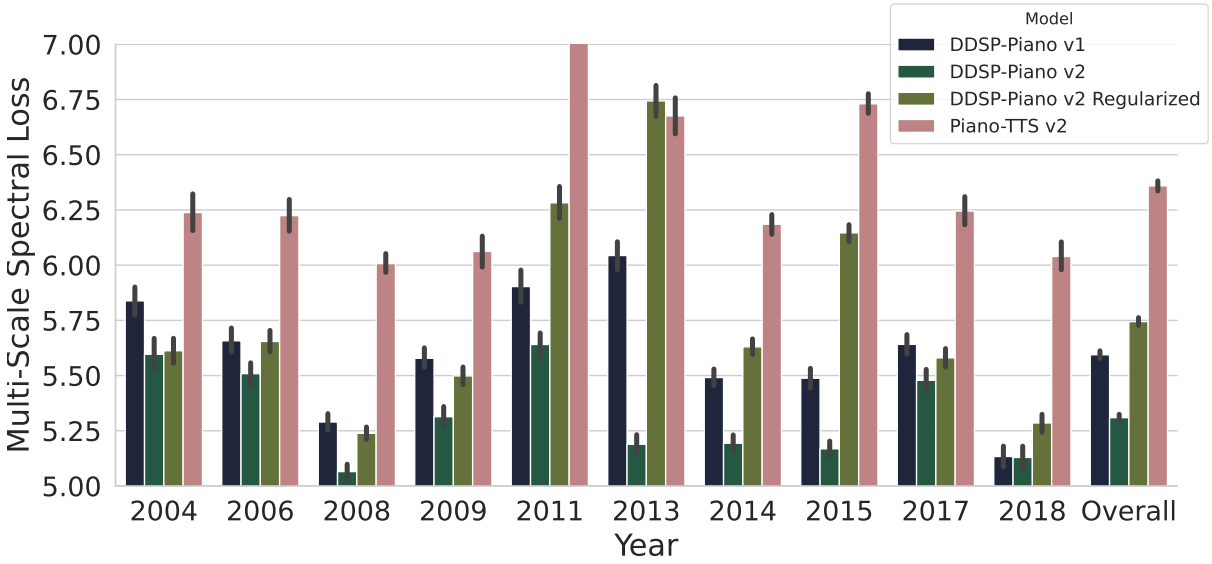


Figure 4.12: Systems re-evaluation on the MAESTRO test set, broken down by recording environments. Measured by mean and standard deviation values of the MSS difference (Equation 3.3) with the original recordings. Lower loss values indicate better reconstruction.

and are thus unrepresentative of sampling-based and physical-based modeling on the MAESTRO pianos.

Objective Evaluation

Figure 4.12 shows the reconstruction quality of the updated DDSP-Piano and Piano-TTS models.

Both the first and second iterations of DDSP-Piano outperform the updated pure neural benchmark on every recording environment, implying that the hybrid approach still holds better reconstruction quality than more advanced TTS techniques.

The second version of DDSP-Piano outperforms the first version on every recording environment, suggesting that the changes made to the model did improve its overall quality. Further ablation studies are necessary in order to pinpoint the most significant improvement, among the slightly deeper neural modules, the revamped modeling and estimation of the frequency tuning, and the modified reverberation layer.

As for regularizing the early reflections of the FDN reverb, the comparison between DDSP-Piano v2 and its regularized variant shows that the reconstruction quality is increased when the reverb is not constrained. As illustrated in Figure 4.13, several room impulse responses modeled by DDSP-Piano v2 (notably from years 2008, 2009, 2011, and 2018) display the expected behavior of real reverberations, with their high-frequency content decaying faster than their low-frequency part. In comparison, the reverberations learned by the Regularized variant, shown in Figure 4.14, have less excessive early reflection coefficients, but they are overall noisier during the late reverberation part. In

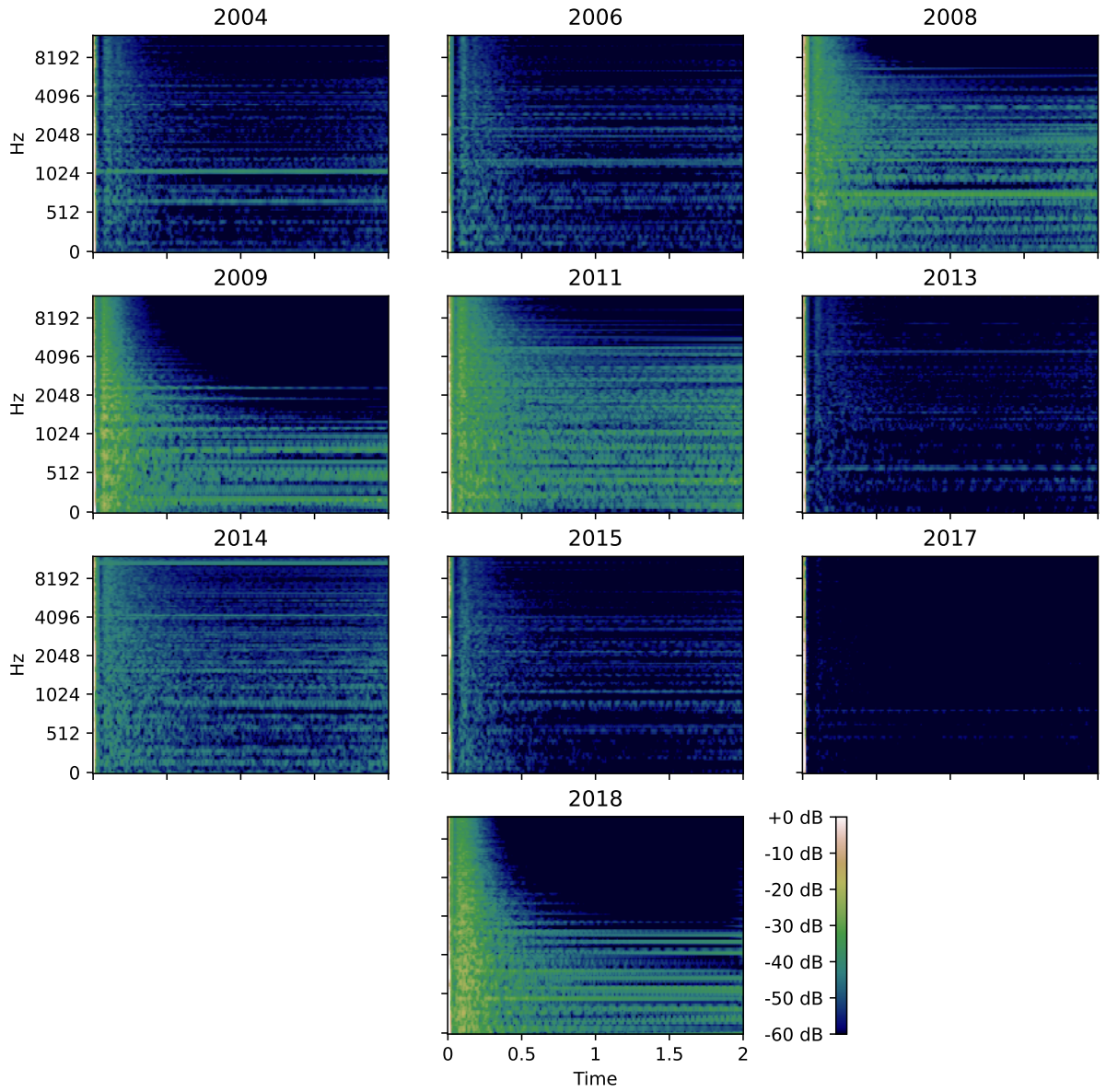


Figure 4.13: Mel-spectrograms of the impulse responses modeled by the FDN reverb layer in DDSP-Piano v2. Loudness values are capped at 0dB for color consistency with other reverb visualizations, but the early part of all impulse responses exceeds +7dB.

both models, very distinct resonances can be seen in the learned reverberations, which is unrealistic for real room responses and it is a known issue of FDN-based models (Valimaki et al., 2012; Santo et al., 2023).

While these results are very encouraging, the improvements need to be confirmed with perceptual tests, which have to be left for future work. Readers are still invited to listen to some synthesis examples on the accompanying website[10].

[10] http://renault.gitlab-pages.ircam.fr/thesis-support/chap_4-2

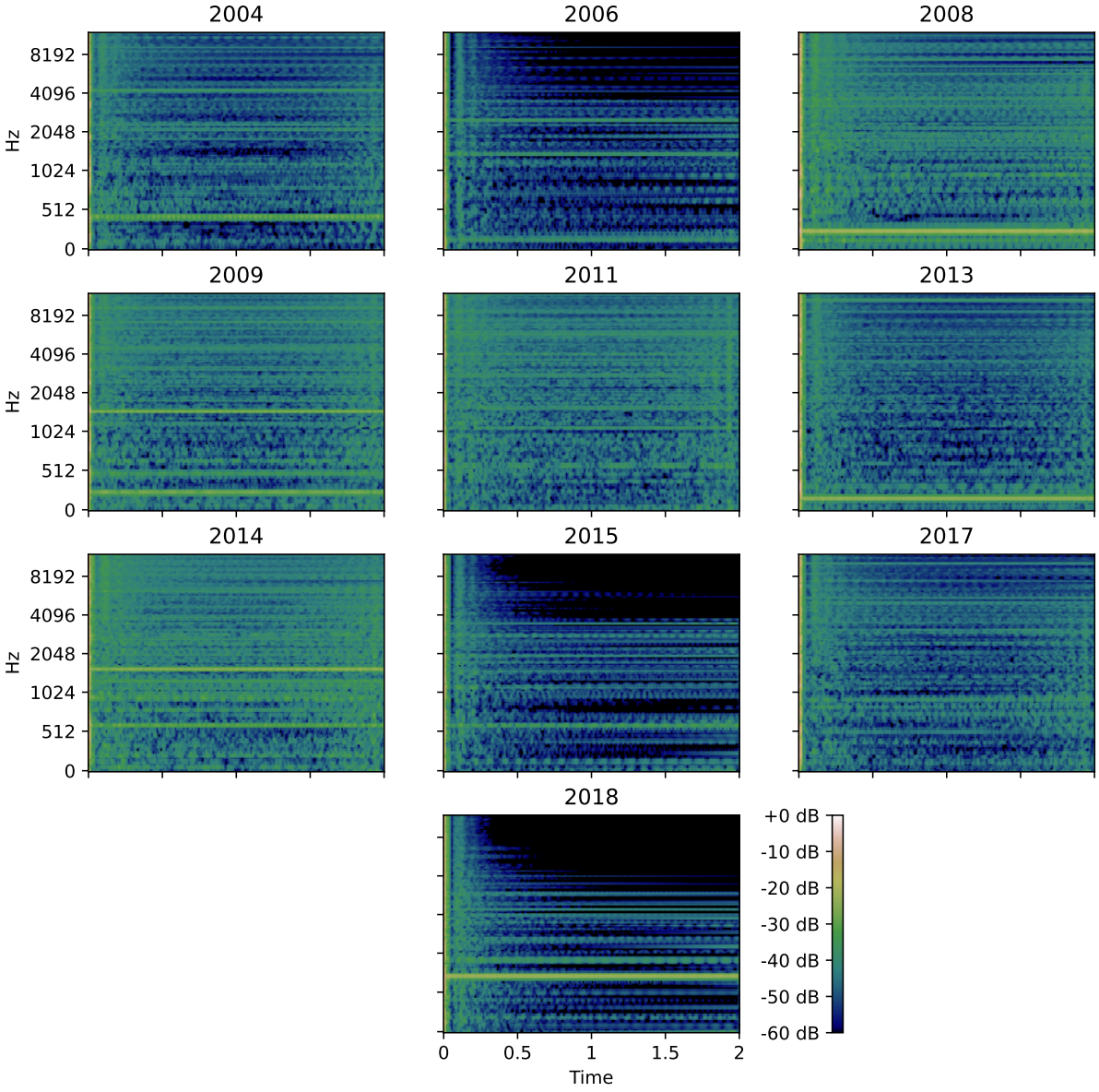


Figure 4.14: Mel-spectrograms of the impulse responses modeled by the *Regularized* FDN reverb layer in the DDSP-Piano v2.

Section summary - Improving DDSP-Piano

The previous section showed several shortcomings of the DDSP-Piano model. Here, multiple improvements are presented to address specifically some of these limitations. The independent Detuner module is replaced by a parametric tuning model that takes the inharmonicity into account for predicting the note deviations from the equal temperament. It is optimized jointly with the inharmonicity model using a more traditional signal-based method, by first extracting the tuning and inharmonicity coefficients from individual notes of the dataset, instead of relying on the spectral reconstruction loss with gradient descent. The context and monophonic neural networks are strengthened by slightly increasing their capacity, to account for the increased audio sampling rate. Also, the reverb layer is changed for a differentiable FDN reverb, whose architecture is motivated by room reverberation modeling and uses less parameters. All the modifications lead to a DDSP-Piano v2 model that displays better reconstruction quality than the first iteration and an improved Piano-TTS neural benchmark, according to an objective evaluation. Further ablation studies and a subjective evaluation still need to be conducted in order to confirm these improvements.

4.3 Discussion

4.3.1 Some Takeaway Lessons

Training a neural piano synthesizer directly on polyphonic performance data enables the reproduction of complex interactions between different sound sources in the instrument. Previous work mainly focuses on achieving realistic-sounding synthesis by adapting generic state-of-the-art NAS models. However, in order to better control the learned model, another challenging issue can be raised in the form of correctly disentangling the sound components; especially when the training data do not present these components separately.

In the continuity of the original DDSP model (Engel et al., 2020a), the proposed DDSP-Piano model further incorporates signal-based and acoustic-modeling knowledge into the differentiable framework for handling multiple instrument specificities. Following the taxonomy of knowledge-inclusion presented in Section 2.4.4, this knowledge is integrated as architectural constraints through explicit sub-models (with the parametric *tuning* and *inharmonicity* models), layers connections motivated by meaningful inputs and outputs (by knowing which variable contributes to which phenomenon) and variable processing (with the *monophonic network* being applied on each monophonic channel for example). The full model successfully achieves better-sounding quality than a pure neural-based synthesizer and shows promising results for disentangling the different sound components.

Such a hybrid framework allows for targeted observation/evaluation of the learned behaviors and puts them into perspective with properties inherited from acoustic and signal-based modeling. This allows for targeted improvements on sub-models in order for them to correctly match their expected behaviors, which was the core motivation behind Section 4.2. Such improvements can be made by leveraging more flexible neural approaches that can reproduce behaviors difficult to model explicitly, such as the soundboard modes in our case. Yet, they can also be responsible for the imperfect disentanglement of sound components between each other. Hence, improvements can be made instead by further including knowledge through more specialized or explicit sub-modules. However, further structuring the architecture hinders the convenient neural optimization, and intermediary optimization steps may be necessary, as it was done with the tuning process in Section 4.2.2.

Finally, the integrated constraints require the data to fulfill associated assumptions: here, DDSP-Piano would not be able to profile instruments with extreme properties, such as highly detuned pianos. On the other hand, if the data fulfills the assumptions, less training data will be required when compared to the case when models do not incorporate any constraints. Future physics-informed models should thus balance the neural network expressivity and the instrument knowledge constraints to the quantity of training data available and the desired model flexibility.

4.3.2 Future works

As mentioned previously, the model and its components can either be improved by including more advanced neural layers and optimization techniques for better flexibility and expressiveness, or by leveraging more physical-modeling knowledge to further constrain them to the specific instrument. Namely:

- the **harmonic content** would benefit from a more robust frequency estimation method that would, ideally, train the frequency parameters jointly with the other piano parameters. Also, more advanced string properties mentioned in 2.3.2 were left out of the model and would help to reach a more realistic piano sound: in particular, the longitudinal modes and phantom partials have been partially included in the differentiable piano model of Simionato et al. (2024). At last, the bank of additive sinuses to generate is memory consuming, which calls for exploring other differentiable operations synthesizing controllable inharmonic spectra.

- The main limitation regarding the current partials **energy distribution** is the lack of a satisfactory model of the partial amplitude modulation. We would argue that optimizing the partial beating with gradient descent techniques can suffer from non-convexity issues in the same way frequency estimation fails with an additive synthesizer and the spectral loss. Indeed, beatings occur regularly at a certain rate that needs to be estimated: to this end, Berendes et al. (2023) have proposed an auto-encoder structure for synthetic piano sounds, while Vahidi et al. (2023) could retrieve modulation parameters leveraging the JTFS loss that exhibits higher structures information than the regular MSS.

- the **residual noise** can also be refined by adding transient modeling, in the manner of Shier et al. (2023), to account for faster percussive elements during note onsets.

- as for **reverberation** modeling, the included FDN layer achieves better realism than raw FIR modeling, but it still exhibits unnatural resonances that could possibly be suppressed with the colorless variant of Santo et al. (2023).

Future works could also leverage the interpretability and the differentiability of DDSP-Piano to address other polyphonic music-related tasks, such as source separation (Schulze-Forster et al., 2023) and self-supervised multi-pitch transcription (Engel et al., 2020b).

4.4 DDSP-Piano, in short

Chapter summary - DDSP-Piano

This chapter presented the main thesis contribution for the task of piano audio synthesis: DDSP-Piano, a differentiable hybrid synthesizer expanding on the DDSP framework for handling polyphonic MIDI inputs and audio output. In tandem with the sound structure priors inherent to the differentiable spectral modeling components of DDSP, the model further incorporates high-level knowledge of the instrument to specifically tackle particularities of the piano sound. Namely, the explicit tuning and inharmonicity sub-modules contribute to the quality of the lightweight and interpretable model, which surpasses a pure neural benchmark according to a conducted listening test. Quantitative and qualitative evaluation of the approach trained on MAESTRO have revealed that the model has successfully reproduced the global spectral envelopes and has produced perceptually convincing residual noises, while the frequency estimation of the partials and the reverb were identified as unsatisfying. To improve these shortcomings, a second iteration of the model has been proposed: it leverages a structurally constrained differentiable reverb layer, and a revised tuning and inharmonicity estimation procedure. The frequency tuning step repurposed a more traditional signal-based method, since the gradient descent-based estimation fails, as found in the literature of DDSP-based models. Early objective evaluation of the revised approach shows that the overall sound quality has improved thanks to these modifications, even in comparison with a more refined neural benchmark. While subjective evaluation of this second iteration of DDSP-Piano needs to be conducted, the model can still be improved on multiple levels, whether on specific elements of the piano sound or the unsupervised disentanglement of these elements, in order to reach the quality of real and physical-based piano models. Nonetheless, the chapter has explored the hybridization of DL with domain knowledge in the context of NAS, from which arise both welcomed interpretability and efficiency properties, but also optimization and disentanglement issues. The balance between the introduced knowledge and the flexibility of DNNs has to be adjusted following the task at hand and the resources available.

Chapter 5

Piano Performance Rendering from Unpaired Data

This chapter exposes the second main contribution of this thesis, which is a piano performance rendering model trained from unpaired data in a low-informed setting. Through adversarial training, the model can transfer expressive qualities found in real performances into new compositions, without seeing how the real performances differ from their original scores in the first place.

Section 5.1 will highlight the motivations behind using unpaired data and not relying on high-level score markings. Section 5.2 will then present the model of Renault et al. (2023b) that performs performance rendering only in the symbolic modality, in the continuity of previous works. Then, Section 5.3 will introduce a cross-modal extension of the model that learns expressive qualities in the audio modality and transfers them to symbolic compositions, leveraging the differentiability of DDSP-Piano. The models presented are discussed in Section 5.4, before a chapter summary in Section 5.5.

Accompanying this chapter, audio examples of the performances rendered by the models can be found online[1].

5.1 Motivation

Following the literature review for performance rendering depicted in Section 3.3, three main requirements for handling the task can be exhibited:

1. **Symbolic representation of performances.** Piano performances are mostly handled in the symbolic modality, usually with the MIDI format. This choice is motivated by the fact that such mid-level encoding is sufficient for grasping all controls inputted by the performers for making expressive interpretations. However, retrieving performances in the symbolic modality requires either a dedicated instrument (such as Disklaviers presented in Section 3.1 and various MIDI controllers) or an audio-to-MIDI transcription model that may introduce transcription errors.

[1] http://renault.gitlab-pages.ircam.fr/thesis-support/chap_5-2

2. **Align performances with scores.** Piano performances are defined as the difference in timing, articulation, and velocity of the played notes compared to the raw and unexpressive rendition of the score. Gathering such features necessitates finding MIDI performances with their associated digital scores and aligning them at note-level. These matching and alignment steps constrain the amount of training data (as shown in the dataset comparison Table 3.1), especially in comparison with the large number of piano audio recordings available.

3. **Use score-specific markings.** The majority of research is highly informed and entailed to the digital score format, as they take different markings (such as rests, position in measure, ornaments, slurs, beams, etc...) into account for guiding the expressive rendering. This reliance on score-specific markings hinders their usage in modern music production frameworks (DAWs, sequencers, live hardware) that do not edit symbolic music on scores but rather manipulate MIDI data directly. Considering simple MIDI sequences as scores can be misleading: for example, the default MIDI tempo is 120 BPM, which may not correspond to the effective tempo in the programmed composition.

These last two points have begun to be circumvented by Zhang and Dixon (2023), which models latent spaces of scores and performance styles directly from performances and ignores score features altogether. However, they still need multiple performances of the same piece by different musicians in order to disentangle the score content from the performing style. Also, while their model is capable of transferring style from one performance to another while preserving the content, they have not included raw scores as an unexpressive style and have not shown how the model behaves on scores deprived of style.

Concurrently, GANs have been successfully applied for various tasks that transfer data from one domain to another without aligned pairs, such as image-to-image translation (Pang et al., 2022), audio timbre matching (Wright et al., 2023) or music genre transfer (Brunner et al., 2018). In light of such results, this work attempts to address all three points by viewing expressive performance rendering as a *domain transfer task*. To this end, an adversarial approach is employed to transform MIDI scores into human-like performances without supervision of the performance features and reliance on score markings. A first method that still uses symbolic performances is presented in Section 5.2 along with related experiments. Then, partially addressing Point 1., a cross-modal extension is made in Section 5.3 to learn expressive features from audio performances. Both methods are trained on publicly available datasets that are larger than those eligible for models requiring aligned score-performance pairs.

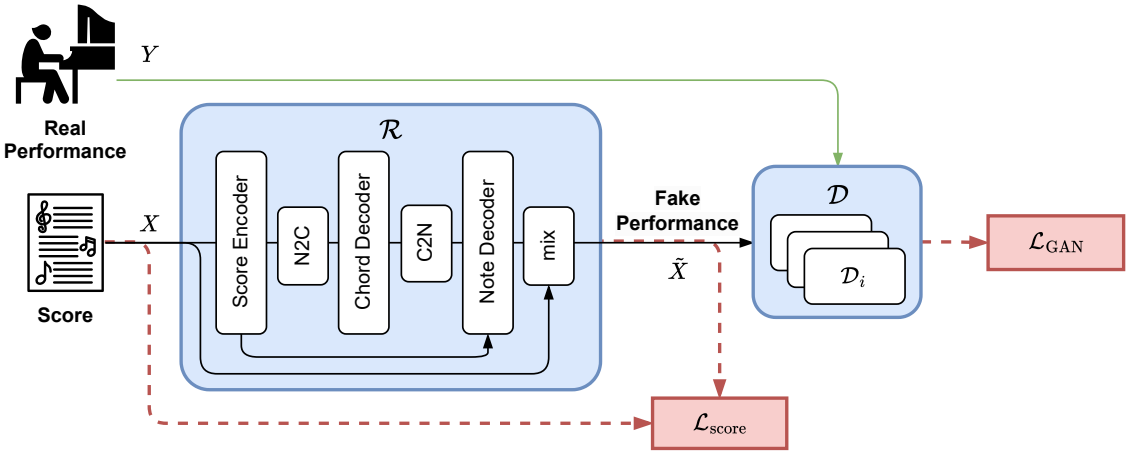


Figure 5.1: Training pipeline for the symbolic performance rendering model $\mathcal{R}$: its final mix function modifies the score X with modifying features output by the Note Decoder, in order to deceive the discriminators $\mathcal{D}_i$. During training, the unaligned score X and performance Y are drawn at random from their respective sets.

5.2 Render Performances in the Symbolic Modality

The proposed approach, illustrated in Figure 5.1, is composed of a performance rendering model $\mathcal{R}$ that takes a score X as input and produces an expressive interpretation $\tilde{X}$ in the symbolic modality. The rendered performances are fed into a discriminator $\mathcal{D}$, among performances Y from a dataset of recorded human performances. The performance rendering model and the discriminator have opposed objectives, as the discriminator $\mathcal{D}$ aims to differentiate the real performances from the ones rendered by the model $\mathcal{R}$, while the latter tries to produce performances indistinguishable from the real ones.

5.2.1 Models Architecture

Both the scores X and real performances Y are encoded with the note-wise representation mentioned in Section 5.2.1, with the *minimal* amount of features needed to describe them:

$$X = \{\boldsymbol{x}_n\}_{n\in[\![1,N]\!]} = \{p_n, o_n, d_n, v_n\}_{n\in[\![1,N]\!]}. \tag{5.1}$$

The N notes are ordered by their absolute onset time: for the n-th note, p_n is its normalized MIDI pitch, o_n its delta-time with the previous note onset, or *relative Inter-Onset-Interval (IOI)*, capped at 4 seconds, d_n its duration in absolute time and v_n its normalized MIDI velocity. While this encoding is common for representing performances, using it as well for encoding the scores ensures compatibility with any composition in the MIDI format and allows the usage of domain transfer techniques.

Performance Rendering Model

The penultimate layer of the performance rendering model $\mathcal{R}$ predicts modifying features ΔX from the score note features in order to modify them into performance-like note

features $\tilde{X}$ through the final mix function:

$$\begin{aligned}\tilde{X} &= \mathcal{R}(X) \\ &= \text{mix}(X, \Delta X) \\ &= \{p_n, o_n + \delta o_n, d_n \delta d_n, v_n \delta v_n\}_{n \in [\![1,N]\!]},\end{aligned} \tag{5.2}$$

with δo_n the micro-onset timing, δd_n the articulation and δv_n the expressive velocity of the n-th performed note.

These modifying features are obtained by first processing the note-wise score features with a MLP Score Encoder. The encoder is composed of two successive Dense layers with Leaky ReLU activation and batch normalization, and a 15% dropout between them.

Then, the same hierarchical modeling from Rhyu et al. (2022) is applied: the note-wise features are merged into chord-wise features, which enables a more coherent modeling of the full sequence. This note-to-chord operation, or $N2C$, is performed by average pooling the features of simultaneous notes into a common chord-wise feature. The inverse operation $C2N$ can later convert chord-wise features into note-wise features by duplicating the chord feature for each of its notes. Both operations require the note-to-chord alignment matrix $M \in \mathbb{R}^{N_{\text{chords}} \times N}$, with N_{chords} the number of chords in the sequence. On the contrary of other hierarchical strategies employed in the literature (Jeong et al., 2019a,b), the note-to-chord alignment matrix M can be directly extracted from our low-informed MIDI data representation. Indeed, $o_n = 0$ indicates that the n-th note is played simultaneously with the previous one, i.e. they belong to the same chord: N_{chords} is thus the number of non-zero IOI in the note sequence, and M can be built from $\{o_n\}_{n \leq N}$ since the notes are ordered.

$$\begin{aligned}\text{N2C}(x) &= \frac{Mx}{\sum_{n=1}^{N} M_{1:C,n}} \\ \text{C2N}(x) &= M^T x\end{aligned} \tag{5.3}$$

Before returning to the note-granularity, the chord-wise features are further processed by a Convolutional Recurrent Neural Network (CRNN) Chord Decoder, composed of a Dense layer with LeakyReLU activation, followed by three CNN layers with increasing output dimensions (32, 64, 128), batch normalization, Leaky ReLU activations and intermediary dropout, and finally a bidirectional GRU and a Dense layer with Leaky ReLU activation. The choice of making the GRU layer bidirectional is motivated by the fact that performers anticipate future notes to guide the expressive direction taken for the notes being played.

Finally, fine-grained adjustments at the note-level are made with the Note Decoder (composed of 3 CNN layers) and a skip connection from the note-wise score encoding. The final micro-onset timing δo_n is obtained through a Dense layer with linear activation function, while the articulation δd_n and the expressive velocity δv_n are mapped to the $[0.25, 4]$ range with the scaled sigmoid function sigmoid† (Equation 3.2) after their respective linear layer.

Layer	Output Dimension	Other parameters
Dense	4	
Conv1D	128	$Q = 3$, BN, dropout
Conv1D	128	$Q = 3$
Dense	128	
bi-GRU	128	dropout
Conv1D	64	$Q = 9$, BN, dropout
Dense	32	
Dense	64	BN
Linear	1	

Table 5.1: Sequential architecture of a note-wise performance discriminator. All layers have Leaky ReLU activation functions (except for the final linear layer). Q indicates the kernel size of the convolutional filters. BN indicates that batch normalization is performed before applying the activation function. Dropout of 15% is applied after the concerned layers.

Discriminator

Taking inspiration from speech processing (Kumar et al., 2019), we use a multi-scale discriminator with $N_{\mathcal{D}} = 3$ sub-discriminators $\mathcal{D}_i$ with identical architectures. The architecture, laid down in Table 5.1, is mirrored from the performance rendering model $\mathcal{R}$, with the exceptions of the $N2C$ and $C2N$ operations, as chords in real performances are not as easily defined as in scores. Each discriminator is fed with a downsampled sequence of (real or rendered) performance notes by average pooling with sizes $\{1, 3, 9\}$. Discriminators with longer pool sizes look at features at higher levels in the performances and thus, can help transferring such knowledge and long-term coherence to the performance rendering model $\mathcal{R}$. To stabilize the GAN training, Gaussian noise is added to the inputs of the discriminators, as in Brunner et al. (2018).

5.2.2 Training Strategy

Loss functions

The unsupervised LS-GAN formulation (Equation 2.23) is used to train the discriminators and the performance rendering model. Their respective loss functions $\mathcal{L}_{D_i}$ and $\mathcal{L}_{\mathcal{R},\text{GAN}}$ are defined as:

$$\begin{aligned} \mathcal{L}_{\mathcal{D}_i} &= \mathop{\mathbb{E}}_{Y\sim\mathcal{P}_{\text{perf}}} \left[\|\mathcal{D}_i(Y) - 1\|_2\right] + \mathop{\mathbb{E}}_{X\sim\mathcal{P}_{\text{score}}} \left[\|\mathcal{D}_i(\mathcal{R}(X))\|_2\right], \\ \mathcal{L}_{\mathcal{R},\text{GAN}} &= \mathop{\mathbb{E}}_{X\sim\mathcal{P}_{\text{score}}} \left[\sum_{i=1}^{N_{\mathcal{D}}} \|\mathcal{D}_i(\mathcal{R}(X)) - 1\|_2\right]. \end{aligned} \tag{5.4}$$

We have observed that the instability of the vanilla adversarial training leads the performance rendering model to displace the notes in extreme values, causing the original

piece to be unrecognizable. To ensure that the performances remain fairly close to their scores, an additional regularization term $\mathcal{L}_{\text{score}}$ is added:

$$\mathcal{L}_{\text{score}}(X) = \boldsymbol{\lambda}_{\text{score}} \| \frac{\mathcal{R}(X) - X}{X} \|_2, \tag{5.5}$$

with $\boldsymbol{\lambda}_{\text{score}}$ a fixed vector weighting how much each performance component (timing, articulation, velocity) can deviate from the score indication. Here, $\boldsymbol{\lambda}_{\text{score}} = \{1, 1, 0.1\}$. Comparing the predicted deviations with the original score features with relative difference instead of the absolute difference is musically more meaningful as large deviations are more detrimental for fast passages (where IOI between notes are short) than in slow passages.

The total loss for the performance rendering model $\mathcal{R}$ is:

$$\mathcal{L}_{\mathcal{R}}(X) = \lambda_{\text{GAN}} \mathcal{L}_{\mathcal{R},\text{GAN}}(X) + \mathcal{L}_{\text{score}}(X), \tag{5.6}$$

with λ_{GAN} the balance between the GAN objective and the score regularization loss. This balance is decisive for the final behavior of $\mathcal{R}$ since the two loss components have opposite influences on its training: $\mathcal{L}_{\text{score}}$ prevents $\mathcal{R}$ from modifying the scores while $\mathcal{L}_{\mathcal{R},\text{GAN}}$ encourages exploring different interpretations in order to deceive the discriminator. In our experiments, $\lambda_{\text{GAN}} = 2$.

Both models are trained simultaneously with their respective Adam optimizer and a learning rate of 10^{-5}.

Datasets

Training and evaluation were conducted using the scores available in ASAP and the MIDI recorded performances of MAESTRO. At the time, ASAP was the largest dataset of piano scores publicly available, while MAESTRO is still the largest dataset of recorded MIDI performances in duration (ASAP surpasses it in terms of number of performances, as shown in Table 3.1). Since the proposed method does not require aligned scores and performance, the entirety of both datasets can be used, which amounts to 962 training performances, 137 validation performances, 107 training scores, 15 validation scores, and 35 test scores (following the train-validation-test split of Liu et al. (2021)).

The velocity indications were kept from the ASAP scores in MIDI format, which can either be constant throughout the piece or mapped from the score nuances and markings using simple rules. The scores and performances are split into segments of 128 consecutive notes, with random pitch shifting during training by ±7 semi-tones for data augmentation, as done by Maezawa et al. (2019). Validation data is used to monitor and avoid potential over-fitting of the performance rendering model by reproducing the training performances from their corresponding scores.

5.2.3 Results and Analysis

A short listening test has been conducted to evaluate the interpretation quality of the performances rendered by the model. 7 scores from the ASAP test subset were selected, covering 5 different composers. 4 MIDI performances were generated by different methods for each score:

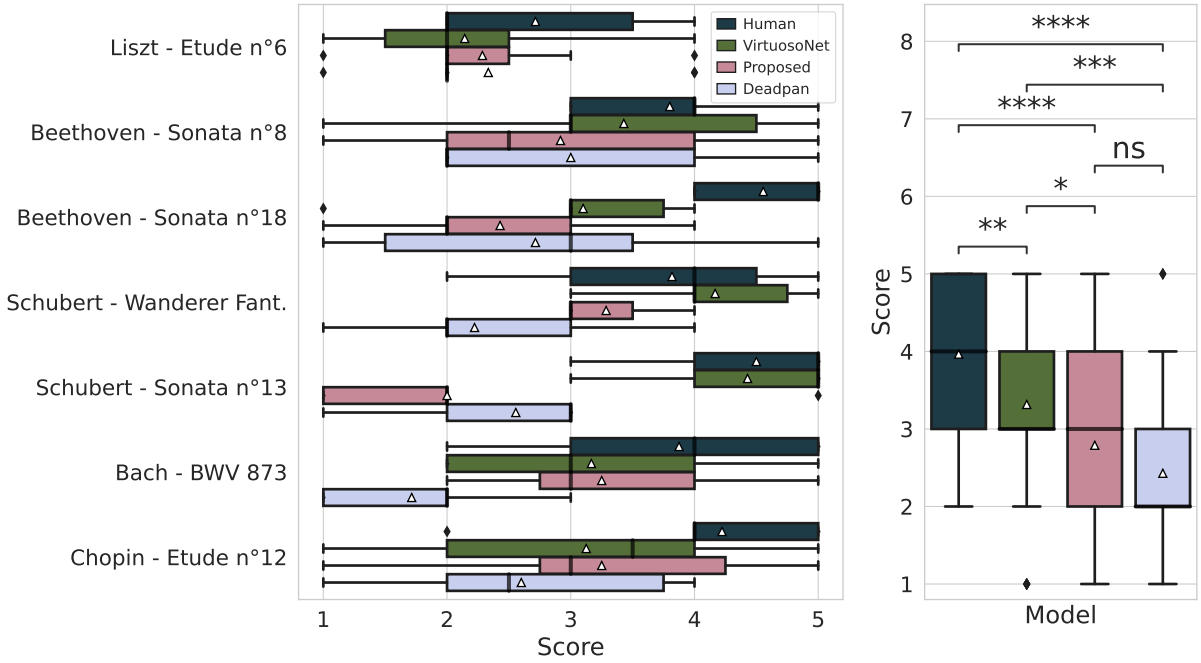


Figure 5.2: Box-plot of the MOS of the different performance rendering methods: piecewise at the left and overall at the right, with Holm-Bonferroni corrected two-sided Mann-Whitney U tests. The thickened bars indicate the median values while the white triangles indicate the mean values. p-value annotation legend: `ns` for $p > 0.05$; * for $p \leq 0.05$; ** for $p \leq 0.01$; *** for $p \leq 10^{-3}$ and **** for $p \leq 10^{-4}$.

- **Human** is a corresponding human performance from the ASAP dataset.
- **Deadpan** is the direct export of the MIDI score.
- a rendition by our **Proposed** approach.
- a rendition from the graph-based variant of **VirtuosoNet** (Jeong et al., 2019b), a highly-informed model using score markings in MusicXML format and is trained with a private dataset of 226 scores matched and aligned with MAESTRO performances, which is larger than ASAP.

The first 20s of each performance were synthesized using the Arturia Piano V3 software [2], a physical-based piano synthesizer. 19 professional audio and piano players were asked to rate the naturalness of the presented performances, using a 5-point Likert scale (from 1 - Bad, to 5 - Excellent). Each trial randomly presented 3 different performances from each method. Results are reported in Figure 5.2.

The Holm-Bonferroni corrected two-sided Mann-Whitney U tests indicate a statistical difference at $\alpha = 0.05$ between the Human rendition and each other methods, and between VirtuosoNet and Deadpan. The overall results show that the proposed approach does enhance the scores with expressive features in comparison to the raw rendition of the

[2] https://www.arturia.com/products/software-instruments/piano-v/overview

piece, but still not with the same amount of naturalness as actual pianists and the highly-informed VirtuosoNet. This was to be expected as our proposed unsupervised training task without score markings is harder than the training objectives of VirtuosoNet, for about the same quantity of training data.

By examining the ratings piece-wise, one can notice the poorer renditions of the proposed method for slower tracks (Schubert's 13th Sonata and Beethoven's 18th Sonata). This may suggest that the model lacks an understanding of the global musical content of the scores and applies similar modifying features for every track, which renders inappropriate performances for slower-paced compositions. However, we have observed during preliminary experiments that some other configurations of the model (with different loss weightings for instance) do not exhibit such an issue, but they render less realistic performances overall than the presented configuration. Such sensibility to training hyperparameters is typical of GANs and we hope strengthening the score understanding of the model would reduce this instability.

Symbolic performance rendering with unpaired data

The presented model addresses piano performance rendering in a novel setting, without learning from score-performance pairs and ignoring markings exclusive to the music sheet format. Instead, it views the task as a domain transfer task in the symbolic modality: using a multi-scale performance discriminator, the model transforms sequences of notes from the unexpressive score domain into the expressive performance domain. In this manner, a score dataset and a performance dataset can be used for training without going through the endeavor of finding and aligning matching pairs. An early subjective evaluation shows that the model shows expressive qualities in the performance renditions in comparison to the plain score, although not with the same naturalness as a fully supervised approach and the human interpretations.

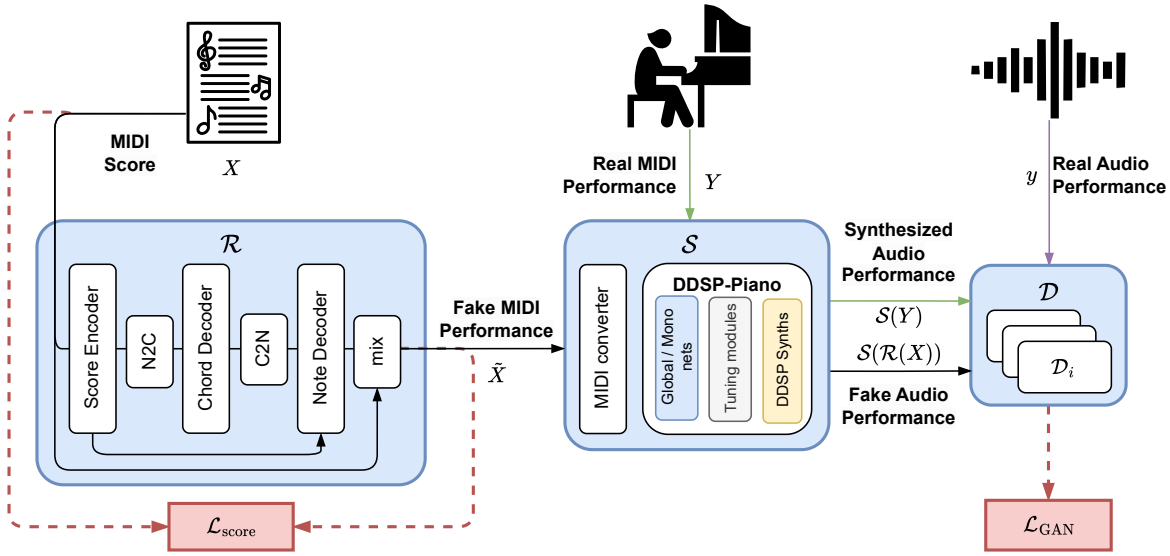


Figure 5.3: Cross-modal extension of the unpaired performance rendering model. The multi-scale audio discriminator is fed with real audio performances y, real symbolic performances that were synthesized into audio $\mathcal{S}(Y)$, and fake performances $\mathcal{S}(\mathcal{R}(X))$ synthesized by the performance rendering model $\mathcal{R}$ and DDSP-Piano $\mathcal{S}$ from scores X. Note that all three inputs are **unaligned**. The pre-trained differentiable DDSP-Piano synthesizer is frozen during the training of the performance rendering model: it allows to transmit the gradient from the audio modality back to the symbolic modality.

5.3 Cross-Modal Extension

As stated in the motivations Section 5.1, the three usual requirements for training a performance rendering model are 1) to get performances in the symbolic modality, 2) to compare the performances with their original compositions and 3) to leverage music sheet-exclusive markings. The model presented during the previous Section 5.2 sets aside the last two requirements by using adversarial training to transfer simple MIDI scores into MIDI performances.

In this section, a cross-modal extension of the method is proposed to remove the first requirement to an extent, by learning expressive features from audio performances instead of symbolic performances. This idea was already proposed in the work of Wang and Yang (2019) that directly synthesizes audio performances from score inputs. However, its controllability is limited since the obtained performances entangle the instrument timbre with the underlying controls, and the users cannot modify them independently afterwards.

Instead, the proposed extension outputs piano performances both as audio waveforms and as MIDI sequences, by inserting a frozen DDSP-Piano as a lightweight differentiable bridge between the two modalities. The overall model architecture is depicted in Figure 5.3.

Due to time constraints, this idea has not been fully developed and evaluated, as the results can still be improved, according to our informal listening.

5.3.1 From note-wise to frame-wise Conditioning

As mentioned in Section 2.1.3, different encodings of symbolic music are possible and need to be chosen efficiently for the task at hand. For piano performance rendering, Section 3.3 showed that encoding performances as sequences of notes is preferred, since they efficiently embed the expressive modifications made to the score. On the other hand, while multiple encodings of MIDI have been used for audio synthesis (Section 3.2), frame-wise encodings are the most commonly used in the literature (the flagship being the piano rolls), since their temporal relation with audio is a simple upsampling from the frame rate to the sample rate.

In the continuity of these works, our proposed performance rendering model encodes performances as note sequences, while DDSP-Piano takes frame-wise performance conditioning inputs. Converting a MIDI sequence from a note-wise to a frame-wise encoding necessitates finding in which *quantized* time frames each note is present, which calls for a rounding operation. While the conversion is easily feasible offline, the non-differentiability of the rounding operation prevents the seamless concatenation of our performance rendering and audio synthesis models.

The problem is similar to the duration modeling task in TTS, where sequences of words have to be adapted to the frame-wise representation of the acoustic model output (often the mel-spectrogram). One can think of adapting their proposed methods (Elias et al., 2021; Nguyen et al., 2023), but they assume that the words do not overlap since the voice is monophonic. However, in our case of polyphonic piano performances, notes can clearly overlap with each other.

Instead, we take inspiration from Mikkonen et al. (2023) that implemented a differentiable delay line for magnetic tape modeling. Notably, they need to predict a time-varying number of delayed samples between the recording and playback heads of the tape recorder. To this end, they allow an input signal sample to "bleed" between two output samples when the delay value is non-integer. This is done by converting the sequence of delay values into an alignment matrix between the input and output time ranges: each delay value is transformed into a "one-hot"-like vector where a value of 1 is located at the corresponding output index and the rest of filled with zeros. The differentiability arises through the fact that non-integer values are distributed between the two covered bins: for example, a value of 1.2 is represented by a vector with values 0.8 at index 1, 0.2 at index 2, and 0 elsewhere. The delayed output signal is obtained by matrix multiplication between the input signal and the alignment matrix.

This **soft one-hot** encoding $\text{sOH}_D(x) \in [0,1]^D$of a value $x \in \mathbb{R}$ along a range of depth $D \in \mathbb{N}^*$ can be expressed as:

$$\text{sOH}_D(x) \coloneqq \max(0, 1 - |\text{range}(D) - x|), \tag{5.7}$$

with $\text{range}(D) \coloneqq [0, 1, ..., D-1]$ the range vector of length D.

Similarly, we define a **soft ceiling** encoding $\text{sCeil}_D(x) \in [0,1]^D$ for the same value x and depth D, which gives 1 for indices lower than x and 0 for those higher, with the same smoothing behavior as sOH:

$$\text{sCeil}_D(x) \coloneqq \min(1, \max(0, x - \text{range}(D))). \tag{5.8}$$

Data: Notes boundaries (onsets, offsets) $\{\mathrm{ON}_n, \mathrm{OFF}_n\}_{n \leq N} \in \mathbb{R}^{N \times 2}$
Result: Voice assignments $\{q_n\}_{n \leq N} \in [\![0, P[\![^{N \times 1}$
State $\leftarrow [0, 0, ..., 0]$; `/* of size` P `*/`
for $n \leftarrow 0$ **to** N **do**
  idx $\leftarrow \arg\min(\text{State})$; `/* get the next available voice */`
  if $\mathrm{ON}_n \geq \text{State(idx)}$ **then**
    $q_n \leftarrow$ idx ;
    State[idx] $\leftarrow \mathrm{OFF}_n$;
  else
    $q_n \leftarrow 0$; `/* the note is not assigned to any voice */`
  end
end

Algorithm 1: Assignment of the ordered MIDI notes among the P non-overlapping voices of the DDSP-Piano conditioning input. It can be implemented as a RNN cell.

To convert a note sequence into the frame-wise conditioning of DDSP-Piano (Equation 4.2), we first need to assign each note into one of P parallel voices, such that notes in each voice do not overlap. Algorithm 1 gives a pseudo-code for getting the sequence of voice assignments[3] $\{q_n\}_{n \leq N}$ given the notes onsets $\mathrm{ON}_n \coloneqq \sum_{n=1}^{N} o_n \times F_{\text{frame}}$ and offsets $\mathrm{OFF}_n \coloneqq \mathrm{ON}_n + d_n \times F_{\text{frame}}$.

Then, given a maximum length T of the frame-wise conditioning, the active pitch component $x^{\text{pitch}} \in \mathbb{R}^{T \times P}$ can be obtained through:

$$x^{\text{pitch}} = [p_n \times (\mathrm{sCeil}_T(\mathrm{OFF}_n) - \mathrm{sCeil}_T(\mathrm{ON}_n))]_{n \leq N}^{T} [\mathrm{sOH}_P(q_n)]_{n \leq N}, \tag{5.9}$$

and the onset velocity component $x^{\text{vel}} \in \mathbb{R}^{T \times P}$ through:

$$x^{\text{vel}} = [v_n \times \mathrm{sOH}_T(\mathrm{ON}_n)]_{n \leq N}^{T} [\mathrm{sOH}_P(q_n)]_{n \leq N}. \tag{5.10}$$

These two formulas allow for a differentiable conversion of a note-wise MIDI sequence into the frame-wise representation of DDSP-Piano. Note that a regular piano roll can be obtained from the above operations by ignoring the voice assignment step. The notes lost during the conversion are those that would exceed the polyphonic capacity of DDSP-Piano (similarly to analog and digital synthesizers) and those that are played after the maximum segment duration supported during training.

5.3.2 Training Setup

The adversarial training of our cross-modal performance rendering model is also extended from the symbolic-only baseline presented previously in Section 5.2.2. The symbolic performance discriminator is replaced by an audio performance discriminator, with also a multi-scaling strategy. Notably, real symbolic performances are also included in the

[3] In this chapter, a monophonic voice is notated q instead of p in the previous chapters, to distinguish it from a single note pitch.

training to prevent the audio discriminator from identifying fake performances solely on the piano timbre of DDSP-Piano, which can differ from the piano used in the real audio recordings. The LS-GAN objectives are thus:

$$\begin{aligned}\mathcal{L}_{\mathcal{D}_i} = &\mathop{\mathbb{E}}_{y\sim\mathcal{P}_{\text{perf}}}\left[\frac{1}{2}\|\mathcal{D}_i(y)-1\|_2\right] + \mathop{\mathbb{E}}_{Y\sim\mathcal{P}_{\text{perf}}}\left[\frac{1}{2}\|\mathcal{D}_i(\mathcal{S}(Y))-1\|_2\right] \\ &+ \mathop{\mathbb{E}}_{X\sim\mathcal{P}_{\text{score}}}\left[\|\mathcal{D}_i(\mathcal{S}(\mathcal{R}(X)))\|_2\right], \\ \mathcal{L}_{\mathcal{R}} = &\mathop{\mathbb{E}}_{X\sim\mathcal{P}_{\text{score}}}\left[\lambda_{\text{GAN}}\sum_{i=1}^{N_{\mathcal{D}}}\|\mathcal{D}_i(\mathcal{S}(\mathcal{R}(X)))-1\|_2 + \mathcal{L}_{\text{score}}(X)\right].\end{aligned} \tag{5.11}$$

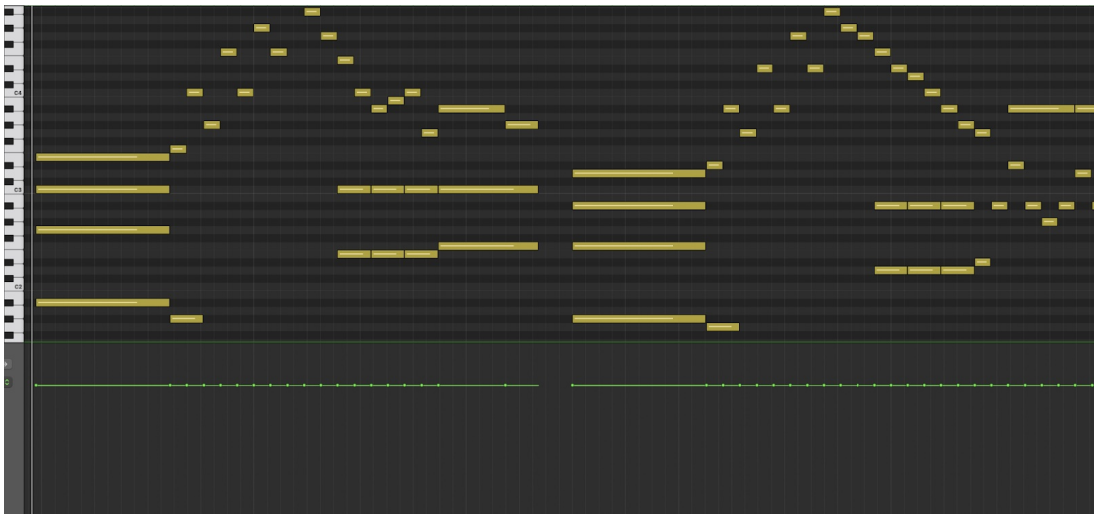
Namely, the CIPI dataset is selected as a representative of the input score distribution $\mathcal{P}_{\text{score}}$. Following the first train-validation-split proposed by Ramoneda et al. (2024), the compositions are still encoded as sequences of notes without score markings. ASAP provides the real MIDI performances $Y \sim \mathcal{P}_{\text{perf}}$ following the train-validation-test split from Liu et al. (2021). Finally, the ATEPP dataset represents the distribution of audio performances $y \sim \mathcal{P}_{\text{perf}}$. Since no train/validation/test split was provided, we created our own split, stratifying along the composers (72% train, 8% validation, and 20% test). Ideally, the split should follow the pieces split from ASAP, but matching the classical piece titles is non-trivial (Zhang et al., 2022) and left for future works.

The symbolic scores and performances are segmented into sequences of 128 notes, while the real audio performances are split into $10s$ chunks at $F_{\text{audio}} = 8\text{kHz}$. A high audio sampling rate is not necessary as the piano timbre should not be considered by the discriminators and note displacements during performances happen on larger time scales. A smaller variant of DDSP-Piano at 8kHz is trained on the ENSTDkCl piano model from MAPS, for which the default tuning parameters correspond.

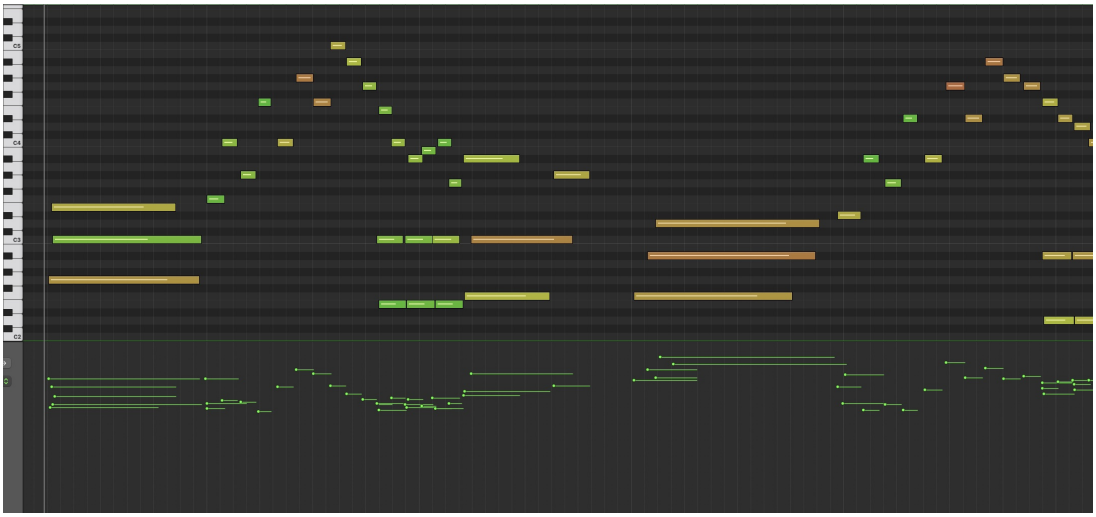
According to the dataset comparison in Table 3.1, the size of ATEPP is an order of magnitude larger than datasets with recorded symbolic performances (ASAP and MAESTRO). Other performance rendering works using this dataset need the symbolic transcription through a large AMT model. They also require either a performance-to-score transcription model (Tang et al., 2023) or rely on the multitude of performances from the same pieces (Zhang and Dixon, 2023). The presented approach leverages instead a lightweight audio synthesizer, but also an audio discriminator. While the auxiliary models required by these three approaches may be of similar sizes, our adversarial approach does not rely on multiple performances per composition. To evaluate this property, we would need to compare our cross-modal method to the model of Zhang and Dixon (2023) on a subset of ATEPP where a single performance is kept per composition. Instead, the comparison between using a side NAS model or an AMT model can be made by training the symbolic-only variant of our approach on the provided transcriptions of ATEPP as an ablation study.

5.3.3 Qualitative Analysis

Through informal listening, we found that the expressive quality of the performances rendered by the cross-modal approach is not sufficient as of writing. Audio examples are



(a) Deadpan/unexpressive rendition of the composition.



(b) Expressive rendition by the proposed cross-modal variant.

Figure 5.4: Excerpt of Beethoven's 8th Sonata, 3rd Movement (Figure (a)) rendered into a performance (Figure (b)) by the proposed cross-modal model. Each symbolic excerpt is visualized by an active velocity piano roll (top) and a velocity plot (below), on the Logic Pro X software.

publicly available and the readers are invited to them on the accompanying website[4].

For instance, one can listen to the rendition of the second part of Beethoven's 8th Sonata, 3rd movement (also illustrated in Figure 5.4). In this example, the model has reproduced several playing techniques that are found in realistic performances, namely:

- Nuances and loudness variations: despite a constant velocity from the composition input, the rendered performance exhibits different velocity values for each note. This

[4] http://renault.gitlab-pages.ircam.fr/thesis-support/chap_5-3

behavior is realistic as human pianists cannot play successive notes at exactly the same velocity. Also, accompanying an ascending (respectively descending) melody line with an increase (respectively decrease) in loudness is also an artistic choice commonly made by interpreters. Yet, voicings are not well reproduced by the model as notes in the melody line do not have louder velocities than the accompanying chords.

- Simultaneous notes spreading: upon chord onsets, the individual notes are not played exactly at the same time, as a real pianist would do. Particularly in the second half of the excerpt, the Bb/F chord is played in an arpeggiated fashion that reinforces a "dramatic" effect.

- Tempo variations: changes in tempo can be heard throughout the rendered performance. However, it has been analyzed that realistic tempo curves often fit to the structure of the piece (Widmer and Tobudic, 2003), which is not the case in the performance rendered by the model as there are too many tempo changes per musical phrase. Therefore, because of this expressive effect being wrongfully applied, the full performance can be perceived as unrealistic (Cancino-Chacón et al., 2018).

As such, since the quality of the generated performances is not satisfying, and given that perceptual evaluation is time and resource-consuming, the perceptual evaluation is left for future work after improvements of the method.

Section summary - Cross-modal Extension

This section extends the previous unsupervised piano performance rendering model to also learn expressive qualities from audio recordings. To this end, DDSP-Piano is inserted as a differentiable bridge between the symbolic and audio modalities, and the symbolic discriminator is changed into an audio performance discriminator. The concatenation of the performance rendering model with the audio synthesizer requires the conversion of the MIDI performances from a note-wise to a frame-wise encoding (which are suited for their respective tasks). To the extent of our knowledge, differentiable conversion between the different symbolic music encodings has not been tackled in the literature: we propose a learning-free method to translate note sequences into piano roll-like encodings. Consequently, the full cross-modal approach can be trained using unpaired symbolic scores and audio performances from the larger ATEPP dataset. However, some amount of symbolic performances are still needed to prevent the discriminator from overfitting on the instrument timbre. As of writing, the model quality can still be improved and evaluations are left for future works.

5.4 Discussion

As suggested by the subjective evaluation, the proposed performance rendering model in the symbolic domain lacks an understanding of the musical content of a score and can apply inappropriate performance features. This could be expected as high-level features of the score, such as tempo, key, and time signatures, are not as explicit as using score markings, like in previous works (Jeong et al., 2019b; Rhyu et al., 2022; Maezawa et al., 2019). Choi et al. (2020) evaluate their performance generation model extracting statistics, such as note densities, mean note durations and velocities, and pitch range. Extracting and providing such statistics from the score may help our performance rendering model in understanding the structure and style of the score without labeling or markings. Moreover, transformer-based models have been popular for modeling both symbolic compositions (Huang et al., 2019; Dong et al., 2023) and symbolic performances (Borovik and Viro, 2023; Tang et al., 2023) as they can more efficiently model long-term dependencies between sequence elements than RNN and CNN. Such architecture could benefit our approach, especially with the increased amount of usable data offered by the unsupervised training setting.

Another limitation of both variants of our performance rendering model is its lack of controllability. While the cross-modal variant still lets the users access the symbolic modality of the rendered performances, no high-level controls are available for guiding the rendering process. Organizing the real performances into sub-domains (by providing composer and/or performer labels for example) and employing an auxiliary classifier during training (Odena et al., 2017) can provide a style control over the rendering. However, this would diminish the purpose of the current unsupervised adversarial training that can be fed with unlabeled real performances. Instead, self-supervised strategies can be considered by training the rendering model to also produce performances that match extractable features, such as the symbolic statistics mentioned previously, or the mid-level perceptual features from Chowdhury and Widmer (2021) that quantify expressive qualities in audio recordings.

Moreover, the present work only focuses on classical piano music to be comparable with previous supervised approaches. Yet, since it does not rely on training score-performance pairs, it can be used for rendering performances for other music genres and instruments, providing symbolic or audio datasets.

Finally, while the cross-modal variant alleviates the need to transcribe performances from audio to MIDI, it instead uses an audio synthesizer and a discriminator to handle the two modalities. As such, both methodologies are entailed to the quality of their auxiliary model (for AMT or NAS). Also, while we leveraged a pre-trained DDSP-Piano model, other learning-free but differentiable methods can be used instead, such as differentiable sampling-based synthesis (Sumino et al., 2020).

5.5 Performance Rendering with Unpaired Data, in short

Chapter summary - Performance Rendering with Unpaired Data

This chapter presented a new approach to piano performance rendering by addressing three training requirements systematically found in the literature: 1) gathering performances in the symbolic modality, 2) comparing the performances with their original compositions and 3) relying on music sheet-exclusive markings. The first two points call for multiple pre-processing steps in order to gather adequate data, while the third point limits the scope of application to music written as scores. Our approach first circumvent the last two points by viewing performance rendering as a domain transfer task from unexpressive score music to expressive interpretations. To this end, scores and performances are encoded in the same manner as sequences of notes with minimal features, and an unsupervised adversarial training is employed to learn expressive qualities without providing score-performance pairs. Through a listening test, the approach demonstrates better expressive qualities than the direct rendition of the scores, but not with the same amount of naturalness than performances made by a supervised baseline and by humans. Nonetheless, a cross-modal extension of the approach is proposed to also learn expressive features from audio recordings, by including the DDSP-Piano synthesizer as a differentiable bridge between the symbolic and audio modalities. The inclusion of DDSP-Piano in the training pipeline requires a differentiable conversion of performances from a note-wise encoding to a frame-wise encoding, which has been developed. Hereafter, datasets of audio performances can be leveraged to train the performance rendering model, which are larger than previous score-performance aligned symbolic datasets and open up the possible usage of more powerful deep learning methods.

Chapter 6

Conclusion

In the context of music creation, the plethora of agents, techniques, and knowledge that can be involved makes for a rich and diverse artistic endeavor. However, gaining access to such expertise can be a time-consuming and/or resource-intensive endeavor, and the application of these skills often involves undertaking non-creative but necessary tasks that hinder the creative process. Therefore, numerous works and products aim to alleviate the technical workload and automate portions of the musical creation process, with recent Deep Learning (DL) approaches pushing the balance to a point where music practitioners may feel excluded from the decision-making.

In this thesis, we explored the design of Deep Neural Network (DNN)s for seamless integration into the music generation process while providing steerable results. They were applied for the tasks of audio synthesis and expressive performance rendering, in the specific case of piano music.

In order to outline the technical principles for manipulating piano music with DNNs, Chapter 2 introduced the related scientific background. It notably presented the symbolic encoding of music with the MIDI protocol, in the forms of note sequences or piano roll images, and its manipulation as waveforms or spectrograms with audio signal processing. Then, the piano was briefly described, with an emphasis on the inharmonicity of its strings and its particular tuning as a polyphonic instrument. Afterward, the principles behind DL were explained, with its ability to learn complex and non-linear relationships from data observations, thanks to the principles of differentiable operations and gradient back-propagation. The chapter concluded with insights on the inclusion of domain knowledge to support the learning process of DNNs.

The thesis work is built upon the knowledge acquired from the literature of audio synthesis and performance rendering, reported in Chapter 3. Several large and high-quality datasets of piano performances were conceived, both in the symbolic and audio modalities and eventually both. They allowed the training of DNN-based models for synthesizing piano audio from MIDI inputs, implicitly modeling the instrument from these input-output observation pairs. Yet, piano modeling was already tackled by physical-based, signal-based, and sampling-based strategies, requiring various depths of explicit modeling and parameter estimation. In a hybrid manner, the Differentiable Digital Signal Processing (DDSP) approach combines signal processing and synthesis tools into DNNs,

combining the expressivity of DL with the a priori knowledge of signal processing. As for performance rendering, DNNs have also been employed to make expressive renditions of musical scores, learning from note-aligned composition-performance pairs. Still, the variety of plausible performances for the same score raises challenges in both, the modeling and evaluation stages.

Neural Audio Synthesis informed by Instrument Knowledge. Chapter 4 presented the main contribution of the thesis for the task of piano audio synthesis: DDSP-Piano, a differentiable hybrid synthesizer expanding on the DDSP framework for handling polyphonic MIDI inputs and audio output. In tandem with the sound structure priors inherent to the differentiable spectral modeling components of DDSP, the model further incorporates high-level knowledge of the instrument to specifically tackle particularities of the piano sound. Namely, the explicit tuning and inharmonicity sub-modules contribute to the quality of the lightweight and interpretable model, which surpasses a pure neural benchmark according to a conducted listening test. After quantitative and qualitative evaluation revealing shortcomings in some module training and the overall disentanglement of sound components, a second iteration of the model was proposed targeting these limitations. The conducted work further explored the hybridization of DL with domain knowledge in the context of neural audio synthesis, obtaining an efficient and interpretable model that can be manipulated and has the potential to be implemented for real-world usages.

Unsupervised Performance Rendering in a Low-Informed Setting. A novel approach to performance rendering was proposed in Chapter 5 to circumvent the usual training needs for gathering symbolic performances aligned with compositions with score markings. By viewing performance rendering as a domain transfer task between unexpressive MIDI music and expressive interpretations, the proposal employs an adversarial training that avoids the need to provide score-performance pairs or score markings. Thanks to the chosen encoding, the model can make expressive renditions out of symbolic music edited as in modern music production software, which has the potential to offer more realistic feedback to composers. Still, a listening test has revealed that the expressive qualities delivered by the approach needs to be improved in order to reach those from a supervised benchmark and from real musicians. Expanding on the unsupervised training setting, a cross-modal extension of the approach was proposed to also learn expressive qualities from audio recordings. Following the development of a differentiable converter between the representations of performances by the rendering model and DDSP-Piano, the latter was included as a differentiable bridge between the symbolic and audio modalities. This enables the usage of existing large datasets of audio performances for training the symbolic performance rendering model, without fully transcribing them into MIDI. Although the full experiments of this extended approach are still pending completion, the design would allow for expressive audio renditions of compositions while still letting the user manipulate and refine the intermediary MIDI performance.

Future works. Future technical work for each of the models presented have been explained in greater detail in their respective chapters. Namely, DDSP-Piano calls for a more flexible and robust frequency estimation method and for further exploration of the DL and instrument modeling literature in order to achieve suitable disentanglement

of the various factors contributing to the piano sound. As for the unsupervised performance rendering model, the main improvements are expected to center on enhancing the high-level understanding of music compositions and the emotional characteristics of performances, without relying on score markings but rather on extracted features with hand-crafted operations and/or with auxiliary models. Hence, self-supervised learning seems suitable for improving and building around the work developed during this thesis. For performance rendering, steering the model training to match extracted high-level emotional features would both increase the discriminator interpretability and the model controllability for better interaction with the user. As for the piano audio synthesizer, in a similar fashion to other DDSP works, it can serve as a data generator and augmentation pipeline (by modifying specific control parameters) for training source separation models and audio-to-MIDI transcription model. In turn, such a transcription model could be used for better estimation of the piano frequencies in a polyphonic context and would also allow the modeling of piano instruments other than Disklaviers. Naturally, extending the approaches to other instruments and music styles is conceivable: other polyphonic instruments could be modeled with DDSP leveraging the polyphonic conditioning developed for DDSP-Piano, while the cross-modal variant of the performance rendering model can motivate the modeling of non-classical piano performances and eventually, to other MIDI-controllable instruments. Notably, the developed differentiable MIDI converter opens up the combination of multiple models dealing with symbolic music in order to achieve complex and coherent modeling over multiple sub-tasks, while maintaining interpretability and controllability thanks to the modular approach.

Bibliography

Agostinelli, A., Denk, T. I., Borsos, Z., Engel, J., Verzetti, M., Caillon, A., Huang, Q., Jansen, A., Roberts, A., Tagliasacchi, M., Sharifi, M., Zeghidour, N., and Frank, C. (2023). Musiclm: Generating music from text. arXiv preprint arXiv:2301.11325.

Apel, W. (1997). *The history of keyboard music to 1700*, chapter 2. Indiana University Press.

Askenfelt, A. and Jansson, E. V. (1990). From touch to string vibrations. I: Timing in the grand piano action. *The Journal of the Acoustical Society of America*, 88(1):52–63.

Askenfelt, A. and Jansson, E. V. (1993). From touch to string vibrations. iii: String motion and spectra. *The Journal of the Acoustical Society of America*, 93(4):2181–2196.

Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. arXiv preprint arXiv:1607.06450.

Badeau, R. (2005). *Méthodes à haute résolution pour l'estimation et le suivi de sinusoïdes modulées. Application aux signaux de musique*. PhD thesis, Télécom ParisTech.

Bank, B. and Chabassier, J. (2019). Model-based digital pianos: From physics to sound synthesis. *IEEE Signal Processing Magazine*, 36(1):103–114.

Bank, B., Zambon, S., and Fontana, F. (2010). A modal-based real-time piano synthesizer. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(4):809–821.

Barahona-Ríos, A. and Collins, T. (2024). Noisebandnet: Controllable time-varying neural synthesis of sound effects using filterbanks. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 32:1573–1585.

Bazin, T. (2023). *Designing Novel Time-Frequency Scales for Interactive Music Creation with Hierarchical Statistical Modeling*. Theses, Sorbonne Université.

Bengio, Y., Frasconi, P., and Simard, P. (1993). The problem of learning long-term dependencies in recurrent networks. In *Proceedings of the IEEE International Conference on Neural Networks*, volume 3, pages 1183–1188.

Berendes, H.-U., Schwär, S., Schäfer, M., and Müller, M. (2023). Towards differentiable piano synthesis based on physical modeling. In *Late-Breaking Demos of the International Society for Music Information Retrieval Conference (ISMIR)*, Milano, Italy.

Bilbao, S. (2009). *Numerical Sound Synthesis: Finite Difference Schemes and Simulation in Musical Acoustics*. John Wiley & Sons.

Bond-Taylor, S., Leach, A., Long, Y., and Willcocks, C. G. (2022). Deep generative modelling: A comparative review of VAEs, GANs, Normalizing Flows, Energy-Based and Autoregressive models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(11):7327–7347.

Borovik, I. and Viro, V. (2023). Scoreperformer: Expressive piano performance rendering with fine-grained control. In *Proceedings of the International Society of Music Information Retrieval Conference (ISMIR)*.

Boutillon, X. and Ege, K. (2013). Vibroacoustics of the piano soundboard: Reduced models, mobility synthesis, and acoustical radiation regime. *Journal of Sound and Vibration*, 332(18):4261–4279.

Brunner, G., Wang, Y., Wattenhofer, R., and Zhao, S. (2018). Symbolic music genre transfer with CycleGAN. In *International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 786–793. IEEE.

Caillon, A. and Esling, P. (2021). RAVE: A variational autoencoder for fast and high-quality neural audio synthesis. arXiv preprint arXiv:2111.05011.

Caillon, A. and Esling, P. (2022). Streamable neural audio synthesis with non-causal convolutions. In *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, pages 320–327, Vienna, Austria.

Cancino-Chacón, C. E., Grachten, M., Goebl, W., and Widmer, G. (2018). Computational models of expressive music performance: A comprehensive and critical review. *Frontiers in Digital Humanities*, 5.

Carson, A., Valentini-Botinhao, C., King, S., and Bilbao, S. (2023). Differentiable grey-box modelling of phaser effects using frame-based spectral processing. In *Proceedings of the International Conference on Digital Audio Effects (DAFx)*.

Caspe, F., McPherson, A., and Sandler, M. (2022). DDX7: Differentiable FM Synthesis of Musical Instrument Sounds. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 608–616, Bengaluru, India.

Castellon, R., Donahue, C., and Liang, P. (2020). Towards realistic MIDI instrument synthesizers. In *Proceedings of the NeurIPS Workshop on Machine Learning for Creativity and Design*.

Cazelles, E., Robert, A., and Tobar, F. (2021). The wasserstein-fourier distance for stationary time series. *IEEE Transactions on Signal Processing*, 69:709–721.

Chabassier, J. (2012). *Modélisation et Simulation Numérique d'un Piano par Modèles Physiques*. PhD thesis, École Polytechnique X, Palaiseau, France.

Chabassier, J., Chaigne, A., and Joly, P. (2013). Modeling and simulation of a grand piano. *Journal of the Acoustical Society of America*, 134(1):648–665.

Chacón, C. E. C. and Grachten, M. (2016). The basis mixer: a computational romantic pianist. In *Late-Breaking Demos of the 17th International Society for Music Information Retrieval Conference (ISMIR)*.

Chen, N., Zhang, Y., Zen, H., Weiss, R. J., Norouzi, M., and Chan, W. (2021). Wavegrad: Estimating gradients for waveform generation. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Choi, K., Hawthorne, C., Simon, I., Dinculescu, M., and Engel, J. (2020). Encoding musical style with transformer autoencoders. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1899–1908.

Chowdhury, J. (2021). Rtneural: Fast neural inferencing for real-time systems. arXiv preprint arXiv:2106.03037.

Chowdhury, S. and Widmer, G. (2021). Towards explaining expressive qualities in piano recordings: Transfer of explanatory features via acoustic domain adaptation. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 561–565. IEEE.

Colonel, J. T., Comunità, M., and Reiss, J. (2022). Reverse engineering memoryless distortion effects with differentiable waveshapers. In *Audio Engineering Society Convention 153*. Audio Engineering Society.

Colonel, J. T. and Reiss, J. (2021). Reverse engineering of a recording mix with differentiable digital signal processing. *The Journal of the Acoustical Society of America*, 150(1):608–619.

Conklin, Harold A., J. (1996). Design and tone in the mechanoacoustic piano. part ii. piano structure. *The Journal of the Acoustical Society of America*, 100(2):695–708.

Conklin, Harold A., J. (1999). Generation of partials due to nonlinear mixing in a stringed instrument. *The Journal of the Acoustical Society of America*, 105(1):536–545.

Cooper, E., Wang, X., and Yamagishi, J. (2021). Text-to-Speech Synthesis Techniques for MIDI-to-Audio Synthesis. In *Proceedings of the ISCA Speech Synthesis Workshop (SSW 11)*, pages 130–135, Budapest, Hungary.

Copet, J., Kreuk, F., Gat, I., Remez, T., Kant, D., Synnaeve, G., Adi, Y., and Défossez, A. (2023). Simple and controllable music generation. In *Proceedings of the Conference on Neural Information Processing Systems (NIPS)*.

Dash, T., Chitlangia, S., Ahuja, A., and Srinivasan, A. (2022). A review of some techniques for inclusion of domain-knowledge into deep neural networks. *Scientific Reports*, 12(1):1040.

De Man, B., Reiss, J., and Stables, R. (2017). Ten years of automatic mixing. In *Proceedings of the 3rd Workshop on Intelligent Music Production*, Salford, UK.

Défossez, A., Copet, J., Synnaeve, G., and Adi, Y. (2023). High fidelity neural audio compression. *Transactions on Machine Learning Research*. Featured Certification, Reproducibility Certification.

Défossez, A., Zeghidour, N., Usunier, N., Bottou, L., and Bach, F. (2018). SING: Symbol-to-instrument neural generator. In *Proceedings of the International Conference on Neural Information Processing Systems (NIPS)*, page 9055–9065, Montréal, Canada. Curran Associates Inc.

Dhariwal, P., Jun, H., Payne, C., Kim, J. W., Radford, A., and Sutskever, I. (2020). Jukebox: A generative model for music. arXiv preprint arXiv:2005.00341. Accessed on 18/03/2024.

Diaz, R., Hayes, B., Saitis, C., Fazekas, G., and Sandler, M. (2023). Rigid-body sound synthesis with differentiable modal resonators. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE.

Donahue, C., McAuley, J., and Puckette, M. (2019). Adversarial audio synthesis. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Dong, H.-W., Chen, K., Dubnov, S., McAuley, J., and Berg-Kirkpatrick, T. (2023). Multitrack music transformer. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Dong, H.-W., Zhou, C., Berg-Kirkpatrick, T., and McAuley, J. (2022). Deep performer: Score-to-audio music performance synthesis. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 951–955, Singapore, Singapore. IEEE.

Douwes, C. (2023). *On the Environmental Impact of Deep Generative Models for Audio*. Theses, Sorbonne Université.

Dudley, H. (1939). Remaking speech. *The Journal of the Acoustical Society of America*, 11(2):169–177.

Elias, I., Zen, H., Shen, J., Zhang, Y., Jia, Y., Skerry-Ryan, R., and Wu, Y. (2021). Parallel Tacotron 2: A Non-Autoregressive Neural TTS Model with Differentiable Duration Modeling. In *Proceedings of Interspeech 2021*, pages 141–145.

Emiya, V., Bertin, N., David, B., and Badeau, R. (2010). MAPS - a Piano Database for Multipitch Estimation and Automatic Transcription of Music. Research report, INRIA.

Engel, J., Agrawal, K. K., Chen, S., Gulrajani, I., Donahue, C., and Roberts, A. (2019). Gansynth: Adversarial neural audio synthesis. In *Proceedings of the International Conference on Learning Representations (ICLR)*, New Orleans, LA, USA.

Engel, J., Hantrakul, L. H., Gu, C., and Roberts, A. (2020a). DDSP: Differentiable digital signal processing. In *Proceedings of the International Conference on Learning Representations (ICLR)*, Addis Abada, Ethiopia.

Engel, J., Resnick, C., Roberts, A., Dieleman, S., Norouzi, M., Eck, D., and Simonyan, K. (2017). Neural audio synthesis of musical notes with wavenet autoencoders. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1068–1077, Sydney, NSW, Australia. PMLR, JMLR.org.

Engel, J., Swavely, R., Hantrakul, L. H., Roberts, A., and Hawthorne, C. (2020b). Self-supervised pitch detection by inverse audio synthesis. In *ICML Workshop on Self-supervision in Audio and Speech*, Online.

Esling, P., Chemla-Romeu-Santos, A., and Bitton, A. (2018). Bridging audio analysis, perception and synthesis with perceptually-regularized variational timbre spaces. In *Proceedings of the International Society for Music Information Retrieval (ISMIR)*, pages 175–181, Paris, France.

Esling, P., Masuda, N., Bardet, A., Despres, R., and Chemla-Romeu-Santos, A. (2020). Flow synthesizer: Universal audio synthesizer control with normalizing flows. *Applied Sciences*, 10.

Esqueda, F., Kuznetsov, B., and Parker, J. D. (2021). Differentiable white-box virtual analog modeling. In *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, pages 41–48.

Fletcher, H. (1964). Normal vibration frequencies of a stiff piano string. *Journal of the Acoustical Society of America*, 36:203–209.

Flossmann, S., Grachten, M., and Widmer, G. (2013). Expressive performance rendering with probabilistic models. *Guide to Computing for Expressive Music Performance*, pages 75–98.

Foscarin, F., Karystinaios, E., Peter, S. D., Cancino-Chacón, C., Grachten, M., and Widmer, G. (2022). The match file format: Encoding alignments between scores and performances. In *Proceedings of the Music Encoding Conference*, Halifax, Canada.

Foscarin, F., Mcleod, A., Rigaux, P., Jacquemard, F., and Sakai, M. (2020). ASAP: a dataset of aligned scores and performances for piano transcription. In *Proceedings of the International Society for Music Information Retrieval (ISMIR)*, Montreal / Virtual, Canada.

French, N. R. and Steinberg, J. C. (1947). Factors governing the intelligibility of speech sounds. *Journal of the Acoustical Society of America*, 19(1):90–119.

Friberg, A., Bresin, R., and Sundberg, J. (2006). Overview of the kth rule system for musical performance. *Advances in cognitive psychology*, 2(2):145.

Frid, E., Gomes, C., and Jin, Z. (2020). Music creation by example. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, page 1–13, New York, NY, USA. Association for Computing Machinery.

Gardner, J. P., Simon, I., Manilow, E., Hawthorne, C., and Engel, J. (2022). MT3: Multi-task multitrack music transcription. In *Proceediings of the International Conference on Learning Representations (ICLR)*.

Good, M. (2001). Musicxml: an internet-friendly format for sheet music. In *Proceedigs of XML*, Boston, MA.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., and Weinberger, K., editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

Gottlieb, D. and Shu, C.-W. (1997). On the gibbs phenomenon and its resolution. *SIAM Review*, 39(4):644–668.

Grasser, C. (1995). Le piano romantique français de 1823 à 1867. In *Le Pianoforte en France: ses Descendants jusqu'aux Années Trente*. Paris Bibliothèques.

Griffin, D. and Lim, J. (1984). Signal estimation from modified short-time fourier transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 32(2):236–243.

Grindlay, G. and Helmbold, D. (2006). Modeling, analyzing, and synthesizing expressive piano performance with graphical models. *Machine Learning*, 65(2):361–387.

Grumiaux, P.-A. and Lagrange, M. (2023). Efficient bandwidth extension of musical signals using a differentiable harmonic plus noise model. *EURASIP Journal on Audio, Speech, and Music Processing*, 2023(1):51.

Gu, Y. and Raphael, C. (2012). Modeling piano interpretation using switching kalman filter. In *Proceedings of the International Society of Music Information Retrieval (ISMIR)*, pages 145–150.

Hahn, H. and Roebel, A. (2013). Joint F0 and Inharmoncity Estimation using Second Order Optimization. In *SMC Sound and Music Computing Conference 2013*, pages 695–700, Stockholm, Sweden.

Han, H., Lostanlen, V., and Lagrange, M. (2023). Perceptual–neural–physical sound matching. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Harris, F. (1978). On the use of windows for harmonic analysis with the discrete fourier transform. *Proceedings of the IEEE*, 66(1):51–83.

Hashida, M., Nakamura, E., and Katayose, H. (2018). Crest-musepedb 2nd edition: Music performance database with phrase information. In *Proceedings of the 15th Sound and Music Computing (SMC) Conference*, Limassol, Cyrpus.

Hashida, M., Nakra, T., Katayose, H., Murao, T., Hirata, K., Suzuki, K., Kitahara, T., et al. (2008). Rencon: Performance rendering contest for automated music systems. In *Proceedings of the International Conference on Music Perception and Cognition (ICMPC)*, volume 5, Sapporo, Japan.

Hawthorne, C., Elsen, E., Song, J., Roberts, A., Simon, I., Raffel, C., Engel, J., Oore, S., and Eck, D. (2018). Onsets and frames: Dual-objective piano transcription. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 50–57, Paris, France.

Hawthorne, C., Simon, I., Roberts, A., Zeghidour, N., Gardner, J., Manilow, E., and Engel, J. (2022). Multi-instrument music synthesis with spectrogram diffusion. In *Proceedings of the International Society of Music Information Retrieval (ISMIR)*, pages 337–344, Bengaluru, India.

Hawthorne, C., Simon, I., Swavely, R., Manilow, E., and Engel, J. (2021). Sequence-to-sequence piano transcription with transformers. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 246–253, Online.

Hawthorne, C., Stasyuk, A., Roberts, A., Simon, I., Huang, C.-Z. A., Dieleman, S., Elsen, E., Engel, J., and Eck, D. (2019). Enabling factorized piano music modeling and generation with the MAESTRO dataset. In *Proceedings of the International Conference on Learning Representations (ICLR)*, New Orleans, Louisiana, USA.

Hayes, B., Saitis, C., and Fazekas, G. (2021). Neural waveshaping synthesis. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 254–261, Online.

Hayes, B., Saitis, C., and Fazekas, G. (2023). Sinusoidal frequency estimation by gradient descent. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Hayes, B., Shier, J., Fazekas, G., McPherson, A., and Saitis, C. (2024). A review of differentiable digital signal processing for music and speech synthesis. *Frontiers in Signal Processing*, 3.

Hernandez-Olivan, C. and Beltrán, J. R. (2023). Music composition with deep learning: A review. *Advances in Speech and Music Technology: Computational Aspects and Applications*, pages 25–50.

Hershey, S., Chaudhuri, S., Ellis, D. P. W., Gemmeke, J. F., Jansen, A., Moore, R. C., Plakal, M., Platt, D., Saurous, R. A., Seybold, B., Slaney, M., Weiss, R. J., and Wilson, K. (2017). Cnn architectures for large-scale audio classification. In *Proceedings of the*

IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 131–135.

Hoffmann, F. (2004). Midi (musical instrument digital interface). In *Encyclopedia of Recorded Sound*, pages 1372–1374. Routledge.

Hsiao, W.-Y., Liu, J.-Y., Yeh, Y.-C., and Yang, y.-h. (2021). Compound word transformer: Learning to compose full-song music over dynamic directed hypergraphs. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35:178–186.

Huang, C.-Z. A., Vaswani, A., Uszkoreit, J., Simon, I., Hawthorne, C., Shazeer, N., Dai, A. M., Hoffman, M. D., Dinculescu, M., and Eck, D. (2019). Music transformer. In *Prroceedings of the International Conference on Learning Representations (ICLR)*.

Huang, Y.-S. and Yang, Y.-H. (2020). Pop music transformer: Beat-based modeling and generation of expressive pop piano compositions. In *Proceedings of the 28th ACM International Conference on Multimedia*, MM '20, page 1180–1188, New York, NY, USA. Association for Computing Machinery.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 448–456. pmlr.

Jeong, D., Kwon, T., Kim, Y., Lee, K., and Nam, J. (2019a). VirtuosoNet: A Hierarchical RNN-based System for Modeling Expressive Piano Performance. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 908–915, Delft, The Netherlands.

Jeong, D., Kwon, T., Kim, Y., and Nam, J. (2019b). Graph neural network for music score data and modeling expressive piano performance. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 3060–3070, Long Beach, California, USA. PMLR.

Jeong, D., Kwon, T., Kim, Y., and Nam, J. (2019c). Score and performance features for rendering expressive music performances. In *Proceedings of the Music Encoding Conference*, Vienna, Austria.

Ji, S., Luo, J., and Yang, X. (2020). A comprehensive survey on deep music generation: Multi-level representations, algorithms, evaluations, and future directions. arXiv preprint arXiv:2011.06801.

Ji, S., Yang, X., and Luo, J. (2023). A survey on deep learning for symbolic music generation: Representations, algorithms, evaluations, and challenges. *ACM Computing Surveys*, 56(1).

Jonason, N., Sturm, B., and Thomé, C. (2020). The control-synthesis approach for making expressive and controllable neural music synthesizers. In *Proceedings of the Joint Conference on AI Music Creativity (AIMC)*, Stockholm, Sweden. AIMC.

Jonason, N., Wang, X., Cooper, E., Juvela, L., L. T. Sturm, B., and Yamagishi, J. (2023). DDSP-based neural waveform synthesis of polyphonic guitar performance from stingwise MIDI-input. arXiv preprint arXiv:2309.07658.

Jozefowicz, R., Vinyals, O., Schuster, M., Shazeer, N., and Wu, Y. (2016). Exploring the limits of language modeling. arXiv preprint arXiv:1602.02410.

Karras, T., Laine, S., and Aila, T. (2021). A style-based generator architecture for generative adversarial networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(12):4217–4228.

Kawamura, M., Nakamura, T., Kitamura, D., Saruwatari, H., Takahashi, Y., and Kondo, K. (2022). Differentiable digital signal processing mixture model for synthesis parameter extraction from mixture of harmonic sounds. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 941–945, Singapore, Singapore. IEEE.

Keiler, F. and Marchand, S. (2002). Survey on extraction of sinusoids in stationary sounds. In *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, pages 51–58.

Kim, J. W., Bittner, R., Kumar, A., and Bello, J. P. (2019). Neural music synthesis for flexible timbre control. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 176–180, Brighton, UK. IEEE.

Kim, J. W., Salamon, J., Li, P., and Bello, J. P. (2018). Crepe: A convolutional representation for pitch estimation. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 161–165.

Kim, T. H., Fukayama, S., Nishimoto, T., and Sagayama, S. (2013). Statistical approach to automatic expressive rendition of polyphonic piano music. *Guide to Computing for Expressive Music Performance*, pages 145–179.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *Proceedings of the International Conference for Learning Representations (ICLR)*, San Diego, California, USA.

Knoblaugh, A. F. (1944). The clang tone of the pianoforte. *The Journal of the Acoustical Society of America*, 16(Supplement):102–102.

Kong, J., Kim, J., and Bae, J. (2020). Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 33, pages 17022–17033. Curran Associates, Inc.

Kong, Q., Li, B., Chen, J., and Wang, Y. (2022). Giantmidi-piano: A large-scale midi dataset for classical piano music. *Transactions of the International Society for Music Information Retrieval (TISMIR)*.

Kong, Q., Li, B., Song, X., Wan, Y., and Wang, Y. (2021). High-resolution piano transcription with pedals by regressing onset and offset times. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:3707–3717.

Kosta, K., Bandtlow, O. F., and Chew, E. (2018). Mazurkabl: score-aligned loudness, beat, expressive markings data for 2000 chopin mazurka recordings. In *Proceedings of the 4th International Conference on Technologies for Music Notation and Representation (TENOR)*, pages 85–94.

Kumar, K., Kumar, R., de Boissiere, T., Gestin, L., Teoh, W. Z., Sotelo, J., de Brébisson, A., Bengio, Y., and Courville, A. C. (2019). Melgan: Generative adversarial networks for conditional waveform synthesis. In *Advances in Neural Information Processing Systems (NIPS)*, volume 32. Curran Associates, Inc.

Kuznetsov, B., Parker, J. D., and Esqueda, F. (2020). Differentiable IIR filters for machine learning applications. In *Proceedings of the International Conference of Digital Audio Effects (DAFx)*, pages 297–303.

Lagrange, M., Marchand, S., and Rault, J.-B. (2007). Enhancing the tracking of partials for the sinusoidal modeling of polyphonic sounds. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(5):1625–1634.

Larsen, A. B. L., Sønderby, S. K., Larochelle, H., and Winther, O. (2016). Autoencoding beyond pixels using a learned similarity metric. In *Proceednigs of the International Conference on Machine Learning (ICML)*, pages 1558–1566. PMLR.

Latorre, F., Liu, C., Sahoo, D., and Hoi, S. C. (2023). Otw: Optimal transport warping for time series. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

LeCun, Y. and Bengio, Y. (1995). Convolutional networks for images, speech, and time series. *The Handbook of Brain Theory and Neural Networks*, 3361(10).

Lee, S., Choi, H.-S., and Lee, K. (2022). Differentiable artificial reverberation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30:2541–2556.

Lehtonen, H.-M., Askenfelt, A., and Välimäki, V. (2009). Analysis of the part-pedaling effect in the piano. *Journal of the Acoustical Society of America*, 126(2):EL49–EL54.

Li, N., Liu, S., Liu, Y., Zhao, S., and Liu, M. (2019). Neural speech synthesis with transformer network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6706–6713.

Liu, L., Kong, Q., Morfi, V., and Benetos, E. (2022). Performance midi-to-score conversion by neural beat tracking. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 395–402, Bengaluru, India.

Liu, L., Morfi, V., and Benetos, E. (2021). ACPAS: a dataset of aligned classical piano audio and scores for audio-to-score transcription. In *Late-Breaking Demos of the International Society for Music Information Retrieval Conference (ISMIR)*.

Liu, Y., Jin, C., and Gunawan, D. (2023). Ddsp-sfx: Acoustically-guided sound effects generation with differentiable digital signal processing. arXiv preprint arXiv:2309.08060.

Maezawa, A., Yamamoto, K., and Fujishima, T. (2019). Rendering music performance with interpretation variations using conditional variational RNN. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 855–861, Delft, The Netherlands.

Malik, I. and Ek, C. H. (2017). Neural translation of musical style. In *Workshop on Machine Learning for Creativity and Design, Neural Information Processing Systems (NIPS)*, Long Beach, California, USA.

Mallat, S. G. and Zhang, Z. (1993). Matching pursuits with time-frequency dictionaries. *IEEE Transactions on signal processing*, 41(12):3397–3415.

Mao, X., Li, Q., Xie, H., Lau, R. Y., Wang, Z., and Paul Smolley, S. (2017). Least squares generative adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

Masuda, N. and Saito, D. (2023). Improving semi-supervised differentiable synthesizer sound matching for practical applications. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 31:863–875.

Mehri, S., Kumar, K., Gulrajani, I., Kumar, R., Jain, S., Sotelo, J., Courville, A., and Bengio, Y. (2017). SampleRNN: An unconditional end-to-end neural audio generation model. In *Proceediings of the International Conference on Learning Representations (ICLR)*.

Michelashvili, M. and Wolf, L. (2020). Hirearchical timbre-painting and articulation generation. In *Proceedings of the International Society for Music Information Retrieval (ISMIR)*.

Mikkonen, O., Wright, A., Moliner, E., and Välimäki, V. (2023). Neural modeling of magnetic tape recorders. In *Proceedings of the International Conference on Digital Audio Effects (DAFx23)*, pages 196–203, Copenhagen, Denmark.

Mitcheltree, C., Steinmetz, C. J., Comunità, M., and Reiss, J. D. (2023). Modulation extraction for lfo-driven audio effects. In *Proceedings of the International Conference on Digital Audio Effects (DAFx)*.

Müller, M., Konz, V., Bogler, W., and Arifi-Müller, V. (2011). Saarland music data (smd). In *Late-Breaking and Demo Session of the International Society for Music Information Retrieval Conference (ISMIR)*.

Nakamura, E., Yoshii, K., and Katayose, H. (2017). Performance error detection and post-processing for fast and accurate symbolic music alignment. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 347–353, Suzhou, China.

Nercessian, S. (2020). Neural parametric equalizer matching using differentiable biquads. In *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, pages 265–272, Online/Vienna, Austria.

Nguyen, B., Cardinaux, F., and Uhlich, S. (2023). AutoTTS: End-to-end text-to-speech synthesis through differentiable duration modeling. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Nierhaus, G. (2009). *Algorithmic composition: paradigms of automated music generation*. Springer Science & Business Media.

Odena, A., Olah, C., and Shlens, J. (2017). Conditional image synthesis with auxiliary classifier GANs. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 2642–2651. PMLR.

Oord, A., Li, Y., Babuschkin, I., Simonyan, K., Vinyals, O., Kavukcuoglu, K., Driessche, G., Lockhart, E., Cobo, L., Stimberg, F., et al. (2018). Parallel wavenet: Fast high-fidelity speech synthesis. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 3918–3926. PMLR.

Oord, A. V. D., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. (2016). Wavenet: A generative model for raw audio. arXiv preprint arXiv:1609.03499.

Oore, S., Simon, I., Dieleman, S., Eck, D., and Simonyan, K. (2020). This time with feeling: learning expressive musical performance. *Neural Computing and Applications*, 32(4):955–967.

Oxenham, A. J. (2012). Pitch perception. *Journal of Neuroscience*, 32(39):13335–13338.

Palmer, C. (1997). Music performance. *Annual Review of Psychology*, 48(1):115–138. PMID: 9046557.

Pang, Y., Lin, J., Qin, T., and Chen, Z. (2022). Image-to-image translation: Methods and applications. *IEEE Transactions on Multimedia*, 24:3859–3881.

Perez, E., Strub, F., de Vries, H., Dumoulin, V., and Courville, A. C. (2018). FiLM: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 3942–3951. AAAI press.

Peter, S. D., Cancino-Chacón, C. E., Foscarin, F., McLeod, A. P., Henkel, F., Karystinaios, E., and Widmer, G. (2023). Automatic note-level score-to-performance alignments in the ASAP dataset. *Transactions of the International Society for Music Information Retrieval (TISMIR)*.

Platz, F. and Kopiez, R. (2012). When the Eye Listens: A Meta-analysis of How Audio-visual Presentation Enhances the Appreciation of Music Performance. *Music Perception*, 30(1):71–83.

Prenger, R., Valle, R., and Catanzaro, B. (2019). Waveglow: A flow-based generative network for speech synthesis. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3617–3621. IEEE.

Railsback, O. L. (1938). Scale temperament as applied to piano tuning. *The Journal of the Acoustical Society of America*, 9(Supplement):274–274.

Ramoneda, P., Jeong, D., Eremenko, V., Tamer, N. C., Miron, M., and Serra, X. (2024). Combining piano performance dimensions for score difficulty classification. *Expert Systems with Applications*, 238:121776.

Rauhala, J., Laurson, M., Välimäki, V., Lehtonen, H.-M., and Norilo, V. (2008). A parametric piano synthesizer. *Computer Music Journal*, 32(4):17–30.

Ren, P., Xiao, Y., Chang, X., Huang, P.-Y., Li, Z., Gupta, B. B., Chen, X., and Wang, X. (2021a). A survey of deep active learning. *ACM Computing Surveys*, 54(9).

Ren, Y., Hu, C., Tan, X., Qin, T., Zhao, S., Zhao, Z., and Liu, T.-Y. (2021b). Fastspeech 2: Fast and high-quality end-to-end text to speech. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Renault, L., Mignot, R., and Roebel, A. (2022). Differentiable piano model for midi-to-audio performance synthesis. In *Proceedings of the International Conference on Digital Audio Effects (DAFx20in22)*, pages 232–239, Vienna, Austria.

Renault, L., Mignot, R., and Roebel, A. (2023a). Ddsp-piano: a neural sound synthesizer informed by instrument knowledge. *Journal of the Audio Engeneering Society (JAES)*, 71:552–565.

Renault, L., Mignot, R., and Roebel, A. (2023b). Expressive piano performance rendering from unpaired data. In *Proceedings of the International Conference on Digital Audio Effects (DAFx23)*, pages 355–358, Copenhagen, Denmark. Demo and Late-Breaking Result Session.

Rhyu, S., Kim, S., and Lee, K. (2022). Sketching the expression: Flexible rendering of expressive piano performance with self-supervised learning. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Bengaluru, India.

Richard, G., Chouteau, P., and Torres, B. (2024). A fully differentiable model for unsupervised singing voice separation. In *Proceednigs of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Seoul, South Korea.

Richards, C. (2023). *Wearable sound : integrative design for hearing and feeling vibrations*. Theses, Sorbonne Université.

Rigaud, F. (2013). *Models of music signals informed by physics.Application to piano music analysis by non-negative matrix factorization.* Theses, Télécom ParisTech.

Rigaud, F., David, B., and Daudet, L. (2011). A parametric model of piano tuning. In *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, pages 394–399, Paris, France.

Roebel, A. and Bous, F. (2022). Neural vocoding for singing and speaking voices with the multi-band excited wavenet. *Information*, 13(3).

Roy, R. and Kailath, T. (1989). ESPRIT-estimation of signal parameters via rotational invariance techniques. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(7):984–995.

Sai Vanka, S., Safi, M., Rolland, J.-B., and Fazekas, G. (2023). Adoption of ai technology in music mixing workflow: An investigation. In *Proceedings of the 154th Audio Engineering Society Convention*.

Santo, G. D., Prawda, K., Schlecht, S., and Välimäki, V. (2023). Differentiable feedback delay network for colorless reverberation. In *Proceedings of the International Conference on Digital Audio Effects (DAFx23)*, pages 244–251.

Schulze-Forster, K., Doire, C. S., Richard, G., and Badeau, R. (2023). Unsupervised audio source separation using differentiable parametric source models. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 31:1276–1289.

Schwarz, D. (2006). Concatenative sound synthesis: the early years. *Journal of New Music Research*, 35(1):3–22.

Schwär, S. and Müller, M. (2023). Multi-scale spectral loss revisited. *IEEE Signal Processing Letters*, 30:1712–1716.

Serra, X. (1990). *A System for Sound Analysis/Transformation/Synthesis based on a Deterministic plus Stochastic Decomposition.* PhD thesis, Stanford University.

Serra, X. and Smith, J. O. (1990). Spectral modeling synthesis: A sound analysis/synthesis system based on a deterministic plus stochastic decomposition. *Computer Music Journal*, 14(4):12–24.

Shan, S., Hantrakul, L., Chen, J., Avent, M., and Trevelyan, D. (2022). Differentiable wavetable synthesis. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4598–4602, Singapore, Singapore. IEEE.

Shannon, C. (1949). Communication in the presence of noise. *Proceedings of the IRE*, 37(1):10–21.

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423.

Shen, J., Pang, R., Weiss, R. J., Schuster, M., Jaitly, N., Yang, Z., Chen, Z., Zhang, Y., Wang, Y., Skerrv-Ryan, R., Saurous, R. A., Agiomvrgiannakis, Y., and Wu, Y. (2018). Natural tts synthesis by conditioning wavenet on mel spectrogram predictions. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4779–4783, Calgary, Canada. IEEE.

Shi, X., Cooper, E., Wang, X., Yamagishi, J., and Narayanan, S. (2023). Can knowledge of end-to-end text-to-speech models improve neural midi-to-audio synthesis systems? In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Rhodes Island, Greece.

Shier, J., Caspe, F., Robertson, A., Sandler, M., Saitis, C., and McPherson, A. (2023). Differentiable modelling of percussive audio with transient and spectral synthesis. In *Proceeding of the 10th Convention of the European Acoustics Association*.

Shlezinger, N., Whang, J., Eldar, Y. C., and Dimakis, A. G. (2023). Model-based deep learning. *Proceedings of the IEEE*, 111(5):465–499.

Simionato, R., Fasciani, S., and Holm, S. (2024). Physics-informed differentiable method for piano modeling. *Frontiers in Signal Processing*, 3.

Smith, D. and Wood, C. (1981). The USI, or universal synthesizer interface. In *Audio Engineering Society (AES) Convention 70*. Audio Engineering Society.

Smith, J. O. and Serra, X. (1987). PARSHL: An analysis/synthesis program for non-harmonic sounds based on a sinusoidal representation. In *Proceedings of the International Computer Music Conference (ICMC)*, Champaign/Urbana, Illinois, USA. Michigan Publishing.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research (JMLR)*, 15(1):1929–1958.

Stefani, D., Peroni, S., and Turchet, L. (2022). A comparison of deep learning inference engines for embedded real-time audio classification. In *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, pages 256–263, Vienna, Austria.

Steinmetz, C. J., Bryan, N. J., and Reiss, J. D. (2022). Style transfer of audio effects with differentiable signal processing. *Journal of the Audio Engeneering Society (JAES)*, pages 708–721.

Stylianou, Y. (1996). *Harmonic plus noise models for speech, combined with statistical methods, for speech and speaker modification*. PhD thesis, Ecole Nationale Superieure des Telecommunications.

Sumino, H., Bitton, A., Kawai, L., Esling, P., and Harada, T. (2020). Automatic music transcription and instrument transposition with differentiable rendering. In *Proceedings of the 1st Joint Conference on AI Music Creativity (AIMC)*. AIMC.

Tan, H. H. T., Luo, Y.-J., and Herremans, D. (2020). Generative modelling for controllable audio synthesis of expressive piano performance. In *Proceedings of the ICML Workshop on Machine Learning for Media Discovery Workshop (ML4MD)*, Vienna, Austria.

Tang, J., Wiggins, G., and Fazekas, G. (2023). Reconstructing human expressiveness in piano performances with a transformer network. In *Proceedings of the 16th International Symposium on Computer Music Multidisciplinary Research (CMMR)*.

Torres, B., Peeters, G., and Richard, G. (2024). Unsupervised harmonic parameter estimation using differentiable dsp and spectral optimal transport. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Tsalakanidou, F., Papadopoulos, S., Mezaris, V., Kompatsiaris, I., Gray, B., Tsabouraki, D., Kalogerini, M., Negro, F., Montagnuolo, M., de Vos, J., van Kemenade, P., Gravina, D., Mignot, R., Ozerov, A., Schnitzler, F., Garcia-Saez, A., Yannakakis, G. N., Liapis, A., and Kostadinov, G. (2021). The AI4Media project: Use of Next-generation Artificial Intelligence Technologies for Media Sector Applications. In *Proceedings of the International Conference on Artificial Intelligence Applications and Innovations (AIAI)*. Zenodo.

Turian, J. and Henry, M. (2020). I'm sorry for your loss: Spectrally-based audio distances are bad at pitch. In *Proceedings of the "I Can't Believe It's Not Better!" (ICBINB) Workshop at the International Conference on Neural Information Processing Systems (NeurIPS)*, Vancouver, Canada.

Ulyanov, D., Vedaldi, A., and Lempitsky, V. (2016). Instance normalization: the missing ingredient for fast stylization. arXiv preprint arXiv:1607.08022.

Uzrad, N., Barkan, O., Elharar, A., Shvartzman, S., Laufer, M., Wolf, L., and Koenigstein, N. (2024). Diffmoog: a differentiable modular synthesizer for sound matching. arXiv preprint arXiv:2401.12570.

Vahidi, C., Han, H., Wang, C., Lagrange, M., Fazekas, G., and Lostanlen, V. (2023). Mesostructures: Beyond spectrogram loss in differentiable time-frequency analysis. *Journal of the Audio Engineering Society (JAES)*, 71(9):577–585.

Valimaki, V., Parker, J. D., Savioja, L., Smith, J. O., and Abel, J. S. (2012). Fifty years of artificial reverberation. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(5):1421–1448.

Van Den Oord, A., Kalchbrenner, N., and Kavukcuoglu, K. (2016). Pixel recurrent neural networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1747–1756. PMLR.

Wang, B. and Yang, Y.-H. (2019). Performancenet: Score-to-audio music generation with multi-band convolutional residual network. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):1174–1181.

Wang, T.-C., Liu, M.-Y., Zhu, J.-Y., Tao, A., Kautz, J., and Catanzaro, B. (2018). High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8798–8807.

Wang, X., Takaki, S., and Yamagishi, J. (2019). Neural source-filter-based waveform model for statistical parametric speech synthesis. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5916–5920, Brighton, UK. IEEE.

Wang, Y., Skerry-Ryan, R. J., Stanton, D., Wu, Y., Weiss, R. J., Jaitly, N., Yang, Z., Xiao, Y., Chen, Z., Bengio, S., Le, Q. V., Agiomyrgiannakis, Y., Clark, R. A. J., and Saurous, R. A. (2017). Tacotron: Towards end-to-end speech synthesis. In *Proceedings of Interspeech 2017*.

Weinreich, G. (1977). Coupled piano strings. *Journal of the Acoustical Society of America*, 62(6):1474–1484.

West, M. L. (1994). The babylonian musical notation and the hurrian melodic texts. *Music and Letters*, 75(2):161–179.

Widmer, G. and Tobudic, A. (2003). Playing mozart by analogy: Learning multi-level timing and dynamics strategies. *Journal of New Music Research*, 32(3):259–268.

Wiggins, A. and Kim, Y. (2023). A differentiable acoustic guitar model for string-specific polyphonic synthesis. In *2023 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*.

Wright, A., Damskägg, E.-P., and Välimäki, V. (2019). Real-time black-box modelling with recurrent neural networks. In *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, Birmingham, UK. University of Birmingham.

Wright, A., Välimäki, V., and Juvela, L. (2023). Adversarial guitar amplifier modelling with unpaired data. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Rhodes Island, Greece. IEEE.

Wu, X., Xiao, L., Sun, Y., Zhang, J., Ma, T., and He, L. (2022a). A survey of human-in-the-loop for machine learning. *Future Generation Computer Systems*, 135:364–381.

Wu, Y., Manilow, E., Deng, Y., Swavely, R. J., Kastner, K., Cooijmans, T., Courville, A., Huang, A., and Engel, J. (2022b). MIDI-DDSP: Hierarchical modeling of music for detailed control. In *Proceedings of the International Conference on Learning Representations (ICLR)*, Online.

Xue, W. and Sandler, M. (2009). Notes on model-based non-stationary sinusoid estimation methods using derivatives. In *Proceedings of the International Conference on Digital Audio Effects (DAFx)*. Citeseer.

Yamamoto, R., Song, E., and Kim, J.-M. (2020). Parallel wavegan: A fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6199–6203.

Ycart, A. and Benetos, E. (2018). A-maps: Augmented maps dataset with rhythm and key annotations. In *Proceedingns of the International Society for Music Information Retrieval Conference (ISMIR)*, Paris, France.

Young, R. W. (1952). Inharmonicity of plain wire piano strings. *Journal of the Acoustical Society of America*, 24(3):267–273.

Zeghidour, N., Luebs, A., Omran, A., Skoglund, J., and Tagliasacchi, M. (2021). Soundstream: An end-to-end neural audio codec. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30:495–507.

Zhang, H. and Dixon, S. (2023). Disentangling the horowitz factor: Learning content and style from expressive piano performance. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Rhodes Island, Greece.

Zhang, H., Tang, J., Rafee, S. R. M., Dixon, S., and Fazekas, G. (2022). ATEPP: A dataset of automatically transcribed expressive piano performance. In *Proceedings of the International Society for Music Information Retrieval (ISMIR)*, pages 446–453, Bengaluru, India.

Appendix

This manuscript used the following icons from the Noun Project [1], under Creative Common License (CC BY 3.0):

- grand piano by Smashicons from Noun Project
- composer by Amethyst Studio from Noun Project
- sheet music by Rifai from Noun Project
- Violinist by ProSymbols from Noun Project
- bow by Valter Bispo from Noun Project
- Cello by Valter Bispo from Noun Project
- sound mixing board by Juicy Fish from Noun Project
- Ear by Alexander Skowalsky from Noun Project

All other icons were gathered from Freepik[2].

[1] https://thenounproject.com/
[2] https://freepik.com/