Technische Universität Berlin

Quality & Usability Lab, Faculty IV

AI Song Generation

Harmonic song generation with the use of animal sounds

Final Report Advanced Study Projects 2024

Name	Matr.-Nr.	E-Mail
Mattia Bruno Stellacci	356439	matt@campus.tu-berlin.de
Lilly Josephine Wiegand	458883	lilly.j.wiegand@campus.tu-berlin.de
Sebastian Piotr Polak	382395	s.polak@tu-berlin.de
Deniz Mert Tecimer	487675	tecimer@campus.tu-berlin.de

Supervisors: Vera Schmitt, Premtim Sahitaj

March 17, 2024

1 Introduction

Despite recent progress in machine learning, music as a fundamental form of artistic expression and human communication poses an interesting challenge, as it's complexity, while elusive to machine learning systems is intuitive to human listeners. The music generation with both rule-based and data-driven methods was investigated in various aspects (14; 4; 29). In the light of it's success in domains such as conversational AI (text-to-text generation) (33), text-to-image generation (36) and translation Generative AI has also been used in the domain of music generation(7; 10) . However, most of the methods use existing instruments or human voices as the chosen timbre for the generation.

In this work, we aim to generate music which creates the illusion of being performed by animals. This task not only lacks a practical frame of reference, as such recordings practically don't exist and because most animals' sounds don't adhere to human musical formalism and are rarely *performed* in harmony. It is further complicated by the great diversity in animals' calls and sounds. As an example, most of the birds have two sets of syrinx to produce sound. Their biological structure allows them to create complex, at times polyphonic, sounds. Some insects make sounds by vibrating body parts. Most mammals have a similar vocal cord structure as humans (12) but can vary in volume and register. For lack of explicit psycho-acoustic models to generate such sounds realistically, we therefor attempt to re purpose existing *tone-transfer* models, trained to transfer audio from one instrument to resemble another.

Beyond experimenting with the synthesis of such audio, we investigate how music can be composed for an ensemble of animal voices, making use of *MIDI*(26), a common protocol/file-format used by digital instruments and DAW software. We aim to constrain the compositions to suit the limitations of the voices performing them, while further adhering to the rules of (western) musical harmony and composition.

In this report, related works are presented and our methodology is motivated, followed by both successful and failed experiment results on using animal sounds as "instruments" to eventually create a song by combining them.

2 Related Work/Literature Review

Generative AI has shown great success and has been widely adopted in recent years with the emergence of products such as ChatGPT (33). In the field of audio synthesis, even though electronic music synthesizers were created in the 1960s and 1970s (35), human speech synthesis had its early attempts around 200 years ago (14).

In 2016, WaveNet (40) was developed based on PixelCNN architecture (41) as an audio generative model. Its success inspired other works and made it a baseline to compare the performance of new models (9; 23; 7; 6; 20). Recent works often adopted variational auto-encoders (VAE) (25) and generative adversarial nets (GAN) (15) due to their efficiency and flexibility. WaveNet focuses on human speech generation and was applied to the music domain by (11), and they compared the results with a spectral autoencoder baseline. Similar to WaveNet, (7) as another successful audio generation model, can generate music in the raw audio domain.

Models such as (23), (21), (44), and (20) ease the problem by working on it in a lower-dimensional space. The symbolic approach often features Mel Frequency Cepstral Coefficients (MFCC), Mel-Spectrogram or Rainbowgram calculated via the Short-Time Fourier Transform (STFT)(16) or Constant-Q Transform (CQT)(5) which transforms the data from the time domain to frequency domain. However CQT is more suitable for music representation, the reconstruction of a waveform from CQT requires phase information, which gets lost, unlike STFT, due to the filters applied. For instance, (20) uses another model to approximate these values. They first generate rainbowgram from the raw audio and use CycleGAN (46) to reconstruct the image. After reconstruction, they generate the raw audio from the generated rainbowgram. Similarly, (44) uses the same approach on CQT representations and (21), (23) on Mel-spectrograms.

2.1 Symbolic composition

The interest in tackling the challenge of composing music with machine learning methods goes back to the 1980s when first experiments with computer based compositions where made and started of with approaches that can be classified as part of algorithmic composition.(31) Over the last two decades the

amount of research and and released models that apply deep learning models to solve this problem has increased substantially and made it the dominant approach to machine learning based music composition. Currently existing approaches can be divided into symbolic and non-symbolic, by the way the input and output is represented. The non-symbolic approach aims to generate raw audio or spectrograms and therefore poses a more complex challenge due to the high dimensionality of the data, compared to the symbolic approach. A prominent example is Jukebox model, which combines VQ-VAE and autoregressive Transformers. (7) For the symbolic approach music is represented in a format that consists of discrete events such as MIDI (26), a protocol that originally aims to enable communication between music instruments and devices, or MusicXML (mus). There is a variety of Neural Network Architectures that have been applied to the problem of symbolic music composition such as Recurrent Neural Networks (RNNs) (e.g. DeepBach in 2017 (17), the Google's Magenta Melody RNN models in 2016 (43)), Variational Auto-Encoders (VAEs) (24) (e.g. MusicVAE in 2018 (37)), Generative Adversarial Networks (GANs) (39) (e.g. MidiNet in 2017 (45), MuseGAN in 2017 (8)) and the Transformer Architecture (42). The Transformer Architecture is able to maintain context and capture long-range dependencies because its self-attention mechanism which is why is it so well suited for the task of symbolic music generation and NLP tasks. There are a lot of similarities in the nature of challenges from within the NLP domain and the task of music composition, if we consider music to be a form of language with a structure, rules and patterns consisting of discrete sequential events that can be learned to predict. Both domains pose the challenge of maintaining and considering multiple contexts at the same time, since musical structures and events happen along different time scales in the scope of motifs and phrases similar to natural language. Because of those similarities and the significantly good performance of Transformers on NLP problems many successful attempts of using the sequence-to-sequence architecture of an Transformer in order to predict sequences of musical events and by that composing music have been made. (22) (19) (34) A prominent example is MuseNet (34), which uses the unsupervised technology of the GPT-2 transformer for next-sequence prediction. It was made available to interact with online and was able to generate up to 4-minute musical compositions with 10 different instruments. In December 2022 this functionality disappeared without information about when or if it will be made available again. Another example is the Music Transformer by Huang et al. (19), which uses an improved attention mechanism called relative attention for music generation. While there are also newer models that use Denoising Diffusion Probabilistic Models (DDPMs) (30) (18), we decided to focus on an approach that uses a Transformer architecture. In our search of existing open-source Transformer models that are applied to music generation based on MIDI files we came across a GitHub Repository (3) by Aleksandr Sigalov, who creates and maintains multiple models, pre-trained models and a midi processing library that are able to generate music such as the Los Angeles Music Composer and the Giant Music Transformer of which we implemented the Giant Music Transformer.

2.2 Tone Transfer

Symbolic representations are also often used in timbre transfer, as they contain style information of the instrument. Jukebox (7) contradicts using the symbolic approach as it limits the generation capability of the model. Rave (6) used a two-phase approach to train a VAE with raw audio, in which first the encoder and decoder are trained, followed by adversarial training for fine-tuning the decoder. Their model is efficient for real-time processing on a CPU, yet data-hungry, and requires around 3 hours of recordings and a long training duration. Even though the symbolic approach limits the generation capabilities, (10) introduced the Differentiable Digital Signal Processing (DDSP) library and explored the capabilities of combining classical signal processing tools with modern automatic differentiation software "Tensorflow" (2). They propose a model that utilizes oscillators to overcome the phase information loss found in many of the aforementioned models (i.e. (9), (40)) while employing deterministic autoencoders. Those are less data-hungry, simpler, and retain numerous advantages the Variational Autoencoders (VAEs) (13). Oscillators, also known as vocoders or synthesizers, require hand-tuned parameters with expert knowledge - however, their expressiveness is bound by the given analysis of these parameters. There are some methods showing success in extracting them, yet not allowing the gradients to flow through the synthesis procedure. To address this limitation, DDSP advocates for an end-to-end approach and employs Digital

Signal Proessing (DSP) components, such as oscillators, envelopes, and filters, as feed-forward functions to increase efficiency. Remarkably, DDSP demonstrates the ability to convert one instrument into another, requiring only 10-minute monophonic recordings of the target instrument (10).

3 Methodology

Given only the task of composing "[...]a song consisting of multiple voices that follows harmonic principles." it was first neccessary to clarify what this means specifically, as the role of the animal voices within a composition is not obvious. We decided to create a composition consisting of only animal voices, with harmony being created by their relative pitches, the animals would therefor *sing* the piece. We opted not to include other instruments or to make use of animal sounds percussively, though this would have ostensibly fit the task as well.

Informed by the observation, that the creation of music can be separated into two separate sub-tasks (and frequently is in classical/western musical tradition[1]) we divided the problem in two: composition and performance/audio synthesis. This separation was useful, because it favors a division of concerns among collaborators, but also because it increased the chances of success, as we could apply simplifying assumptions/constraints in either domain. These simplifications will be elaborated upon in their respective sections. Further, we could re-purpose existing models, available for the sub-problems.

Before considering these tasks individually, we will briefly touch upon a series of observations which guided our work.

- **Monophony**: With some exceptions (e.g. some bird voices), most animal voices are monophonic, while many instruments are polyphonic (e.g. any instrument with multiple strings such as a piano or a guitar). In an effort to produce realistic animal renditions, it would make sense to limit the voices to be monophonic.

- **Harmony**: While monophonic voices are *simpler* than polyphonic ones, the wider task of composition is complicated, because a piece's harmony at any given time emerges as the interplay of multiple voices and cannot be created by simply including chords for a single instrument. Any system generating compositions would need to be aware of this context.

- **Percussion/Rhythm** Though melody and harmony play an important role in western music, arguably one of the most universal and fundamental aspects is rhythm. While some animals produce unpitched/percussive sounds which could serve as percussion, these are very hard to identify without context. [2]

3.1 Composition

Symbolic music composition, i.e. the generation of a formal structure of a composition without the actual performance of the piece (comparable to writing sheet music), is very related to the far more common and understood task of NLP. Models can (and commonly are (43)) conveniently be trained on next-token prediction in the same unsupervised scheme that LLMs are trained in (32).

Building on the long-standing MIDI standard, originally conceived to enable digital instruments to communicate via a standardized protocol but soon adapted as a serial file format storing symbolic music representations, existing models take advantage of a wealth of public domain MIDI data available online (32). While large parts of the task of transferring sheet music to a well-defined unambiguous symbolic serial representation is therefor solved by MIDI, a preliminary research revealed, that most of the existing models' complexity and domain-specificity lie in the embedding of the MIDI data. We decided to base our first experiments on a model available on GitHub called *Allegro Music Transformer*[3] but later, upon consulting the author of the repository about some technical obstacles, switched to the *Giant Music*

[1] e.g. the role of a composer vs. that of an instrumentalist/conductor

[2] e.g. a woodpecker is identified as such by the fact that it is heard in a forest/in nature and it's sound is not immediately obvious as such when taking out of context

[3] `https://github.com/asigalov61/Allegro-Music-Transformer`

Transformer[4]. Both models were developed as general purpose composition models, capable of generating compositions in varying styles of music and their code is made available by the author as Google Collab notebooks and on GitHub .

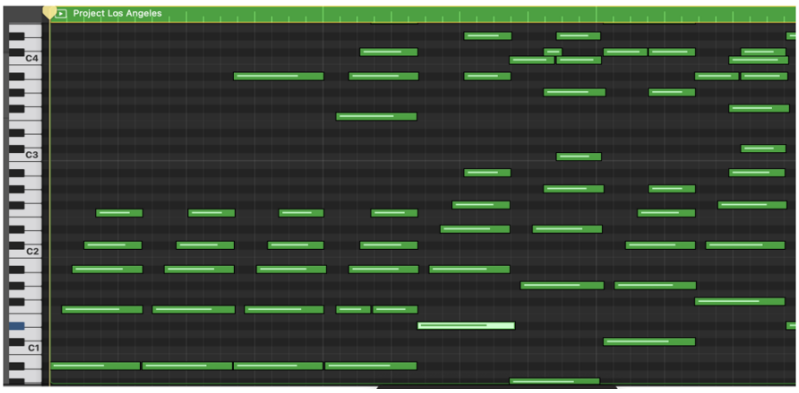


Figure 1: Example visualization of a MIDI-File

Considering the observations mentioned in the previous section and in an effort to simplify the task as much as possible, we decided to use choral compositions. A series of factors influenced this decision:

- All requirements resulting from said observations are met
- Choral compositions are often limited to the 4 most common vocal registers (Alto, Soprano, Tenor, Bass).
- Classical choral compositions[5] observe fairly strict formal structure which an ML model would hopefully learn. Examples of such structure include the rules of *Four-part Movement* and Counterpoint.

3.1.1 Challenges

First experiments were conducted using the unaltered *Giant Music Transformer*, with it's default preprocessing pipeline and training it from scratch with a set of approx 150 MIDI files of the collected choral works of *Johann Sebastian Bach*[6]. Unfortunately, the models in their default configuration turned out to be very large and some parameters needed to be changed to suit the available hardware's VRAM constraints. It further soon became apparent, that the model would not learn to restrict itself to 4, monophonic voices, simply based on the training data, but that some modification to the preprocessing pipeline would be required.

Analogous to the observation that the common use of vanilla transformers in notable recent composition models reduces their significant diffences to how they embed their data, all major challenges encountered with the symbolic composition were either directly related to the MIDI file-format and it's intricacies or to the barely documented embedding/preprocessing pipeline of the *Giant Music Transformer*. Though the large size of the model also required some attention/modification during training, the major challenge in it's size was not the memory requirement *per se*, but rather the fact that the necessity for this size derives from the model's intended ability to learn many different styles of music, performed by up to 16 different MIDI instruments and with varying velocities. This formulation exhibits interesting behaviour when trained on powerful hardware with a large dataset, but was not warranted by the limitations we set. Sticking to the default embedding space of 22K tokens and training with our limited dataset resulted in smaller models producing degenerated outputs, with spurious bursts of activity on all 16 instruments and clearly underfitting the task.

Most of the development time was spent "reverse engineering" the undocumented embedding pipeline in an effort to make modifications which meaningfully reduce the dimensionality of the feature space.

[4] `https://github.com/asigalov61/Giant-Music-Transformer`
[5] especially early Baroque works
[6] `https://www.tobis-notenarchiv.de/bach/07-Choraele/02-Vierstimmig/`

At times these efforts where complicated by confusing aspects of MIDI [7] and at times by the seemingly arbitrary assignment of integers in the embedding code, which maps various values retrieved from MIDI events to numbers. In hindsight, it can be said that it would probably have made more sense to start *from scratch* building a custom pre-processing pipeline around popular and well documented python libraries for MIDI such as `music21` [8]. As the simplicity of the underlying sequence-to-sequence model was not immediately apparent and hoping to not reinvent the proverbial wheel, a substantial amount of time was spent grappling with the *Giant Music Transformer*. Unfortunately even the active support by it's author via *GitHub Issues* didn't solve the issue, that the default Giant Music Transformer's embedding pipeline doesn't pass a *roundtrip*-test, i.e. transferring a MIDI file to it's embedding space and back, expecting an identical (or at least very similar) MIDI file. Rather the velocities and durations of notes are significantly altered and at times they are mapped to other instruments.

Irrespective of the difficulties encountered, we eventually succeeded in developing a simplified, custom embedding pipeline which allowed us to reduce the embedding's dimensionality (approx. 22K $\rightarrow$ 13K tokens). The smaller embedding space enabled us to reduce the model parameters(approx. 300m $\rightarrow$ 4m tokens) and eventually a model was trained which produced satisfactory compositions, utilizing 4 voices in harmony.

3.1.2 Data sourcing and processing

Having started out with only compositions by Johann Sebastian Bach (in an effort to *start small*) we realized, that more data would benefit the training process. Constrained by our ability to effectively limit the dimensionality of our model and embedding space on account of issues described in 3.1.1 we therefor decided to extend our dataset to train a more robust model with more data. We proceeded as follows:

- We wrote a script to scrape *ChoralWiki*[9], an online database of public domain sheetmusic and MIDI files for choir. Approx. 22K files were downloaded.
- The files were filtered, discarding all but the ones which contain exactly 4 channels(assuming these to be the ones containing only the 4 common choral voices with no instrumental accompaniment).
- The files were analyzed for their key and transposed to the key of C-major/A-minor(depending on their mode).
- The channels were ordered and assigned to the range $[0, 4]$.

3.2 Audio synthesis

To help the animals "sing" the composed music, we've chosen to use a method called tone transfer, specifically a Differentiable Digital Signal Processing Model (DDSP) ((10)). A challenge in data collection we faced was, that recordings of animal sound data often contain background noises, like the wind and other ambient sounds, like the human speech or the sound of other animals. As existing datasets were inadequate for our needs, we aggregated diverse animal sounds from publicly available sources and created our own dataset. Thus, different animal sounds were collected and manually pre-processed from the YouTube platform[10] and the XenoCanto database[11]. At the time of our research, to our knowledge, no existing model targeted precisely our objective that would directly fit our use case of timbre transfer of animal sounds on a song. Consequently, we treated animal sounds akin to musical instruments. Given the data shortage of relavant data, we adopted the DDSP model for our experiments, which allows training with data that is at least 10 minutes long and can generate audio with the timbral characteristics of the instrument it was trained on.

[7] e.g. the inconsistent use of channels/patches to encode instruments
[8] `https://github.com/cuthbertLab/music21`
[9] `https://www.cpdl.org/wiki/index.php/Main_Page`
[10] `https://www.youtube.com/`
[11] `https://xeno-canto.org/`

The DDSP model consists of three encoders and a decoder. The fundamental frequency, style parameters (latent codes), and the loudness of the input are encoded through the encoder layer, and used by the decoder to generate 101 harmonics guided by the fundamental frequency and the filtered noise. Afterwards, reverb is added to complete the generation. Spectrograms are 2D representations of a signal and combine time and frequency domains. Colors or brightness represents the frequency components of each time frame. They are often used as an image-like interpretation of the audio, which makes it possible to be processed by Convolutional Neural Networks(CNNs) (27). The training goal of the model is defined by minimizing the Multi-Scale Spectrogram Loss, which calculates the difference between spectrograms of the input and output (see Figure 2) (10).

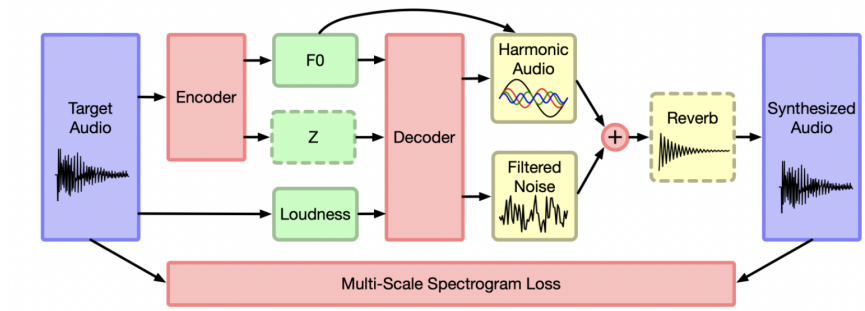
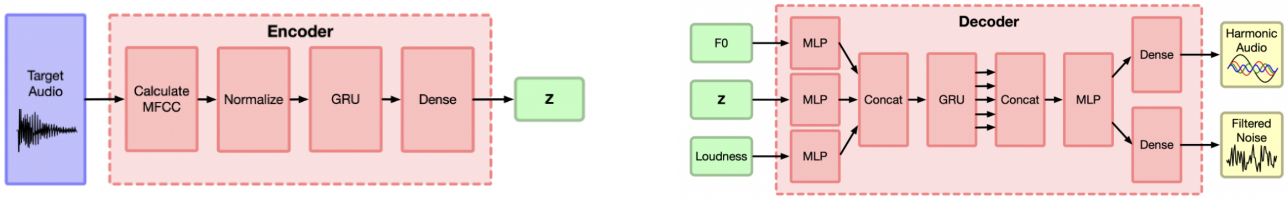


Figure 2: DDSP Model Architecture (10)

The encoding is achieved through three encoders. First, the loudness of the input is calculated via the A-weighting method from raw audio. Then, using STFT they generate the spectrogram. This spectrogram is fed to a ResNet model, which is trained with the main model, to obtain fundamental frequency. 30 MFCC parameters, representing the style of the input audio, are derived from this spectrogram and mapped into 16 latent variables (see Figure 3a) (10).



(a) Latent Encoder Architecture (10)

(b) Decoder Architecture (10)

Figure 3: DDSP Architecture Components

The outputs of the encoder are fed into the decoder to obtain the amplitudes of the harmonics given the fundamental frequency and the transfer function of the FIR filter. The generated harmonic audio, filtered noise, and reverb are combined using an additive synthesizer to yield the main output (see Figure 3b) (10). Figure 4 taken from (10) shows the effect of each synthesizer parameter determined by the model.

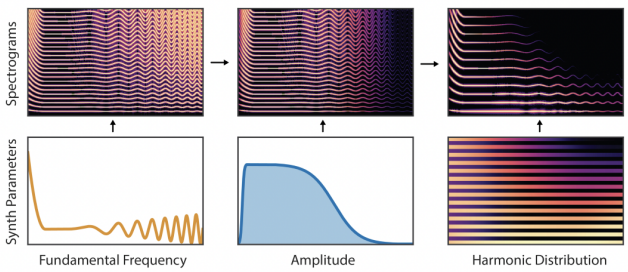


Figure 4: Diagram of the Additive Synthesizer component (10)

We experimented with the DDSP model using default parameters with Adam optimizer in an iterative workflow for each animal. First, the data was collected from the resources mentioned earlier. Then in some experiments, the data was cleaned up, i.e. through the removal of the noise and irrelevant sounds. The model was trained for around 50.000 steps depending on the respective experiment and evaluated using the free music sequencer Helio[12], which supports VST3 plugins (in our case the DDSP-VST plugin, see Figure 5), using a template MIDI file. This MIDI file consists of short and long notes, that cover one octave and can be shifted to find the best usage for the specific trained model. The DDSP-VST plugin features various settings, bringing more control on the generated output. The DDSP-VST plugin contains a DDSP Effect component, which runs the model on an audio input, and a DDSP Synthesizer, which applies the tone transfer on a given MIDI. The scripts used for remote and local training are shared[13][14]. Due to dependency problems, the local training script runs on docker.

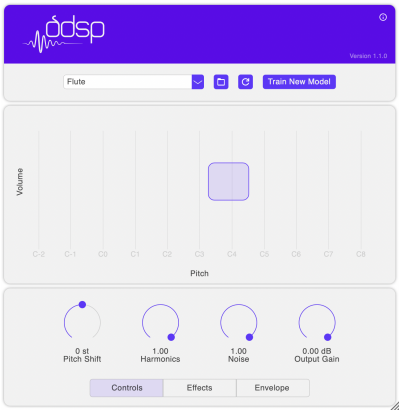


Figure 5: DDSP-VST Plugin (10)

4 Analysis and discussion

4.1 Symbolic Composition

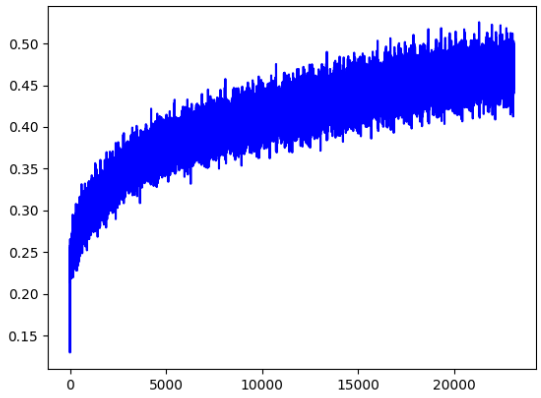
Our attempts to develop an AI that composes choral music resulted in a model which produces plausible, at times erratic compositions. Like natural languages, music is a sequential form of expression, with local and global context within a given work. Re-purposing *standard* sequence-to-sequence architectures, successful in the field of NLP, proved to be a viable way of generating MIDI sequences.

We include here the training and validation loss visualizations of our final model. For lack of a standard benchmark for this specific task, we can only provide this data and sample outputs for subjective evaluation.

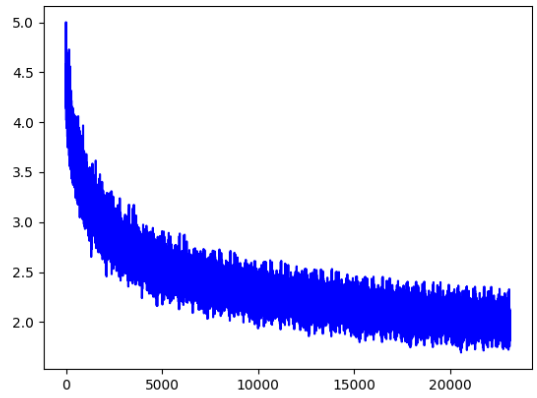
4.2 Tone Transfer

In the Google Colab notebook shared for training the autoencoder (28), it is suggested that the model performs well when the total loss reaches values around 4.5 or 5.0. However, in our experiments, this was not valid for every use case, making the loss irrelevant for the performance examination of the model. As shown in the spectrograms in Figure 8a, the output contains the harmonics representing the elephant sounds and still maintains the melodic content of the target sound. This state is achieved when the loss lies around 7 contradicting the recommendation (see Figure 7a). In addition, the spectrograms of the different successful and failed experiments are presented in Figure 8b and Figure 8c.

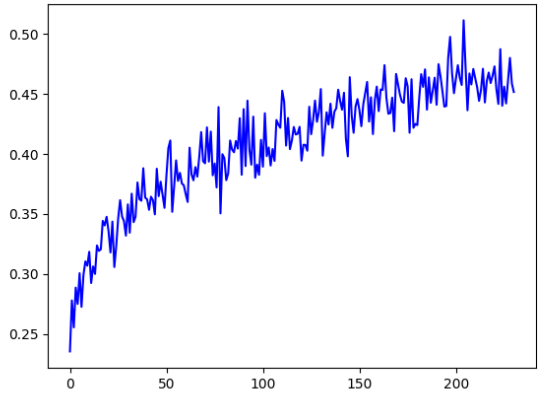
[12] `https://helio.fm/` - (38)
[13] Online Colaboratory - `https://colab.research.google.com/drive/1t6L_4cC1OylOLLPoFHGRU123IID8zJRt`
[14] Docker Colaboratory - `https://colab.research.google.com/drive/1UE2zTZhlfe-CDYlg5AK4oqf55ksLXc7H`



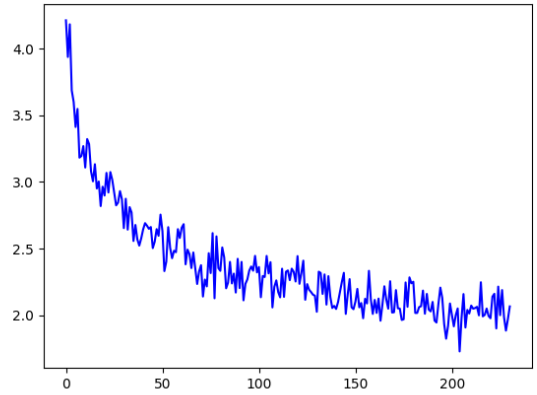
(a) Training accuracy graph



(b) Training total loss graph

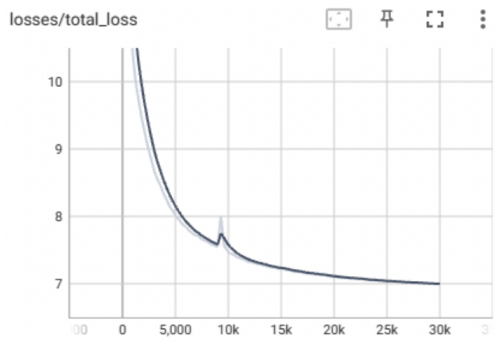


(c) Validation accuracy graph

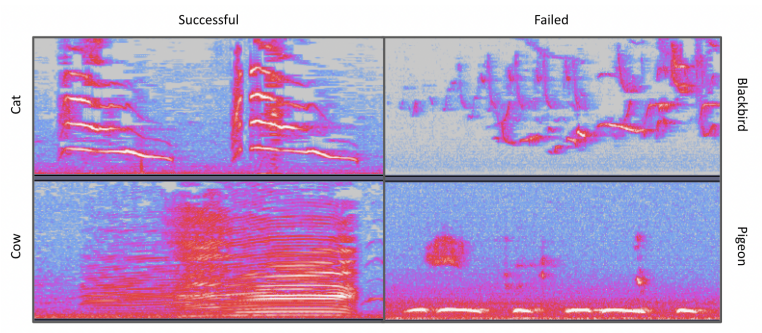


(d) Validation total loss graph

Figure 6: Visualization of the final model's training stats (accuracy and loss) of the Symbolic Composition



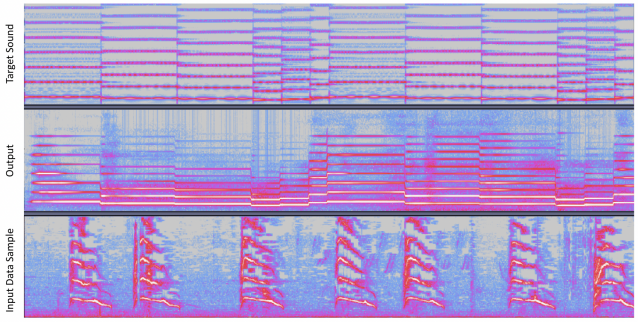
(a) The DDSP Training Loss on Elephant Sounds



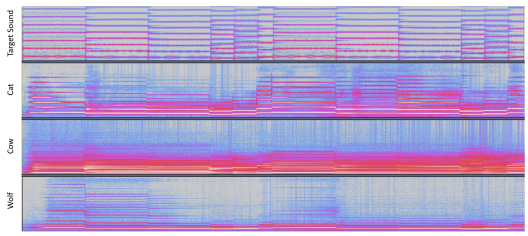
(b) Comparison of Input Data of Successful and Failed Experiments

Figure 7: Training loss and comparison of experiments of the Tone Transfer

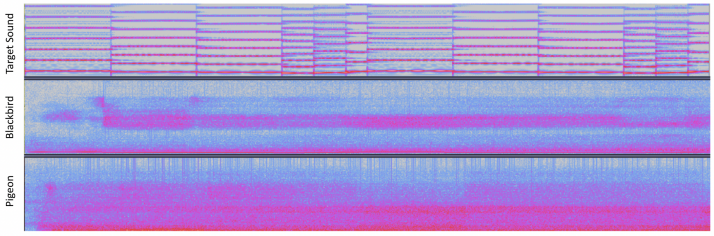
As the training goal was set to minimize the spectrogram difference, we further examined the input data spectrograms to investigate the failed results. As shown in Figure 7b, we compared the spectrograms of input samples of both successful and failed experiments. While successful experiment samples contain a harmonic line stack, the failed samples lack this feature which is against the training goal of the model. The outputs show that the experimented mammals (i.e. cat, cow) share a similar structure in terms of harmonics and the respective experiments performed relatively better than the experiments with birds (i.e. blackbird, pigeon, etc.).



(a) Timbre Transfer with Elephant Sounds



(b) Successful Timbre Transfer Outputs



(c) Failed Timbre Transfer Outputs

Figure 8: Spectrograms of the Timbre Transfer

5 Discussion

5.1 Symbolic Composition

Unfortunately, within the scope of this project it was only possible to reach a basic working model and most interesting, domain-specific limitations we considered, could not be made to meaningfully influence our results. Too much time was spent getting familiar with MIDI and symbolic music representations. While the current results mostly adhere to principles of harmony and rhythm, they exhibit little to no imitation of "artistic intent" or dramatic development over the course of a sequence.

It would have been very interesting to keep developing, as it is our opinion that with current models, the task as outlined and the available data, significant improvements in subjective "musicality" could be achieved.

5.2 Tone Transfer

The DDSP model proposed by Magenta (10) could manage to transfer the timbre of some animal sounds on a given MIDI. However, we acknowledge the limitations of the model mentioned by the authors. The model requires a minimum of 10 minutes of clean, monophonic recordings. As they used in their experiment, the input is expected to come from the same or similar distribution. Varieties in recordings, such as background noise, microphone specifications, etc. might affect the model performance. Animals have different biological structures to produce unique sounds and different animals may have a different note range (12). Obtaining clear recordings in the wild is harder than recording a musician in a controlled

studio environment as they play different songs covering many notes with different lengths. Before choosing animal instruments to develop, the candidate animal sound recordings should be examined concerning the model's capabilities. Thus, the currently available data and technology, constrain further improvement in achieving song generation with recognizable animal sounds.

6 Conclusion

The song generation task using animal sounds was divided into two subtasks, composition (MIDI generation) and audio synthesis (Tone Transfer). The Giant Music Transformer, a publicly available transformer-based autoregressive model, was modified and trained for the MIDI generation with 4 channels, which are then mapped to different DDSP models trained for the tone transfer. The DDSP model transfers the harmonics learned from the input data on the target sound guided by the fundamental frequency extracted through a ResNet architecture trained alongside the model. The generated MIDI file was imported on the Helio music sequencer, which supports the DDSP-VST plugin standard containing the trained DDSP model. Each channel was connected to a different model representing different animals to generate a song with 4 animal voices. With the faced limits and challenges, a MIDI-composition was successfully generated, which was mapped to different trained models of the timbre transfer.

There are not enough publicly available clean animal sound recordings that fit the model's needs and capabilities. The DDSP model performs better when the harmonic structure is visible on the input. Mammals share a similar structure for sound generation and on their input harmonics spectrograms were easier to identify. However, there is no one-fit-for-all solution on timbre transfer using animal sound recordings and the dataset must be examined before choosing animal samples.

Regarding the sound generation, replacing the Differentiable autoencoder (DAE) with a Variational autoencoder (VAE) could improve performance as denoted by (10). Increasing the sampling rate may help the model capture more instances from the data that contain short notes (i.e. bird chirping). However, this improvement could lead to an increase in the data hunger of the model. The goal of the model can be precisely designed for this task via a more suitable loss definition, which may require a brand new model to be engineered to focus on this task. In addition, a model that catches the melody of the sounds of a certain animal would be beneficial, as it adds structure and coherence to the composition. As DDSP is primarily designed for synthesizing musical instrument sounds, it does not capture the intricate nuances and complexities of animal vocalizations effectively. Animal vocalizations often exhibit a wide range of non-musical characteristics, including complex patterns, irregular rhythms, and diverse timbres. Knowing those limitations, we hoped to overcome those challenges, by choosing animals that represent instruments most likely. Sadly this only worked to a certain extent and is reflected in our final generated song.

References

[mus] Musicxml. `https://www.musicxml.com/`.

[2] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., and Zheng, X. (2016). Tensorflow: A system for large-scale machine learning.

[3] Asigalov, A. (ongoing). asigalov61's github repository. Accessed: March 17, 2024.

[4] Beauchamp, J. (2007). *Analysis, Synthesis, and Perception of Musical Sounds*.

[5] Brown, J. (1991). Calculation of a constant q spectral transform. *Journal of the Acoustical Society of America*, 89:425–.

[6] Caillon, A. and Esling, P. (2021). Rave: A variational autoencoder for fast and high-quality neural audio synthesis.

[7] Dhariwal, P., Jun, H., Payne, C., Kim, J. W., Radford, A., and Sutskever, I. (2020). Jukebox: A generative model for music.

[8] Dong, H.-W., Hsiao, W.-Y., Yang, L.-C., and Yang, Y.-H. (2017). Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment.

[9] Défossez, A., Zeghidour, N., Usunier, N., Bottou, L., and Bach, F. (2018). Sing: Symbol-to-instrument neural generator.

[10] Engel, J., Hantrakul, L. H., Gu, C., and Roberts, A. (2020). Ddsp: Differentiable digital signal processing. In *International Conference on Learning Representations*.

[11] Engel, J., Resnick, C., Roberts, A., Dieleman, S., Eck, D., Simonyan, K., and Norouzi, M. (2017). Neural audio synthesis of musical notes with wavenet autoencoders.

[12] Forinash, K. and Christian, W. (2012). Sound: An interactive book.

[13] Ghosh, P., Sajjadi, M. S. M., Vergari, A., Black, M., and Schölkopf, B. (2020). From variational to deterministic autoencoders.

[14] Gold, B. (1978). Historical background. *The Journal of Criminal Law*, 42:109 – 121.

[15] Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial networks.

[16] 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.

[17] Hadjeres, G., Pachet, F., and Nielsen, F. (2016). Deepbach: a steerable model for bach chorales generation. *ArXiv*, abs/1612.01010.

[18] Ho, J., Jain, A., and Abbeel, P. (2020). Denoising diffusion probabilistic models. *ArXiv*, abs/2006.11239.

[19] Huang, C.-Z. A., Vaswani, A., Uszkoreit, J., Shazeer, N., Simon, I., Hawthorne, C., Dai, A. M., Hoffman, M. D., Dinculescu, M., and Eck, D. (2018). Music transformer.

[20] Huang, S., Li, Q., Anil, C., Bao, X., Oore, S., and Grosse, R. B. (2023). Timbretron: A wavenet(cyclegan(cqt(audio))) pipeline for musical timbre transfer.

[21] Jain, D. K., Kumar, A., Cai, L., Singhal, S., and Kumar, V. (2020). Att: Attention-based timbre transfer. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–6.

[22] Jiang, J., Xia, G. G., Carlton, D., Anderson, C. N., and Miyakawa, R. H. (2020). Transformer vae: A hierarchical model for structure-aware and interpretable music representation learning. *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 516–520.

[23] Kim, J. W., Bittner, R., Kumar, A., and Bello, J. P. (2018). Neural music synthesis for flexible timbre control.

[24] Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *CoRR*, abs/1312.6114.

[25] Kingma, D. P. and Welling, M. (2022). Auto-encoding variational bayes.

[26] Lehrman, P. D. and Tully, T. (2017). What is midi?

[27] Lonce, W. (2017). Audio spectrogram representations for processing with convolutional neural networks. *CoRR*, abs/1706.09559.

[28] Magenta Team (2024). train_autoencoder.ipynb - colaboratory. `https://colab.research.google.com/github/magenta/ddsp/blob/main/ddsp/colab/demos/train_autoencoder.ipynb#scrollTo=fT-8Koyvj46w`. Accessed: 2024-03-09.

[29] Martinez-Zorrilla, D. (2008). *Synthesizers: A Brief Introduction.*

[30] Mittal, G., Engel, J., Hawthorne, C., and Simon, I. (2021). Symbolic music generation with diffusion models. *ArXiv*, abs/2103.16091.

[31] Nierhaus, G. (2008). Algorithmic composition: Paradigms of automated music generation.

[32] OpenAI (2019). Musenet.

[33] OpenAI (2023). Gpt-4 technical report.

[34] Payne, C. (2019). MuseNet. Blog post.

[35] Pekonen, J. and Välimäki, V. (2011). The brief history of virtual analog synthesis.

[36] Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., and Sutskever, I. (2021). Zero-shot text-to-image generation.

[37] Roberts, A., Engel, J., Raffel, C., Hawthorne, C., and Eck, D. (2018). A hierarchical latent vector model for learning long-term structure in music. *ArXiv*, abs/1803.05428.

[38] Rudenko, P. (2024). Helio - open-source music sequencer. `https://helio.fm/`. Accessed: 2024-03-16.

[39] Treat, I. and Yoon, J. (2018). Generative adversarial nets.

[40] van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. (2016a). Wavenet: A generative model for raw audio.

[41] van den Oord, A., Kalchbrenner, N., Vinyals, O., Espeholt, L., Graves, A., and Kavukcuoglu, K. (2016b). Conditional image generation with pixelcnn decoders.

[42] Vaswani, A., Shazeer, N. M., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In *Neural Information Processing Systems.*

[43] Waite, E. (2016). Generating long-term structure in songs and stories.

[44] Wu, Y., He, Y., Liu, X., Wang, Y., and Dannenberg, R. B. (2023). Transplayer: Timbre style transfer with flexible timbre control. In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5.

[45] Yang, L.-C., Chou, S.-Y., and Yang, Y.-H. (2017). Midinet: A convolutional generative adversarial network for symbolic-domain music generation.

[46] Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. (2020). Unpaired image-to-image translation using cycle-consistent adversarial networks.

Appendix

In this additional section, the usage of our code is explained. With this report, we've included pre-trained models, as well as the code that can be used to train even more models.

Tone Transfer

To make the code universally usable, we've decided to use the online service Google Colaboratory (Colab) for the generation of DDSP-Models. Each trained model represents an individual instrument - which in our case, is a model representing a specific animal species. Due to the complexity of the installation of a docker-image, just the usage of the simpler online-version will be addressed here.

Model generation

With a google-account, a Colab-File can be created, in which the **.ipynb*-File can be imported. Alternatively, this colab[15] can be imported.

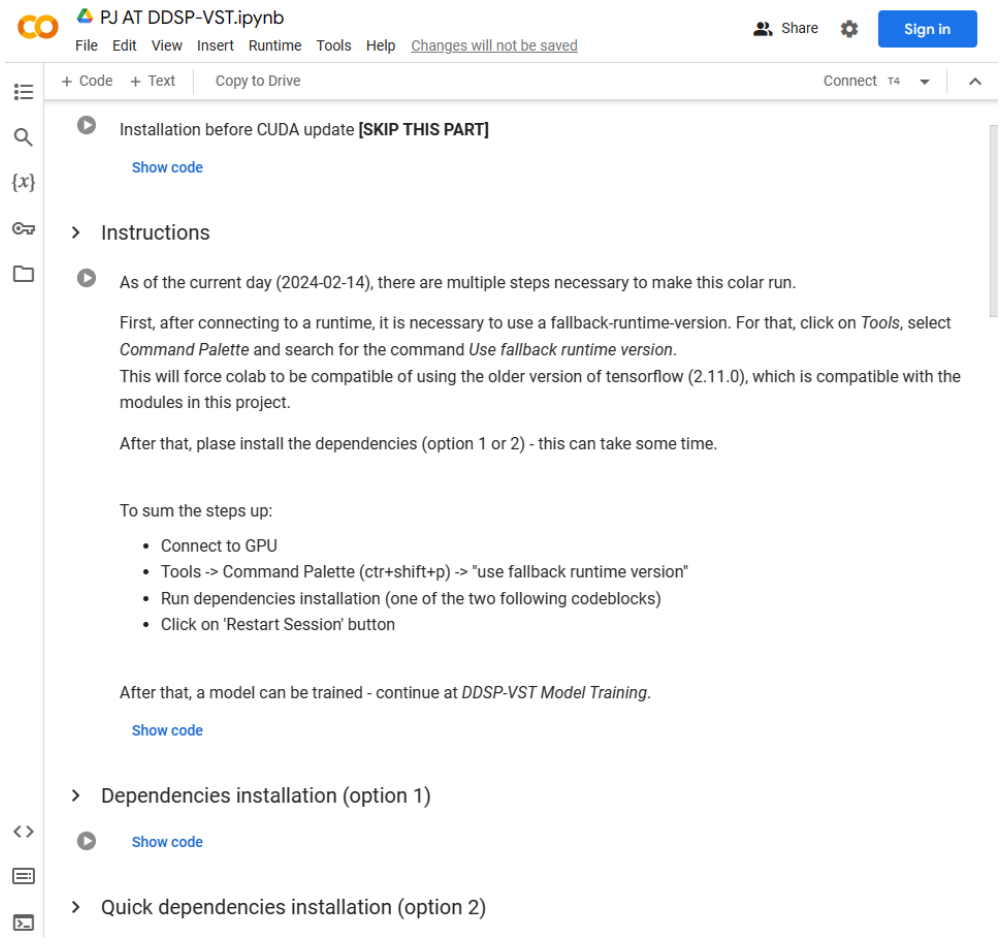


Figure 9: DDSP Colab

To train a model, it is a simple as following the instructions and clicking the "play"-buttons on each Codeblock. As it is mentioned in the colab-file, it is necessary to switch to a fallback runtime version, as the now one does not support the required *Tensorflow* version 2.11.0.

[15] `https://colab.research.google.com/drive/1t6L_4cC1OylOLLPoFHGRU123IID8zJRt`

Model usage

In accordance with section 3.2, any application that supports VST3-Plugins can be used to apply the trained model on the input audio. We've decided to use Helio[16] ((38)), as it is available for a variety of operating systems. For the DDSP-Plugin to be usable, please follow the instructions as mentioned on their Github-Page[17]. After a successful installation, a new instrument in the *Orchestra pit* (fig. 10) can be created. Each instrument represents one animal model.

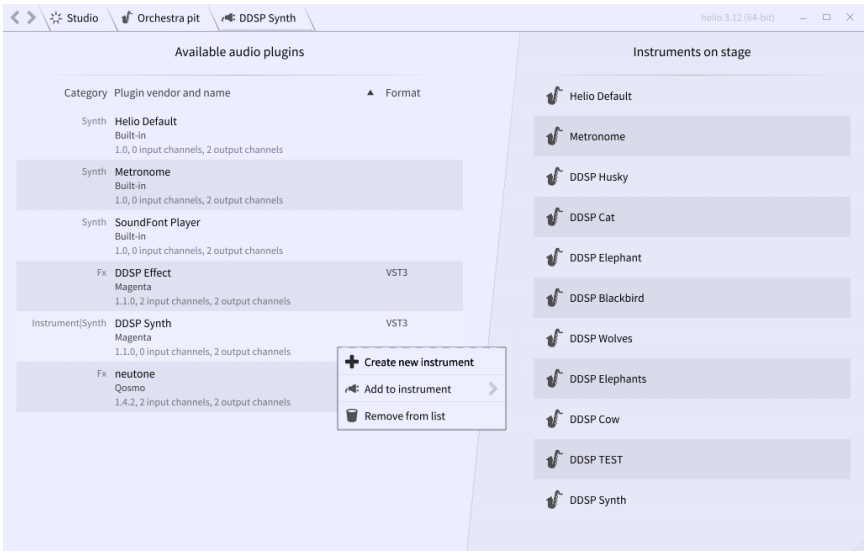


Figure 10: Helio - Orchester pit

To edit the settings of an instrument, right-click on it *(Instruments on stage)* and *Show UI*. From there, adjustments can be made, as shown in figure 5, like changing the harmonics or noise, to make the model sound like the actual animal as close as possible.

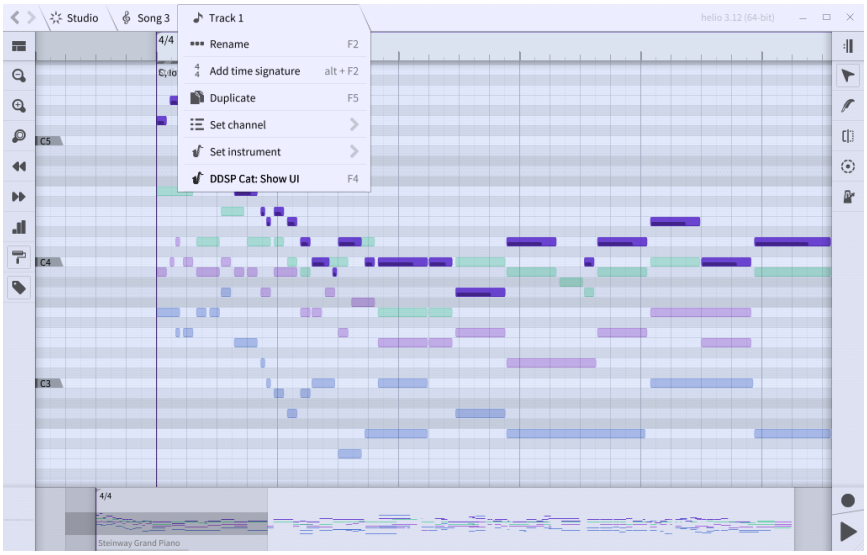


Figure 11: Helio - Instrument selection

With a generated composition, each track may contain a different instrument, as seen in fig 11. It is important to pay attention to which animal is applicable on which track, by analysing it's properties like the pitch and the length of it's notes.

[16] https://helio.fm/
[17] https://github.com/magenta/ddsp-vst