Tahiroğlu, Koray; Wang, Shenran; Tampu, Eduard; Lin, Jackie

Deep Learning with Audio: An Explorative Syllabus for Music Composition and Production

Published: 01/08/2023

Document Version
Publisher's PDF, also known as Version of record

Please cite the original version:
Tahiroğlu, K., Wang, S., Tampu, E., & Lin, J. (2023). *Deep Learning with Audio: An Explorative Syllabus for Music Composition and Production*. https://aimc2023.pubpub.org/pub/fkmqnlfh/release/1

AIMC 2023

Deep Learning with Audio: An Explorative Syllabus for Music Composition and Production

Koray Tahiroğlu[1] Shenran Wang[1] Eduard Mihai Tampu[1] Jackie Lin[1]

[1]**Aalto University**

Published on: Aug 29, 2023

URL: https://aimc2023.pubpub.org/pub/fkmqnlfh

ABSTRACT

This paper introduces a new course on deep learning with audio, designed specifically for graduate students in arts studies. The course introduces students the principles of deep learning models in audio and symbolic domain as well as their possible applications in music composition and production. The course covers topics such as data preparation and processing, neural network architectures, training and application of deep learning models in music related tasks. The course also incorporates hands-on exercises and projects, allowing students to apply the concepts learned in class to real-world audio data. In addition, the course introduces a novel approach to integrating audio generation using deep learning models in Pure Data realtime audio synthesis environment, which enables students to create original and expressive audio content in a programming environment that they are more familiar with. The variety of the audio content produced by the students demonstrates the effectiveness of the course in fostering creative approach to their own music productions. Overall, this new course on deep learning with audio represents a significant contribution to the field of artificial intelligence (AI) music and creativity, providing arts graduate students with the necessary skills and knowledge to tackle the challenges of the rapidly evolving AI music technologies.

Author Keywords

AIMC, Deep Learning with Audio, syllabus, curriculum design, AI music education

Introduction

The growing use of artificial intelligence (AI) in music to support musical creativity enables new possibilities in music production. With the rapid advancements in deep learning models, AI has become an invaluable tool for analysing and processing audio data, as well as for generating new and original audio content. As such, it is crucial for graduate students in arts to gain knowledge of deep learning models and their applications in audio processing. In this paper, we introduce a new course on deep learning with audio, specifically designed for graduate-level course in arts studies. The course aims to equip students with the necessary skills and knowledge to explore and practice the rapidly evolving music technologies.

The course covers a range of topics, including data preparation and processing, neural network architectures, and training models. The course provides an overview of recent AI implementations such as, Google Magenta's[1] AI Duet [1], NSynth [2], GANSynth [3], DDSP [4] as well as GANSpaceSynth [5], SampleRNN [6] and optional content for RAVE [7]. We provide code templates that integrate the functionality from open source deep learning audio projects into Pure Data programming environment. We also provide detailed setup instructions and automated scripts to make installation of the required tools as easy as possible (for Pure Data, Python, Conda, Magenta, PyExt). This novel approach to audio generation using deep learning models in Pure Data environment allows students to apply the concepts learned in class to real-world audio data, further explore a particular model and incorporate it into their own project work.

The variety and richness of the audio content produced by the students demonstrates the effectiveness of the course in fostering their creative thinking in music production. Furthermore, learning-diary survey of the participating students indicates a high level of satisfaction and engagement with the course material. In the following sections we provide an overview of the related courses in deep learning and audio processing. We also present our course structure and content, describing the hands-on exercises and projects included in the course. We conclude the paper presenting the feedback received from the students and with a summary of the contributions of the course to the field of AI and creativity.

Related Course Syllabus in Deep Learning and Audio Processing

Several tools and libraries have been developed to facilitate the use of deep learning in audio processing, such as TensorFlow[2], librosa[3] and PyTorch[4]. The accessibility of these libraries and deep learning algorithms has certainly had an impact on the development of syllabi for deep learning courses. With the increasing availability of online resources and open-source software libraries, it has become easier for individuals to learn and experiment with deep learning models. At the same time, the proliferation of deep learning algorithms has also resulted in a need for university level courses that focus on specific applications of deep learning, reflecting the growing demand for experts in these areas.

In terms of education, there have been efforts to incorporate deep learning models and AI methods into music and audio-related curricula. For instance, Center for Computer Research in Music and Acoustics (CCRMA) at Stanford University has been offering courses in deep learning for music and audio[5]. In this early example of a research seminar based course in 2017, the focus has been on introducing neural networks, signal processing, machine learning, including ways to teach computers how to compose music as well as speech recognition and content-based music recommendations. More recently, Music and AI course[6] has been scheduled to be offered during winter 2023 in CCRMA with an extended focus on AI, HCI and Music, introducing tools and techniques with a critical reflection on aesthetics, ethical concerns in relation to human interaction. One of the most interesting graduate level courses in music computing with a more comprehensive focus on AI is the course Musical Machine Learning[7] offered at Sorbonne Université and Institute for Research and Coordination in Acoustics/Music (IRCAM). The course covers a wider range of topics varying from data and GPU computing to classification of music structures and from data complexity to deep learning models for music generation. Deep learning content covers recent approaches in deep learning architectures, introducing tutorials for coding exercises that are developed as set of Python Notebooks. The course explores the intersection of music and machine learning, providing students the fundamental concepts of machine learning and how they can be applied to music-related problems. It is important to note that the course structure is rooted in advanced machine learning domain with specific application to audio data. It is a project based course, which involve tasks for developing tools with machine learning models that solve a problem in music processing, analysis, or generation.

Department of Music and Performing Arts Professions at New York University also offers a course in deep learning for music and audio processing. The course Music, the Mind and Artificial Intelligences[8] is an interdisciplinary course that combines music perception and cognition with artificial intelligence. The course aims to provide students with an understanding of how humans perceive and process music, and how this understanding can be applied to building a conceptual framework for exploring AI models. Similarly, Berklee College of Music offers the Machine Learning for Musicians [9] course, which is designed to introduce students to the practical machine learning basics and its applications to music and art. It is aimed at musicians who want to explore how machine learning can be used to enhance their creative process. The course also covers the topics of data and generative algorithms, with a focus on exploring machine learning applied to a wide range of creative works.

Our course on deep learning with audio for arts graduate students builds on the existing research and educational efforts in the field. The course aims to provide a comprehensive understanding of deep learning and its applications in audio processing, as well as to foster creativity in audio-related tasks.

Course Structure and Content

In 2019, we started teaching DOM-E5129 Intelligent Computational Media, a course aiming to enable students to use deep learning and artificial intelligence tools for applications in arts, design and games. The course included teaching material on audio related applications, however we found that the broad syllabus left us with insufficient time to focus deeply on the audio content. As a result, the Intelligent Computational Media course did not meet the expectations of the graduate students with music production interest. To address this, we intended to move the AI audio content into a new dedicated course and we started teaching the course Deep Learning with Audio first time in 2021.

During the three weeks course period, every week we meet with students from Tuesday to Friday everyday for three hours, in total for 36 contact hours. Students have 24/7 access to the Microsoft Azure cloud computing Virtual Machines in Aalto University and they are able to train their deep learning models for the course exercises as well as for their projects during this three weeks period. The current module of installations and course exercises regarding the content generation require Linux computers. University library provides laptop machines with Ubuntu 20.04 LTS for those students of this course. Deep Learning with Audio is a project-based course, we dedicate half of the contact hours for project work, the lecturer and the teaching assistants support students by giving sufficient guidance, feedback and tutoring. At the end of the course, students submit and present their music composition projects.

Intended learning outcomes of the Deep Learning with Audio course is:

- Gain general knowledge of the recent audio and symbolic domain deep learning models, AI methods and network architectures.

- To be able to prepare data sets and train deep learning models using cluster network computers in Aalto University.
- Explore the differences in input, computational cost and sonic characteristics between different models.
- Create music compositions using audio contents generated through these deep learning models.

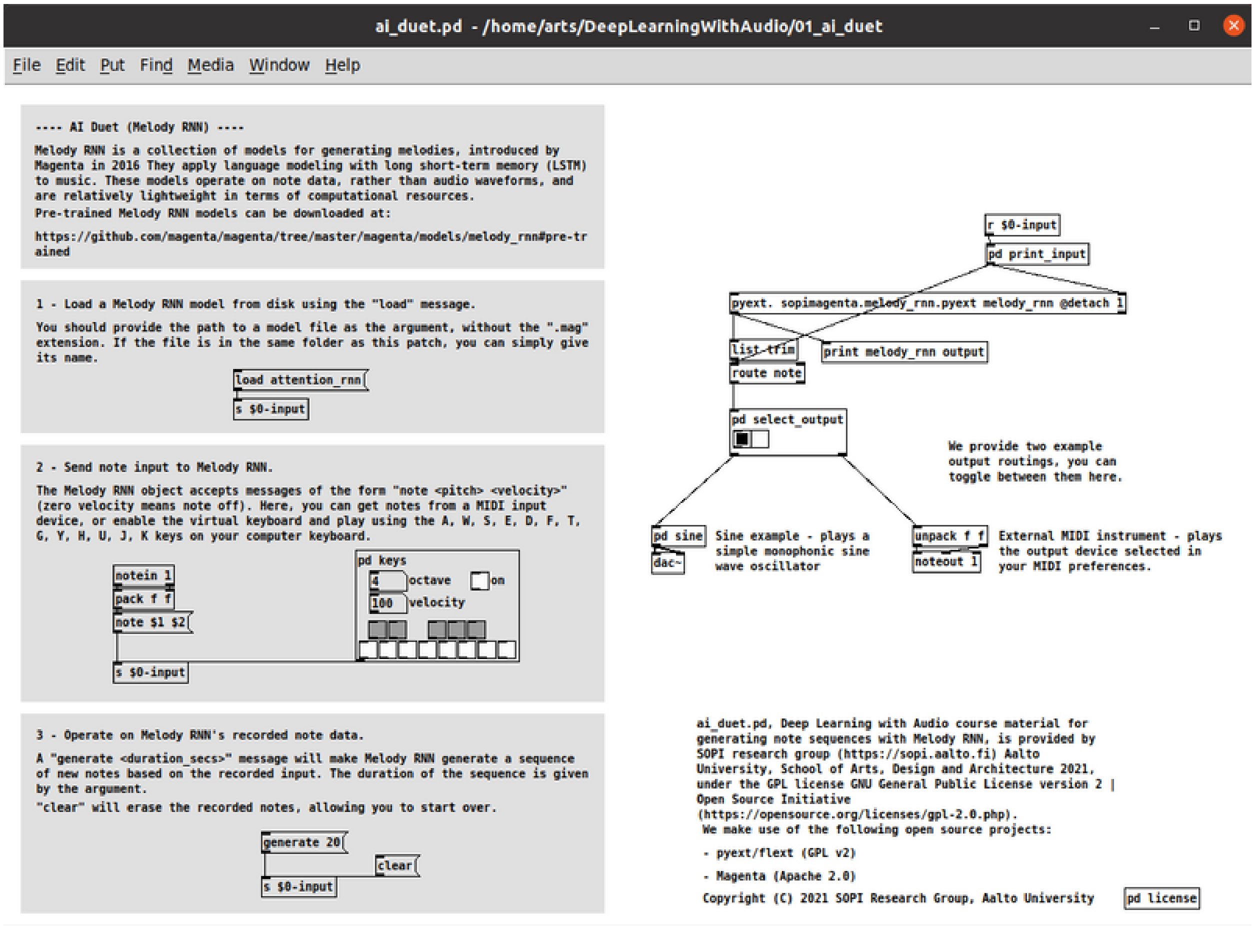
Course Syllabus and Hands-on Exercises

The course begins with an introduction to audio and symbolic domain applications with deep learning models and AI methods. The course examples present the state of the art applications. Following the introduction to the field, we continue with the installation of the required tools; Pure Data, Python, Conda, Magenta, PyExt. The GitHub page of the course[10] hosts and provides detailed setup instructions and automated scripts to make the required tools installed as easy as possible. One of the important objectives of this course is to make deep learning models more accessible to arts students by incorporating them into Pure Data (Pd), a data-flow programming language for audio processing. To achieve this, we have created Pd objects for each module of the course, utilising deep learning algorithms. Pyext external [8] that is built with flext [9] development layer for Pd enables the creation of Pd objects to run Python scripts. Previously, Pyext only supported Python 2, so we had to modify it to work with Python 3. This was a challenging task as the Python APIs underwent significant changes between these two versions [10].

Day1 - Introduction to Deep Learning with Audio

Starting with the symbolic domain content, the course introduces AI Duet (Melody RNN). In 2016, Magenta introduced Melody RNN, a set of models for generating melodies using long short-term memory (LSTM) applied to note data. These models are lightweight in terms of computational resources as they operate on note data instead of audio waveforms. Yotam Mann's AI Duet is an online experiment that enables users to play a duet with the computer using Melody RNN [1]. Figure 1 shows the Pure Data implementation for the AI Duet with Melody RNN.

The first day content, while introducing the RNN model, also covers the building blocks of a neural network components, such as layer connections, inputs weights, bias, summation function, activation and output. The teaching exercise develops into a discussion in the class, questioning in what ways training a Neural Network means finding the appropriate weights of the neural connections, calculating the difference between output and an expected output. First day continues with recurrent neural network architecture for sequence prediction. Students explore different types of AI-Duet's pre-trained models and discuss how their output differ.



---- AI Duet (Melody RNN) ----

Melody RNN is a collection of models for generating melodies, introduced by Magenta in 2016 They apply language modeling with long short-term memory (LSTM) to music. These models operate on note data, rather than audio waveforms, and are relatively lightweight in terms of computational resources.

Pre-trained Melody RNN models can be downloaded at:

https://github.com/magenta/magenta/tree/master/magenta/models/melody_rnn#pre-trained

1 - Load a Melody RNN model from disk using the "load" message.

You should provide the path to a model file as the argument, without the ".mag" extension. If the file is in the same folder as this patch, you can simply give its name.

2 - Send note input to Melody RNN.

The Melody RNN object accepts messages of the form "note <pitch> <velocity>" (zero velocity means note off). Here, you can get notes from a MIDI input device, or enable the virtual keyboard and play using the A, W, S, E, D, F, T, G, Y, H, U, J, K keys on your computer keyboard.

3 - Operate on Melody RNN's recorded note data.

A "generate <duration_secs>" message will make Melody RNN generate a sequence of new notes based on the recorded input. The duration of the sequence is given by the argument.

"clear" will erase the recorded notes, allowing you to start over.

We provide two example output routings, you can toggle between them here.

Sine example - plays a simple monophonic sine wave oscillator

External MIDI instrument - plays the output device selected in your MIDI preferences.

ai_duet.pd, Deep Learning with Audio course material for generating note sequences with Melody RNN, is provided by SOPI research group (https://sopi.aalto.fi) Aalto University, School of Arts, Design and Architecture 2021, under the GPL license GNU General Public License version 2 | Open Source Initiative (https://opensource.org/licenses/gpl-2.0.php).

We make use of the following open source projects:

- pyext/flext (GPL v2)
- Magenta (Apache 2.0)

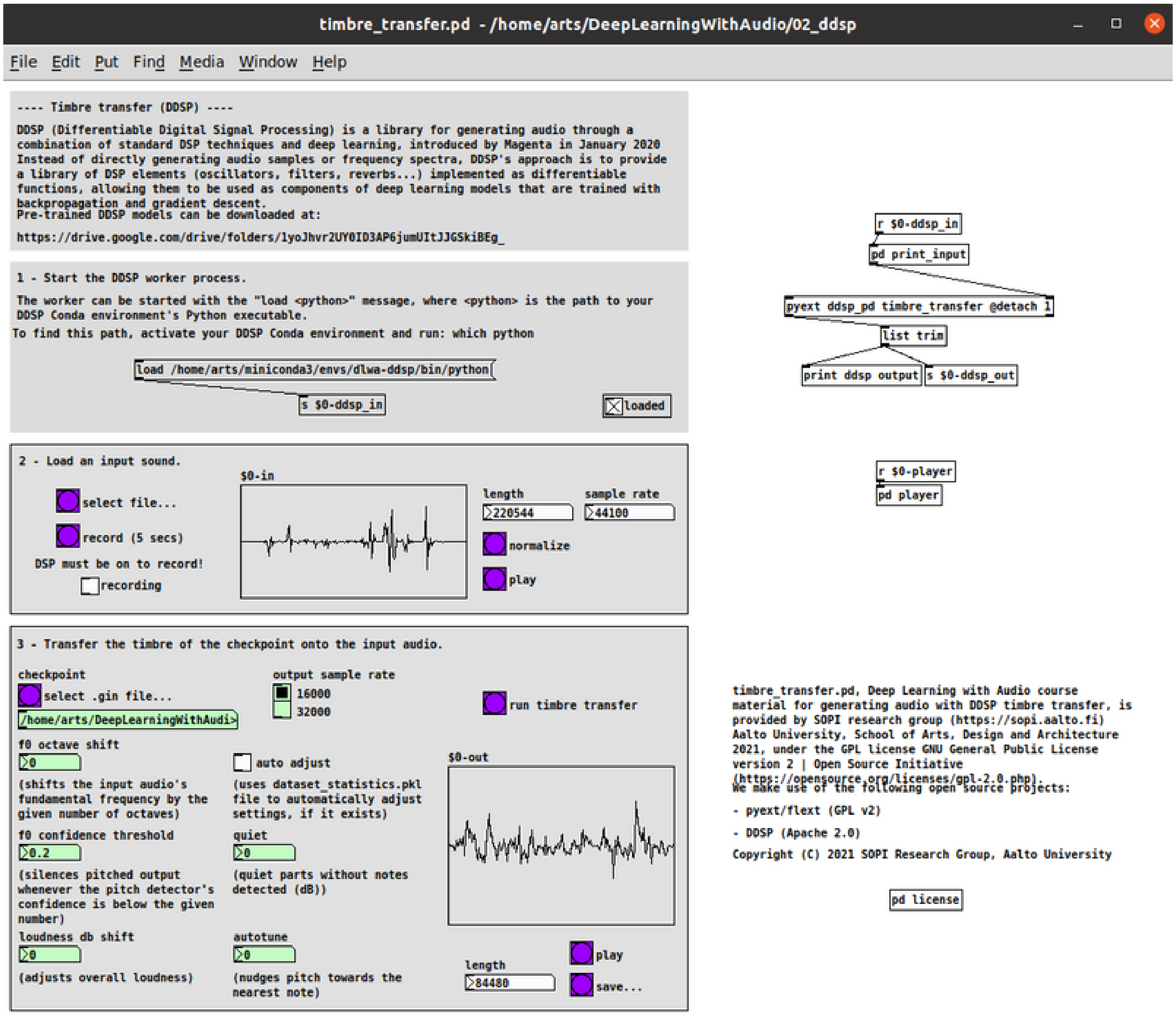
Copyright (C) 2021 SOPI Research Group, Aalto University

Figure 1. AI-Duet Pd patch allows for midi input as well as key input to feed Melody RNN, generating sequences of notes.

Day 2 - DDSP (Differentiable Digital Signal Processing)

In the second day, the course introduces the DDSP (Differentiable Digital Signal Processing) library by Magenta that combines standard DSP techniques and deep learning to generate audio [4]. Unlike other methods that directly generate audio samples or frequency spectra, DDSP provides a library of DSP elements, such as oscillators, filters, and reverbs, which are implemented as differentiable functions. This allows them to be used as components in deep learning models trained with backpropagation and gradient descent. In a DDSP model, the neural network generates parameters for the DSP elements, which then synthesize or process audio based on these parameters. Because standard DSP elements have a well-understood behaviour, this approach enables greater interpretability than typical black-box deep learning models [4].

Students follow the conda dlwa-ddsp setup in GitHub and explore the DDSP timber-transfer through `timber-transfer.pd` patch. The structure of the Pd patch is shown in Figure 2. The course provides pre-trained DDSP models for the class exercise in which students try a few different combinations of input audio and checkpoints as well as discuss how the inputs' characteristics affect the output. They also experiment with shifting different range of values for the the *fo* octave shift, *fo* confidence threshold and loudness *dB* parameters. The course assignment for the following day involves students to prepare their own audio data set and start training their own model following the instructions provided on the Github page[11].



timbre_transfer.pd - /home/arts/DeepLearningWithAudio/02_ddsp

---- Timbre transfer (DDSP) ----

DDSP (Differentiable Digital Signal Processing) is a library for generating audio through a combination of standard DSP techniques and deep learning, introduced by Magenta in January 2020 Instead of directly generating audio samples or frequency spectra, DDSP's approach is to provide a library of DSP elements (oscillators, filters, reverbs...) implemented as differentiable functions, allowing them to be used as components of deep learning models that are trained with backpropagation and gradient descent.
Pre-trained DDSP models can be downloaded at:

https://drive.google.com/drive/folders/1yoJhvr2UY0ID3AP6jumUItJJGSkiBEg_

1 - Start the DDSP worker process.

The worker can be started with the "load <python>" message, where <python> is the path to your DDSP Conda environment's Python executable.
To find this path, activate your DDSP Conda environment and run: which python

```
load /home/arts/miniconda3/envs/dlwa-ddsp/bin/python
```

☒ loaded

2 - Load an input sound.

DSP must be on to record!

☐ recording

length: 220544

sample rate: 44100

3 - Transfer the timbre of the checkpoint onto the input audio.

checkpoint: /home/arts/DeepLearningWithAudi>

output sample rate: ■ 16000 □ 32000

f0 octave shift: 0

(shifts the input audio's fundamental frequency by the given number of octaves)

☐ auto adjust

(uses dataset_statistics.pkl file to automatically adjust settings, if it exists)

f0 confidence threshold: 0.2

(silences pitched output whenever the pitch detector's confidence is below the given number)

quiet: 0

(quiet parts without notes detected (dB))

loudness db shift: 0

(adjusts overall loudness)

autotune: 0

(nudges pitch towards the nearest note)

length: 84480

timbre_transfer.pd, Deep Learning with Audio course material for generating audio with DDSP timbre transfer, is provided by SOPI research group (https://sopi.aalto.fi) Aalto University, School of Arts, Design and Architecture 2021, under the GPL license GNU General Public License version 2 | Open Source Initiative (https://opensource.org/licenses/gpl-2.0.php).
We make use of the following open source projects:

- pyext/flext (GPL v2)
- DDSP (Apache 2.0)

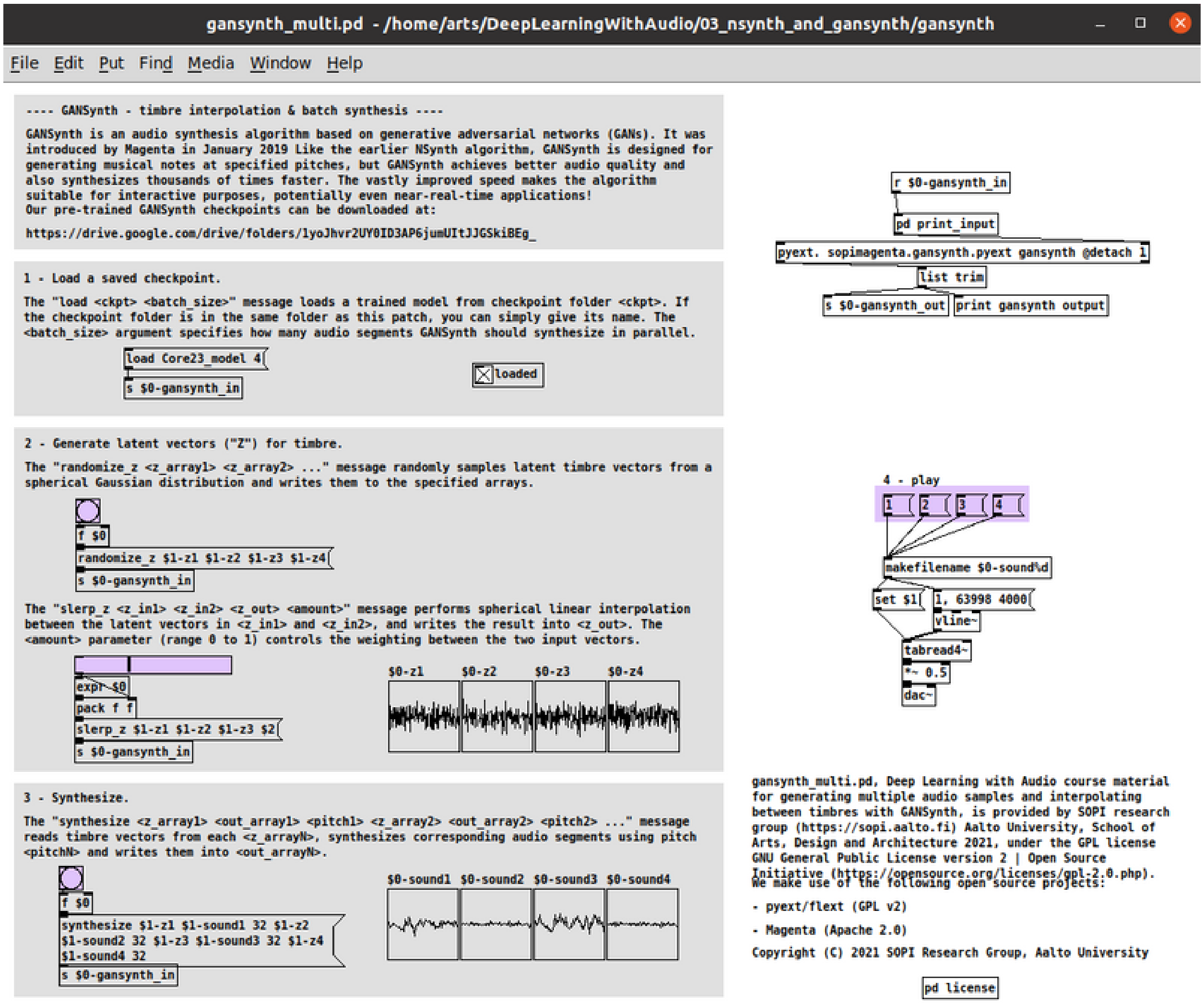
Copyright (C) 2021 SOPI Research Group, Aalto University

Figure 2. Students go through the timbre transfer exercise using DDSP implantation in timbre_transfer Pd patch.

Day 3 - GANSynth: Adversarial Neural Audio Synthesis

The third day of the course introduces the GANSynth, Magenta's deep learning model for generating audio [3]. After discussing with the students the initial results of the DDSP training, we introduce generative adversarial networks (GANs) [11] through GANSynth architecture. While earlier algorithms like NSynth [12] were designed for generating musical notes at specific pitches, GANSynth achieves better audio quality and can synthesize audio thousands of times faster. This dramatic increase in speed makes the algorithm well-suited for interactive purposes, including near-real-time applications. While GANs have been used successfully for generating high-resolution images since at least 2016, adapting them for audio generation has proved challenging due to their difficulty in capturing local latent structure. This often results in audio lacking phase coherence and quality. GANSynth overcomes this problem by improving the network architecture and audio representation, allowing for better modelling of global latent structure and more efficient synthesis [3].

Students generate some random latent vectors, synthesising sounds using `gansynth.pd` with the *all_instruments* checkpoint [12] and experimenting with different timbres that the GANSynth model generates. Furthermore they interpolate between different latent vector points using `gansynth_multi.pd` patch and elaborate how the resulting synthesised sound compare to the sounds from the original latent vectors. Figure 3 presents step-by-step instructions to generate audio samples using the Pd patch. Similar to the previous day content, students prepare their own audio dataset and start a GANSynth model training following the GitHub instructions.



---- GANSynth - timbre interpolation & batch synthesis ----

GANSynth is an audio synthesis algorithm based on generative adversarial networks (GANs). It was introduced by Magenta in January 2019 Like the earlier NSynth algorithm, GANSynth is designed for generating musical notes at specified pitches, but GANSynth achieves better audio quality and also synthesizes thousands of times faster. The vastly improved speed makes the algorithm suitable for interactive purposes, potentially even near-real-time applications!
Our pre-trained GANSynth checkpoints can be downloaded at:

https://drive.google.com/drive/folders/1yoJhvr2UY0ID3AP6jumUItJJGSkiBEg_

1 - Load a saved checkpoint.

The "load <ckpt> <batch_size>" message loads a trained model from checkpoint folder <ckpt>. If the checkpoint folder is in the same folder as this patch, you can simply give its name. The <batch_size> argument specifies how many audio segments GANSynth should synthesize in parallel.

2 - Generate latent vectors ("Z") for timbre.

The "randomize_z <z_array1> <z_array2> ..." message randomly samples latent timbre vectors from a spherical Gaussian distribution and writes them to the specified arrays.

The "slerp_z <z_in1> <z_in2> <z_out> <amount>" message performs spherical linear interpolation between the latent vectors in <z_in1> and <z_in2>, and writes the result into <z_out>. The <amount> parameter (range 0 to 1) controls the weighting between the two input vectors.

3 - Synthesize.

The "synthesize <z_array1> <out_array1> <pitch1> <z_array2> <out_array2> <pitch2> ..." message reads timbre vectors from each <z_arrayN>, synthesizes corresponding audio segments using pitch <pitchN> and writes them into <out_arrayN>.

gansynth_multi.pd, Deep Learning with Audio course material for generating multiple audio samples and interpolating between timbres with GANSynth, is provided by SOPI research group (https://sopi.aalto.fi) Aalto University, School of Arts, Design and Architecture 2021, under the GPL license GNU General Public License version 2 | Open Source Initiative (https://opensource.org/licenses/gpl-2.0.php).
We make use of the following open source projects:

- pyext/flext (GPL v2)

- Magenta (Apache 2.0)

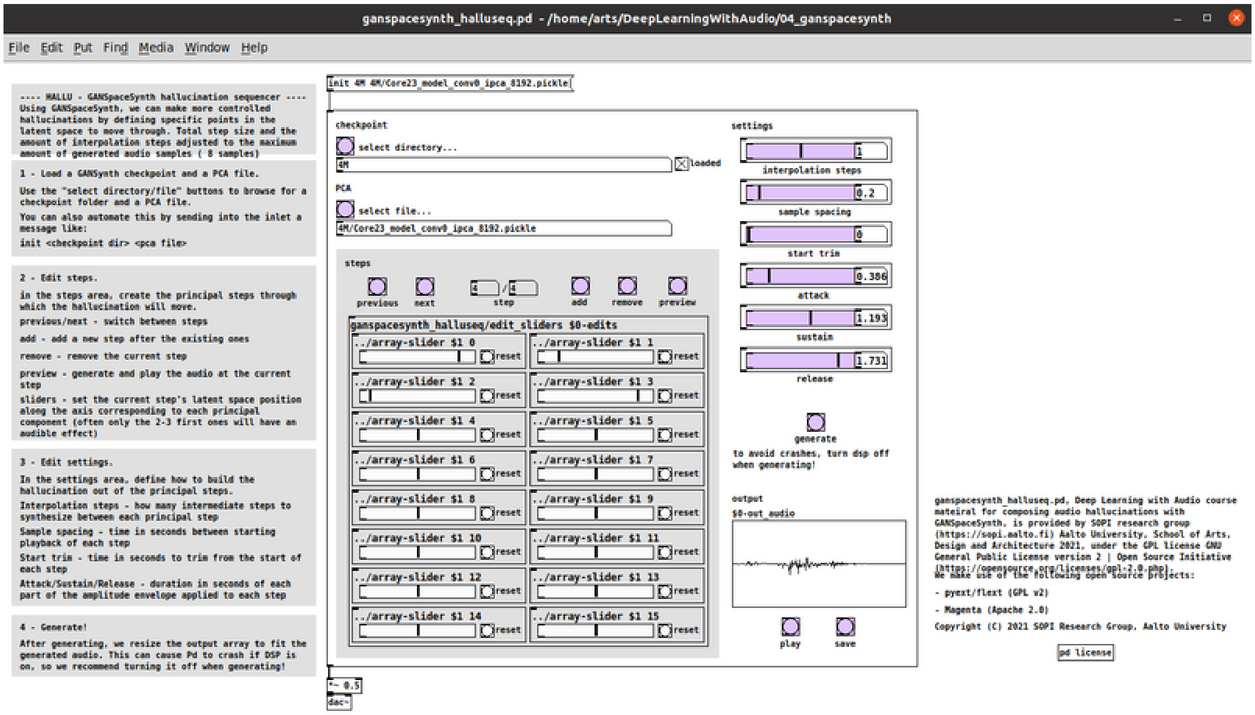
Copyright (C) 2021 SOPI Research Group, Aalto University

Figure 3. gansynth_multi.pd patch allows students to generate multiple random latent vectors and and generate audio samples.

Day 4 - GANSpaceSynth: Dimensionality Reduction for Audio Synthesis

Magenta's GANSynth algorithm generates novel sounds, at the same time it provides limited control over the generation process. While users can specify the pitch, the timbre is determined by a latent vector in a high-dimensional space that is difficult to navigate. To obtain a variety of timbres, random latent vectors can be sampled, and interpolation between them can be used to morph from one timbre to another. However, this approach still offers limited human input into the generation process. To address this limitation, day 4 content provides methods and tools for students with more control over the generated sounds.

The course covers the GANSpaceSynth, a hybrid architecture in which the GANSpace technique is applied to GANSynth [5]. GANSpaceSynth feeds random latent vectors into GANSynth and compute a principal component analysis (PCA) of the activations on the first two convolution layers, `conv0` and `conv1`. The output shape of these layers is (2, 16, 256), i.e. a total of 8192 values. Incremental PCA is used to compute in batches and limit memory consumption. Interpreting the output of GAN models can be challenging due to the complexity of their latent space. However, by using principal component analysis PCA output, it is possible to gain more control over the navigation of the latent space in a GANSynth model [13][14]. This allows for a more structured and controlled exploration of the latent space, enabling a better understanding of the underlying structure and potentially improving the interpretability of GAN models. Students work with the `ganspacesynth_halluseq.pd` patch and generate audio samples by moving along some parts of the latent space that they find interesting (Figure 4). Hands-on exercise also includes interpolating between different vector points and snitching those samples together in a short composition structure. In this exercise, using two different checkpoints, students compare the audio features that are extracted from PCA on 3-dimensions and describe their semantic meanings. Once the previously started GANSynth trainings are completed then students compute the PCA for GANSpaceSynth using the `gansynth_ganspace` script.



---- HALLU - GANSpaceSynth hallucination sequencer ----
Using GANSpaceSynth, we can make more controlled hallucinations by defining specific points in the latent space to move through. Total step size and the amount of interpolation steps adjusted to the maximum amount of generated audio samples (8 samples)

1 - Load a GANSynth checkpoint and a PCA file.

Use the "select directory/file" buttons to browse for a checkpoint folder and a PCA file.

You can also automate this by sending into the inlet a message like:

init <checkpoint dir> <pca file>

2 - Edit steps.

in the steps area, create the principal steps through which the hallucination will move.

previous/next - switch between steps

add - add a new step after the existing ones

remove - remove the current step

preview - generate and play the audio at the current step

sliders - set the current step's latent space position along the axis corresponding to each principal component (often only the 2-3 first ones will have an audible effect)

3 - Edit settings.

In the settings area, define how to build the hallucination out of the principal steps.

Interpolation steps - how many intermediate steps to synthesize between each principal step

Sample spacing - time in seconds between starting playback of each step

Start trim - time in seconds to trim from the start of each step

Attack/Sustain/Release - duration in seconds of each part of the amplitude envelope applied to each step

4 - Generate!

After generating, we resize the output array to fit the generated audio. This can cause Pd to crash if DSP is on, so we recommend turning it off when generating!

```
init 4M 4M/Core23_model_conv0_ipca_8192.pickle
```

4M

4M/Core23_model_conv0_ipca_8192.pickle

../array-slider $1 0 ../array-slider $1 1 ../array-slider $1 2 ../array-slider $1 3 ../array-slider $1 4 ../array-slider $1 5 ../array-slider $1 6 ../array-slider $1 7 ../array-slider $1 8 ../array-slider $1 9 ../array-slider $1 10 ../array-slider $1 11 ../array-slider $1 12 ../array-slider $1 13 ../array-slider $1 14 ../array-slider $1 15

1 interpolation steps

0.2 sample spacing

0 start trim

0.386 attack

1.193 sustain

1.731 release

to avoid crashes, turn dsp off when generating!

ganspacesynth_halluseq.pd, Deep Learning with Audio course mateiral for composing audio hallucinations with GANSpaceSynth, is provided by SOPI research group (https://sopi.aalto.fi) Aalto University, School of Arts, Design and Architecture 2021, under the GPL license GNU General Public License version 2 | Open Source Initiative (https://opensource.org/licenses/gpl-2.0.php).
We make use of the following open source projects:

- pyext/flext (GPL v2)
- Magenta (Apache 2.0)

Copyright (C) 2021 SOPI Research Group, Aalto University

```
*~ 0.5
dac~
```

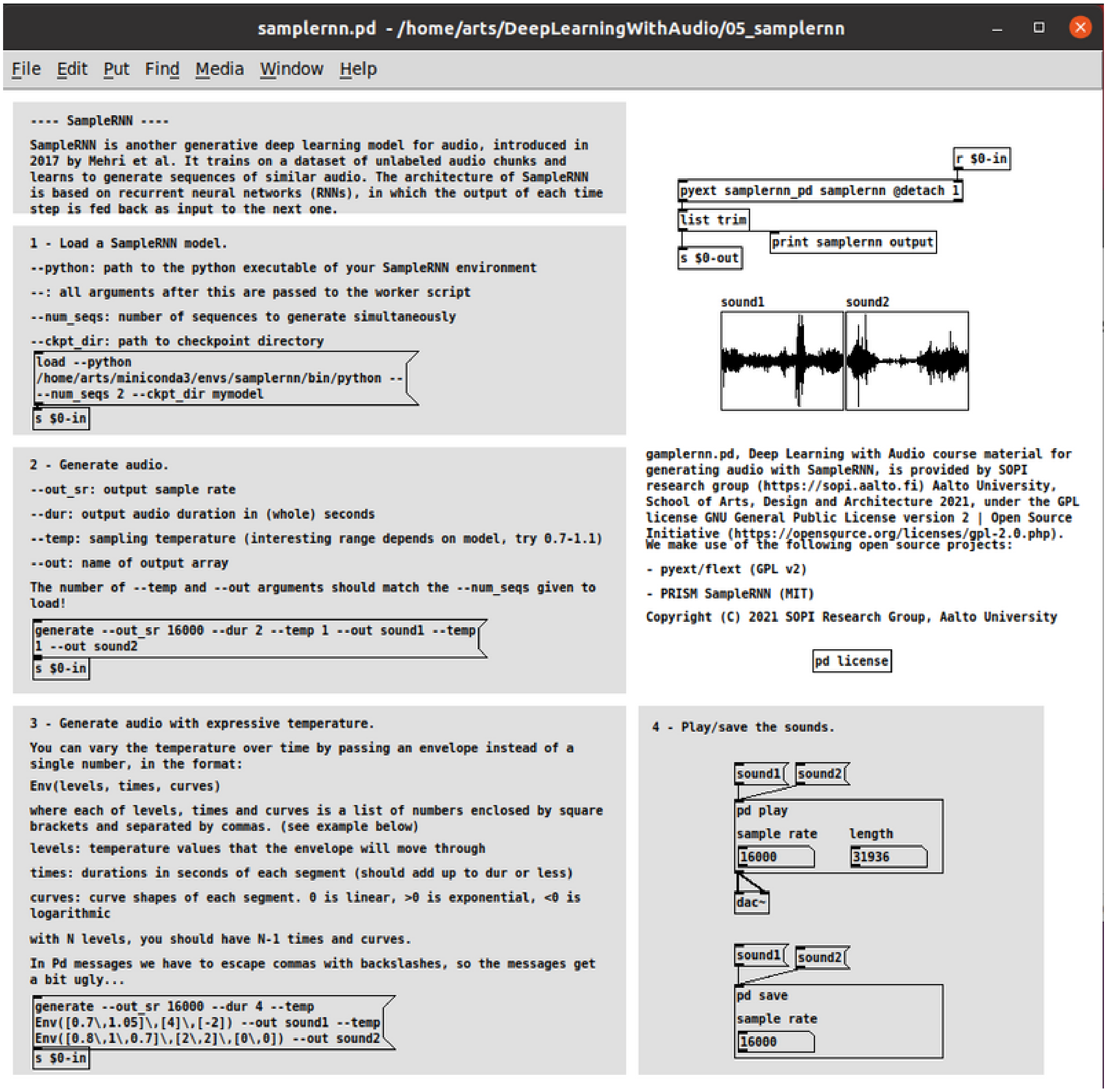
Figure 4. Hallu implementation in ganspacesynth_halluseq.pd patch provides students a template for generating audio samples while moving along the latent space through reduced dimensionality.

Day 5 - SampleRNN: Generating Sequences of Audio

SampleRNN is a deep learning model for audio generation, created in 2017 by Soroush Mehri, Kundan Kumar, Ishaan Gulrajani, et al. [6]. It utilises unlabeled audio chunks to learn and generate sequences of similar audio. The model is based on recurrent neural networks (RNNs), which are particularly suitable for modeling sequential data. However, RNNs often encounter the vanishing gradient problem, where the network's ability to learn is hindered by gradients shrinking exponentially during backpropagation.

To tackle the vanishing gradient problem, two RNN variants have been developed: gated recurrent units (GRU) and long short-term memory (LSTM) [6]. SampleRNN can be configured to work with either variant and it is currently unclear which one is universally better. Most importantly, SampleRNN utilises RNNs, enabling it to generate audio sequences of any length. This is in contrast to the fixed-length sequences of GANSynth, however SampleRNN is significantly slower. As the original implementation of SampleRNN is unmaintained and difficult to set up, in the Deep Learning with Audio course we use our own fork of the `PRiSM SampleRNN` implementation.

Following the conda SampleRNN setup, students generate audio samples with different values for the sampling temperature parameter using pre-trained models (Figure 5). After the contact teaching hours, students also prepare their own audio dataset and start SampleRNN training using cluster computers. Five to seven minutes waiting time for generation of two four-seconds length of audio samples, using CPU only, brings in some challenges to this model to be used with generative algorithms.


samplernn.pd - /home/arts/DeepLearningWithAudio/05_samplernn
File Edit Put Find Media Window Help

---- SampleRNN ----

SampleRNN is another generative deep learning model for audio, introduced in 2017 by Mehri et al. It trains on a dataset of unlabeled audio chunks and learns to generate sequences of similar audio. The architecture of SampleRNN is based on recurrent neural networks (RNNs), in which the output of each time step is fed back as input to the next one.

1 - Load a SampleRNN model.

--python: path to the python executable of your SampleRNN environment

--: all arguments after this are passed to the worker script

--num_seqs: number of sequences to generate simultaneously

--ckpt_dir: path to checkpoint directory

```
load --python
/home/arts/miniconda3/envs/samplernn/bin/python --
--num_seqs 2 --ckpt_dir mymodel
s $0-in
```

2 - Generate audio.

--out_sr: output sample rate

--dur: output audio duration in (whole) seconds

--temp: sampling temperature (interesting range depends on model, try 0.7-1.1)

--out: name of output array

The number of --temp and --out arguments should match the --num_seqs given to load!

```
generate --out_sr 16000 --dur 2 --temp 1 --out sound1 --temp
1 --out sound2
s $0-in
```

3 - Generate audio with expressive temperature.

You can vary the temperature over time by passing an envelope instead of a single number, in the format:

Env(levels, times, curves)

where each of levels, times and curves is a list of numbers enclosed by square brackets and separated by commas. (see example below)

levels: temperature values that the envelope will move through

times: durations in seconds of each segment (should add up to dur or less)

curves: curve shapes of each segment. 0 is linear, >0 is exponential, <0 is logarithmic

with N levels, you should have N-1 times and curves.

In Pd messages we have to escape commas with backslashes, so the messages get a bit ugly...

```
generate --out_sr 16000 --dur 4 --temp
Env([0.7\,1.05]\,[4]\,[-2]) --out sound1 --temp
Env([0.8\,1\,0.7]\,[2\,2]\,[0\,0]) --out sound2
s $0-in
```

```
r $0-in
pyext samplernn_pd samplernn @detach 1
list trim
print samplernn output
s $0-out
```

gamplernn.pd, Deep Learning with Audio course material for generating audio with SampleRNN, is provided by SOPI research group (https://sopi.aalto.fi) Aalto University, School of Arts, Design and Architecture 2021, under the GPL license GNU General Public License version 2 | Open Source Initiative (https://opensource.org/licenses/gpl-2.0.php).
We make use of the following open source projects:

- pyext/flext (GPL v2)
- PRISM SampleRNN (MIT)

Copyright (C) 2021 SOPI Research Group, Aalto University

pd license

4 - Play/save the sounds.

Figure 5. Samplernn.pd is the course material for generating audio samples with SamleRNN model.

We have already included RAVE [7] implementation in our syllabus, however regarding the three weeks schedule and the project work tutoring, we have decided to present this module as an optional course content to the students this year, for those students who would like to go through the session content by themselves following the course Github[13] instructions.

Assessment Methods and Criteria

13 students registered to the Deep Learning with Audio course in 2023 and 11 of them completed the course. Each student project work has been assessed with the following criteria: sound synthesis design decisions through the challenges they face working with deep learning models, aesthetic and originality of their sound synthesis and / or audio analysis implementation, analysis of their project components, composition strategies in relation to the production of the project, code design quality in terms of the ways they use deep learning models to come up and develop alternative solutions for their idea generation and project implementation. At the same, active participation in the course, returning course assignments and project interim as well as project final presentation and final delivery of the music composition project including the video demonstration is required to complete the course. Students also write project final concept paper / learning diary (~750 words) and deliver together with their final compositions.

Student Projects and Compositions

Following the content and hands-on exercise sessions, students begin the process of working on their compositions. First they present their composition ideas, describing a particular narrative that links the composition to the deep learning models introduced in this course. Students spend time going through the course materials regarding their chosen topic and brainstorming ideas for their composition. Once students have a clear idea of what they want to do in their composition, they begin outlining their composition. This might involve creating a rough draft or breaking the composition down into sections. With their outline or plan in place, they may revise their initial ideas or make other changes to improve the overall flow and coherence of the piece. Throughout this process, students are encouraged to think about how they can incorporate the deep learning models into their compositions. This might involve using specific concepts or techniques to enhance the narrative, or exploring how the models can be applied to the structure of the compositions being discussed. Here below we present, in their own words, three student projects composed during the Deep Learning with Audio course this year.

Bagatelle, Shenran Wang

The idea for my composition project was to use AI/DL models as a tool in the composition process. I have faced multiple creativity blocks when composing music, where the lack of inspiration or musical ideas impeded the progress of a composition. By experimenting with varying degrees of autonomy for both the models and myself in a composition process, I would gain an initial understanding about how AI/DL can both help and influence the creative process in a positive way.

The composition Bagatelle is realised using AI Duet by Shenran Wang

The composition I worked on in this course is part of a set of piano bagatelles (Figure 6). The new bagatelle is based on music set theory, and the idea is that at any given time, only certain pitch classes can be used in the composition. The available pitch classes gradually change over time, which brings about motion in the piece from a harmonic aspect. In addition, there is a temporal coherency that the music adheres to, namely the temporal distance at which consecutive notes play at is determined by the following additive process: 1-2-3-4-3-2-1-2-3-4-3-etc. For example, the first note plays one quarter note before the second one, the third note plays a half note after second, the fourth plays three quarters after the third, and so on. This sequence may become shorter or longer during the composition. This brings an ambient feel to the music, yet still contains great temporal oscillation that can keep the music fresh when combined with gradually changing pitch classes.

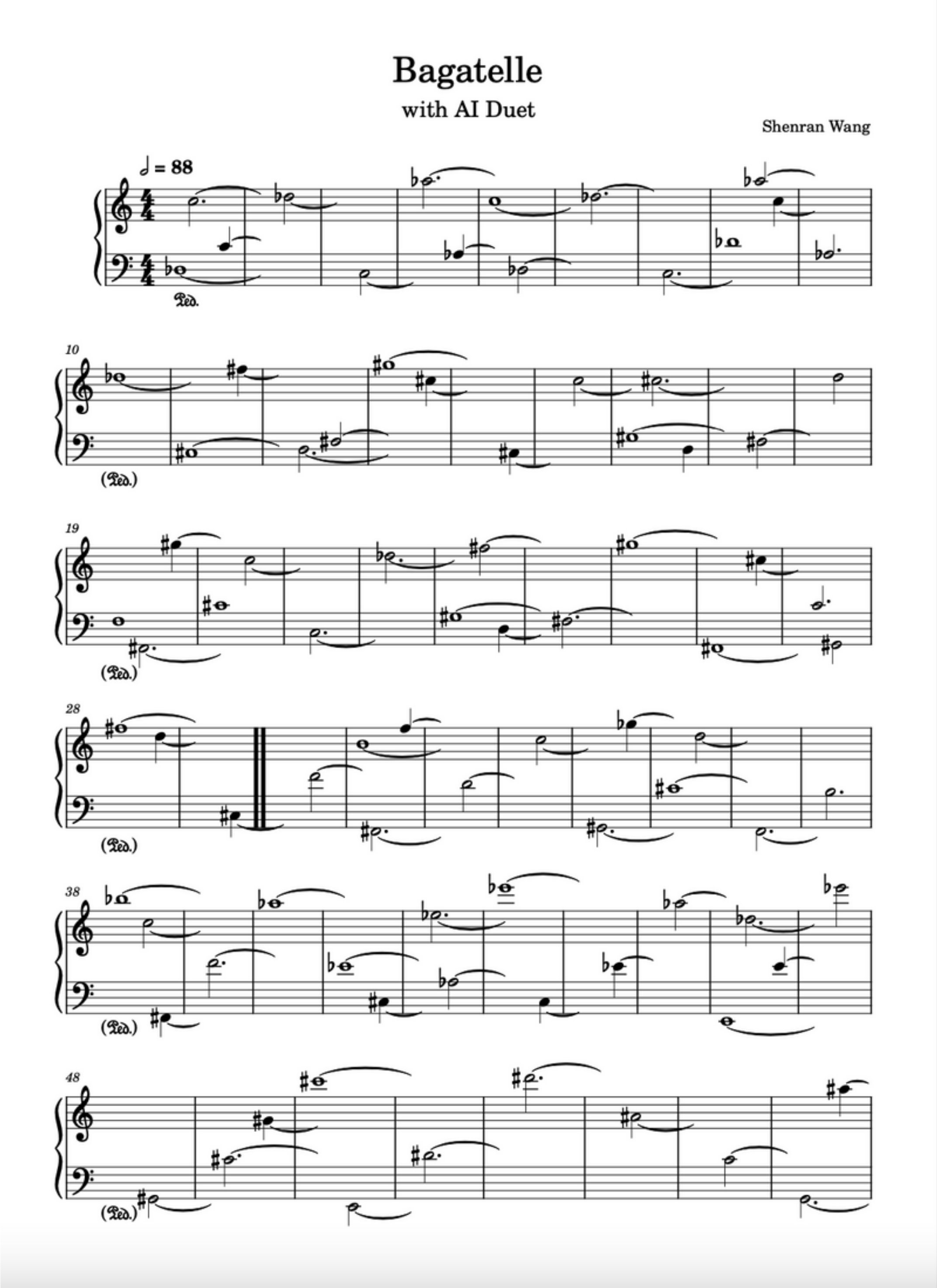


Figure 6 .First page of the *Bagatelle*. Music from the thick double barlines onwards is composed with the help of AI Duet.

The primary experiment in this project was using AI Duet as a tool in the composition process. I tried to see if the different RNN models would be capable of recognizing and imitating the fairly simple patterns from the beginning of the composition, while also further developing them. Based on earlier experiments in the course, AttentionRNN outperformed other models, but the disparity was not as apparent in this composition. In general, the sequences generated by the models were not that good. Although the pitches used in the sequences were mostly used in the inputted melody, the temporal aspect of the music was not captured at all well. Due to this, I decided to simply use the pitch classes as an indicator for which notes should appear after next as part of the pitch set on a larger scale within the music. I generated multiple samples using different input sequences as the composition progressed, and used elements from the generated sequences in the composition.

Noise to Noise, Eduard Mihai Tampu

The composition presented at the end of this course results from the exploration of new types of narratives and new ways of sound generation. The realised piece sees the use of Artificial Intelligence (AI), and in particular the generative model GANSpaceSynth as main technique for the generation of sounds which were then used for this piece.

The composition *Noise to Noise* by Eduard Mihai Tampu, tells the story of the lifecycle through the samples generated using GANSpaceSynth

It was initially difficult to find a narrative that used the sound materials generated from the different AI implementations to tell a story regarding the sounds themselves. However, through a careful analysis of the produced sounds, the attention was brought to the timbers generated by the GANSpaceSynth model in which the combination of sounds ranges from defined and recognisable tones to rough and noisy ones (Figure 7).

The concept was to delve further into the noise present in the system, not merely in the final stage, but as a means of examining the system's own story: from noise to sound, the learning process of an AI. The aim was to explore how the system begins to learn, how the model absorbs information from the provided dataset, and how it produces refined sound materials.

Several question originated from this analysis: What happens once the model is operational? What is the lifecycle of an AI system? Is there a finite lifespan, or will it live indefinitely?

These questions have a clear philosophical nature rather than a technical one. While it could be argued that the life of an AI model lies with the technology and data that represents it and make it possible, the interest of this study is different. The purpose of this composition is to examine an AI model's learning process while attempting to give it human traits.

The composition is therefore structured in three sections: the birth of the model, its life and its disintegration. The materials were generated using the GANSpaceSynth model trained with sounds coming from different sources such as previous compositions, field and saxophone recordings and different classical music recordings. The model was trained with 8 million images. Different epochs were then used inside the the Pure Data real-time audio synthesis environment to generate a collection of samples. The samples from different stages of model training were then selected and combined to obtain the composition.

The sounds obtained in the first stages are very similar in characteristics: very incoherent, mainly composed by long tones and with equal timber traits. These materials were used at the beginning of the piece, manipulated using Emit, a granular spectrogram synthesiser[14]. It allows the generation of pad materials, representing the idea of birth of the AI model. More sound materials were introduced as first movements of the new born model: like an infant that starts walking and exploring the surrounding world.

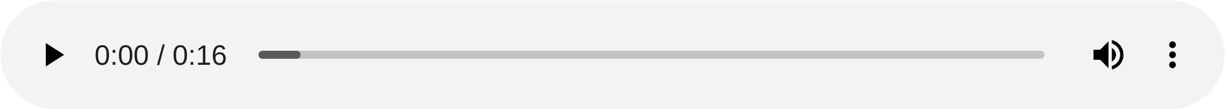
Samples from the stages between eight and 12 were selected and elaborated with different techniques. Mainly through the use of Granulator II, a granular synthesiser[15], the sounds are elaborated in order to obtain more cohesive materials that could be used as rhythmical/melodic materials. Portions of samples are used in order to retrieve transients used as kicks through the composition, in combination with a low bass sound. This stage represents the creative moment of the life of the AI model: as a child growing, fast, without realising, becoming an adult though living.

The last section was transitioned with a chaotic and complex mix of elements, still derived from the mentioned stages. Different elements can still be recognized, as out of focus memories. The transition leads to the disintegration of these elements. The pad material produced by the initial stages of life of the model were elaborated and used to realise a fast rhythmical element in combination with a fast kick, used to create a passe for the conclusion, creating tension and pushing forward to the conclusion. Materials processed with the Granulator II, was used in combination with effects (mainly Redux, a digital signal manipulator tool that uses down-sampling and bit reduction as main techniques[16]) in order to create the idea of disintegration.

Figure 7. Graphic notation of the piece Noise to Noise

Game Music Generation, Jackie Lin

My main composition blends together multiple short audio generated from randomly sampled latent vectors. My sample choices were inspired by the imagery and story line of Bloodborne (2015) [17], an action role-playing game with gore and Lovecraftian horror themes. The samples chosen all contained a deep bass note as anchor with some ethereal noise. One sample had a hair raising high pitch screech reminiscent of nails on a chalkboard. Other samples had slow but strong rhythmic components which added to the foreboding and dread. The result is a menacing dark composition full of anticipation for the start of a game scene.



Bloodborne-theme inspired composition is utilised by the samples generated with the GANSynth model

Additionally, I found a great sample again by randomly sampling the latent space, an uplifting flute audio with counterpoint in three voices. I matched this audio to Genshin Impact (2020) [18] another action role-playing game with an anime style and open world environment. The drawback of GANSynth is that it generates a mono output without awareness of multitrack decomposition. I couldn't decompose this three voice counterpoint sample, but it is the ideal example output of a tool for generating ready-to-use audio (Figure 8).

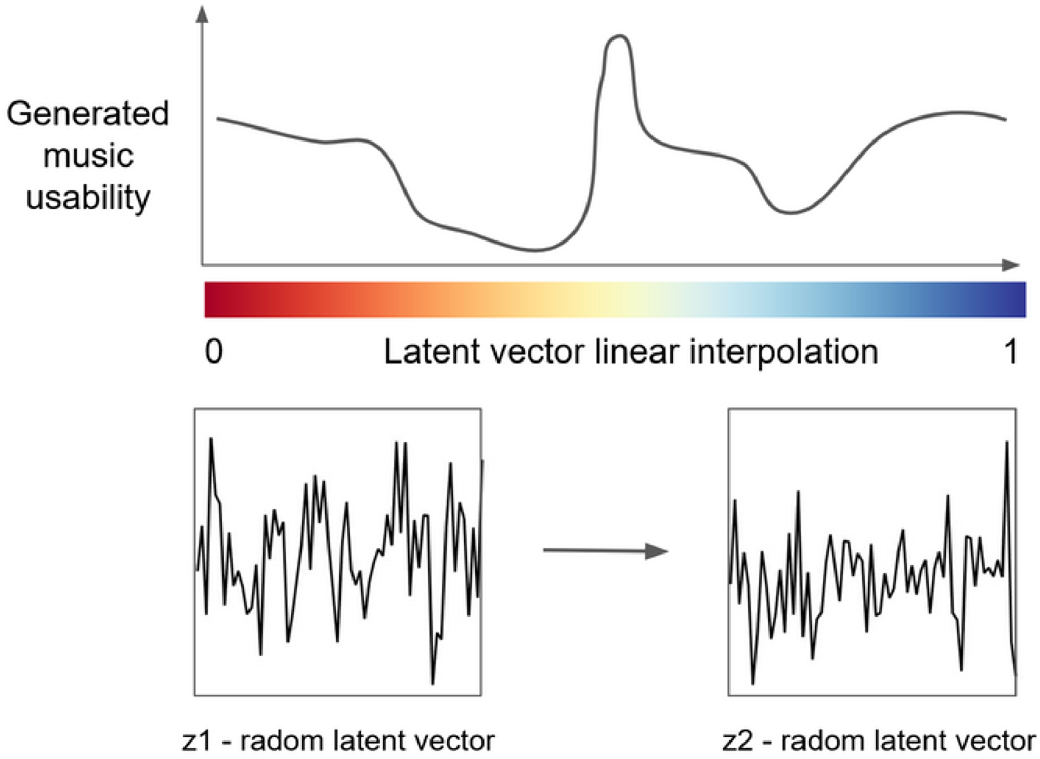


Figure 8. Linear interpolation between randomly sampled latent vector representations of a GANSynth model generates interesting and varied densities of usable audio.

Discussion

Throughout the course, the students were exposed to various deep learning models and tools for audio processing and generation. They were also provided with opportunities to experiment with these models through hands-on projects, where they applied deep learning to create new music compositions. The results of the students' projects demonstrate the effectiveness of the course in fostering their creative thinking in music production. The audio content produced by the students was diverse and rich, ranging from complex soundscapes to rhythmic compositions. This indicates that the students were able to use deep learning as a tool to push the boundaries of their creativity and explore new possibilities in music production.

> After doing this course I have a much greater understanding of how AI machine learning applications work and how important the data set you give them is for what it outputs in terms of variety of sounds that it models. I would have liked to have more time to really tweak the data set, removing and adding samples until I carved out the exact sounds I was looking for be led to new sounds I had not imagined.

MA student #5, Deep Learning with Audio course 2023

In addition to the quality of the audio content produced, the feedback received from the students through learning-diary surveys also indicates a high level of satisfaction and engagement with the course materials. The feedback revealed that the students found the course material to be relevant, interesting and challenging. They also reported that the hands-on projects were particularly helpful in consolidating their understanding of the concepts and models learned in the lectures.

At the same time, the learning diaries highlighted that the setup of the required components was still more challenging as there were too many steps and took much time then expected, similarly the training times of each model. Especially the challenges we faced with the storage space configuration with Microsoft Azure VM during the course as well as with the parameter settings regarding the number of images in the first GANSynth trainings, did not meet with some students expectation of rapid exploration of these models with their own datasets. In addition, students pointed out a particular request for deeper content on the theories and technical implementations behind these deep learning models as well as for expanding the contact teaching hours over a longer period of time; the current schedule of teaching was mentioned as very intense.

> I want also to underline what had me more interested of this technology and about the sounds that we were able to produce during this course. As raw material it was interesting to explore the timbers and materials that the system was able to generate.

MA student #7, Deep Learning with Audio course 2023

The importance of knowing other coding languages and practicalities of Linux operating system was mentioned as “new skills would need to be developed with new technologies”. Equally important views in student learning diaries on copyright and ethical issues that are raised on the data usage for generating these audio samples as well as dystopian technology views for replacing human musician with AI tools shows that the course also gave students an opportunity to critically approach to this rapidly developing AI music technologies.

> Apart from the final project of the course, the methods introduced in the class excited me to work more on that particular domain. Considering the huge hype toward popular AI tools in the image domain such as Dall-e or stable diffusion, the audio tools are not receiving the same attention that their predecessors already have.

MA student #11, Deep Learning with Audio course 2023

Conclusions

In this paper we presented our Deep Learning with Audio course that we started teaching since 2021 in graduate arts studies. The Deep Learning with Audio course has proven to be a valuable addition to our graduate arts studies program. The course content, which covers a range of deep learning models and their applications in audio and symbolic domains, has provided students with a solid foundation in this area. The hands-on exercises have allowed students to gain practical experience with these models. The three student project compositions that were presented in this paper demonstrate the creativity and ingenuity of our students, as well as their ability to apply the concepts they learned in the course to real-world problems. These projects are a testament to the effectiveness of the course in equipping students with the skills they need to tackle the challenges of the rapidly evolving AI music technologies. Finally, the student reflections on the course will be invaluable in refining and improving the course in future iterations.

Acknowledgments

This work is supported by the Academy of Finland (project 316549) and A!OLE funding. We acknowledge Miranda Kastemaa, Oskar Koli and Pierre Chouteau for their work in preparing the teaching materials and Mohammad Bakharzy for his support on Microsoft Azure cloud computing Virtual Machines.

Ethical Statement

The teaching content presented in this paper is based on the Deep Learning with Audio course, which includes course materials and examples of students' works. The course was conducted in accordance with ethical principles and standards of Aalto University. All materials used in the course were created by the authors or Sound and Physical Interaction - SOPI research group research assistants in which the third party materials are presented with their individual copyright license statements. Student project works were used with their explicit consent as they are also the co-authors of this article. Any personal or sensitive information shared in the students' reflections in the discussions section was anonymised to protect their privacy.

Footnotes

[1] https://research.google/teams/brain/magenta/ ↩

[2] https://github.com/tensorflow/tensorflow ↩

[3] https://github.com/librosa/librosa ↩

[4] https://github.com/pytorch/pytorch ↩

[5] https://ccrma.stanford.edu/courses/mus421n/ ↩

[6] https://explorecourses.stanford.edu/search?q=CS+470%3a+Music+and+AI&view=catalog&filter-coursestatus-Active=on&academicYear=20222023 ↩

[7] https://esling.github.io/teaching/ ↩

[8] https://steinhardt.nyu.edu/courses?search=artificial&op=Search&field_department_sgl_target_id=All&field_level_of_study_value=All&tid=All&field_course_units_min_value=All ↩

[9] https://college.berklee.edu/courses/mtec-345 ↩

[10] https://github.com/SopiMlab/DeepLearningWithAudio ↩

[11] https://github.com/SopiMlab/DeepLearningWithAudio ↩

[12] Error ↩

[13] https://github.com/SopiMlab/DeepLearningWithAudio/tree/master/06_rave ↩

[14] https://www.ableton.com/en/packs/inspired-nature/ ↩

[15] https://www.ableton.com/en/packs/granulator-ii/ ↩

[16] https://www.ableton.com/en/manual/live-audio-effect-reference/#24-34-redux-legacy ↩

[17] https://en.wikipedia.org/wiki/Bloodborne ↩

[18] https://genshin.hoyoverse.com/pc-launcher/?utm_source=EU_google_EUT2_search_20220719&mhy_trace_channel=ga_channel&new=1&gclid=Cj0KCQiApKagBhC1ARIsAFc7Mc7OVVKtjRmfDsnLh9TQ6Dc56dXPm562kBz7O9Y-TwBqYO88Ax-1tusaAkluEALw_wcB#/GI008 ↩

References

1. Mann, Y. 2016. AI duet. *Experiments with Google. See, https://experiments.withgoogle.com/ai-duet* ↩

2. Engel, J., Resnick, C., Roberts, A., Dieleman, S., Norouzi, M., Eck, D., & Simonyan, K. (2017, July). Neural audio synthesis of musical notes with wavenet autoencoders. In *International Conference on Machine Learning* (pp. 1068-1077). PMLR. ↩

3. Engel, J., Agrawal, K. K., Chen, S., Gulrajani, I., Donahue, C., & Roberts, A. (2019). GANSynth: Adversarial Neural Audio Synthesis. *International Conference on Learning Representations*. Retrieved from https://openreview.net/forum?id=H1xQVn09FX ↩

4. Engel, J., Hantrakul, L., Gu, C., & Roberts, A. (2020). DDSP: Differentiable digital signal processing. *arXiv preprint arXiv:2001.04643*. ↩

5. Tahiroglu, K., Kastemaa, M., & Koli, O. (2021, July). Ganspacesynth: A hybrid generative adversarial network architecture for organising the latent space using a dimensionality reduction for real-time audio synthesis. In *Proceedings of the 2nd Joint Conference on AI Music Creativity*. ↩

6. Mehri, S., Kumar, K., Gulrajani, I., Kumar, R., Jain, S., Sotelo, J., ... & Bengio, Y. (2016). SampleRNN: An unconditional end-to-end neural audio generation model. *arXiv preprint arXiv:1612.07837*. ↩

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

8. Grill, T. 2002-2005. py/pyext - python script objects for pd and maxmsp. In Web URL. http://grrrr.org/ext/py/. ↩

9. Grill, T. 2001-2015. flext - c++ layer for max/msp and pd (pure data) externals. In *Web URL*. http://grrrr.org/ext/flext/. ↩

10. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2020). Generative adversarial networks. *Communications of the ACM*, *63*(11), 139-144. ↩

11. Tahiroğlu, K., Kastemaa, M., & Koli, O. (2021). AI-terity 2.0: An Autonomous NIME Featuring GANSpaceSynth Deep Learning Model. *NIME 2021*. https://doi.org/10.21428/92fbeb44.3d0e9e12 ↩

12. Tahiroğlu, K., Kastemaa, M., & Koli, O. (2020, July). Al-terity: Non-rigid musical instrument with artificial intelligence applied to real-time audio synthesis. In *Proceedings of the international conference on new interfaces for musical expression* (pp. 337-342). ↩