Visualization Research Center of the University of Stuttgart (VISUS)

Masterarbeit

Visualization for Human-AI Collaborative Music Composition

Simeon Rau

Course of Study:	Informatik
Examiner:	Prof. Dr. Michael Sedlmair
Supervisor:	Frank Heyen, M.Sc.
Commenced:	May 5, 2021
Completed:	November 30, 2021

Abstract

We propose an AI-assisted approach based on interactive visualizations to support users in composing music and getting insights into the AI through hyperparameter analysis. Our user-centered approach allows the user to better control the composition by steering the AI's suggestions. We use symbolic music data and piano rolls as visual music notation for easier understanding for amateur users and interaction with the notes of a melody. As the user requests multiple possible continuation for a given seed melody, and also multiple continuations for each of the previous continuations, a tree or graph structure of melodies occurs. We visualize this structure with an icicle plot, where the nodes are represented by a piano roll, to show the hierarchical structure of the melody samples. To add sorting options for easier sample selection, while still displaying the structure, we added links between the nodes. Both visualizations enable listening to selected melodies. For larger numbers of generated suggestions, we added a similarity-preserving scatterplot to visualize all samples at the same time with different glyphs representing melody samples. The scatterplot improves the efficiency of sample selection, as similar samples are close together and the user can disregard entire neighborhoods if one sample does not fit at all. We support brushing the scatterplot to select neighborhoods for which we then show visual aggregations to allow for insights into groups. To evaluate our design, we conducted a pair analytics study with two participants with limited musical knowledge. Both participants were able to quickly create compositions they liked and found our approach helpful. They also learned new things about the AI, like the influence of the hyperparameter temperature on the resulting melody.

Contents

List of Figures

List of Tables

List of Listings

1 Introduction

People like music but composing a new song is hard for different reasons: Successfully composing music often requires years of experience and learning theory, making it hard for beginners. Furthermore, the composer might lack creative ideas, which can result in the stagnation of the composing process or production of unsatisfying music for the composer himself.

With the recent advances in technology, many machine learning (ML) methods were discovered and developed to generate music [BHP20]. Especially different neural networks (NN) showed the most promising results, based on architectures like recurrent NN [MKG+17], convolutional NN [LCH+20], variational autoencoders [RER+18], transformers [HVU+18], or combinations of them [KDW18]. Some of these ML methods were designed to predict notes, while others generate the audio directly, but both focus on automation [FGJ20]. Although the imagination of generated music sounds interesting and is useful, for instance, as video background music, in reality people show some scepticism towards AI generated music at the current state [KC20]. However, purely AI generated music often lacks personality, long-term structure and likely will not completely replace human-composed pieces.

Instead, users want to have control over the action and use AI as an additional approach [KC20], even though steering the AI in a desired direction can be hard [GB21]. The AI can support the user with creative initial seed melodies or generating alternative ideas for continuations or replacements. This can speed up the progress and move the composing process forward quickly, while also granting the user authorship of the composition, because the user needs to decide when and how often the user requests the AI [SYTC21]. Then, the AI plays a collaborative, co-composer role and could benefit beginners, as well as experienced musicians [LSM18]. To improve the collaboration between AI and the user, interactive visualization as a way to communicate can help, while visualization gives users insight into abstract data, revealing possibly important information [BDG+20].

We combine interactive visualization and machine learning models for music generation and propose a user-centered approach. In this approach, the user queries the AI for multiple melody samples at the same time. For more personalized results, the user controls the AI through an initial melody and hyperparameters. After creating multiple suggestions, the user is supported by different visualizations to choose a fitting melody sample efficiently, without listening to all suggestions. The tree/graph structure, resulting from generating continuations and continuations of the continuations, is visualized using a icicle plot that shows the hierarchical structure. The nodes of this icicle plot contain piano rolls that visualize the respective melody sample.

We added links between the nodes to enable sorting options, while still displaying the hierarchical structure, to support more efficient sample selection, based on the selected sorting attributes. The user is able to listen to samples or a path of samples directly in the visualization.

As both previous visualizations struggle with a larger number of samples, we used a similarity-preserving scatterplot and different glyphs, to represent multiple samples and its attributes at once, while showing similar melodies close together. The user can select groups with a circular brush to get insights into groups, supported by aggregations of the samples, or select completely different melodies by moving away from that group.

We evaluated our proposed approach with a pair analytics study with two participants, testing the usability of the visualizations for different tasks. Both participants had limited knowledge about music theory and therefore represent a beginner user composing music or getting insights into the relationship between AI hyperparameter and the output melodies.

In summary, we contribute a user-centered approach that combines AI and visualization to help hobby musicians or composers with a lack of creative ideas in composing melodies interactively. We further support getting insights into the influence of hyperparameters on the output of the AI. Our approach consists of the following three parts: leveraging AI for piece-wise melody generation with multiple samples at a time, two graph-like visualizations to support more efficient sample selection, and a similarity-preserving scatterplot with melody glyphs for an overview over larger numbers of samples and their attributes.

This thesis is structured as follows: Chapter 2 summarizes background and related work, regarding visual representation of music, machine learning, visualization in music, and interactive music creation. In Chapter 3, we explain general concept of this thesis, consisting of target users and tasks, workflow of our approach, used data, design of our visualizations, and details on our implementation. An evaluation of our approach through pair analytics studies is presented in Chapter 4 and results and limitations are discussed in Chapter 5. Chapter 6 contains a conclusion of the presented work, followed by an outlook into possibilities for future work. It is important to notice, that mentioning temperature in this thesis always refers to the hyperparameter of the AI and not a musical temperature.

2 Background & Related Work

We divided this chapter into separate sections: First, we take a look at the used visual representation of sheet music. Next, related work regarding machine learning in music generation is presented and we discussed and tested some of the available models by TensorFlow's Magenta[1]. Then, we take a look at visualizations in combination with music data, followed by interactive systems for music composition.

2.1 Visual Representation of Sheet Music

The representation of music notes can be important in relation to the knowledge of the user. Amateur users with limited knowledge about music theory often struggle with the common modern staff notation while expert users are used to the notation. Due to the amateur users described in Section 3.1 the modern staff notation is not suitable for an efficient workflow. As an easy alternative, the concept of a Piano Roll is used, based on the mechanic piano roll, which were programmable via holes in the roll and used to play music automated [Roa85]. For experts, their favorite instrument could play an important role for the intuitive usage of piano roll notation. Especially for musicians using a piano it is more intuitive, while musicians playing other instruments, like guitar, might struggle with a piano roll for the first time. In this concept the notes are represented as rectangles in a graph, where the x-axis corresponds to the timeline and the y-axis represents the keys of a piano. Therefore the position and length of a rectangle encode all the relevant data of a note: the pitch, start time, and duration.

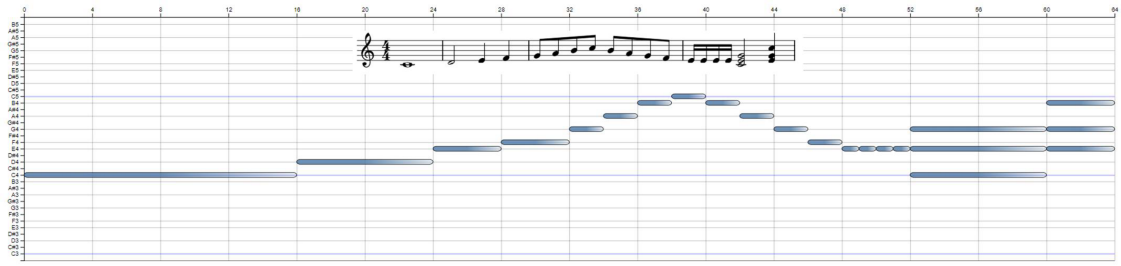


Figure 2.1: The same melody in modern staff notation (black) and visualized as piano roll (blue).

[1] https://magenta.tensorflow.org/

2.2 Machine Learning for Music Generation

The right Neural Network selection for music generation is significant, to meet the user's expectations of generated music samples. In this thesis, the Neural Network needs to be able to generate a continuation of a given musical sample but also to fill the space between two parts of the composition. To ease the complexity of a composition and the music generation we decided to only use monophonic NNs for the time being, as monophonic refers to only one voice and note at a time.

As mentioned in the Chapter 1, ML models exist, that predict the audio directly, like [MKG+17], but are not suitable for a iterative, composing workflow, which is a goal of this thesis. These are therefore not considered as possible models for this work.

Koh et al. [KDW18] developed a combination of different networks to reduce the problems of understanding higher level semantics of musical structure and generating novel patterns with little repetition. Therefore they combined a Convolutional Neural Network (CNN), to keep musical structure over several voices, a Variational Autoencoder (VAE), to create completely novel sequences, and a Recurrent Neural Network (RNN), to detect repetitive patterns. This proposed method generates novel, polyphonic music based on the structure of the input sample, but is not used in this thesis, as we start with monophonic music.

To achieve more creativity and diversity Dean et al. [DF20] developed a combination of RNN and CNN, which should generate a music sequence that is distinct from the output but not random, comparable to an improviser. While the controllable results showed signs of success, most of the time poor results are generated, which is one of the reasons for why we do not use this model in this thesis.

Roberts et al. [RER+18] proposed MusicVAE, a Recurrent VAE build with a LSTM encoder and a RNN decoder for music generation. The VAE is used to generate novel music samples without input or interpolate between two given melodies, using a latent space, where melodies are encoded as points. As the model can interpolate between two sequences, it is more fitting for filling space between two parts but lacks in the possibility to continue a given sequence. Therefore this model is not used in this thesis as a single solution, but could fit as additional model for a fill-in task.

Another VAE by Weber et al. [WATS19] uses two VAEs controlled by a 'creativity knob', to generate a novel, variation of the given seed melody. This should support musicians to explore the space of possibilities. As this model outputs only variations, the possibility of different continuations is missing, which is a key task for an iterative workflow and therefore not used in this thesis.

Simon et al. [SO17] propose Performance RNN, an effective and ready to use RNN with LSTMs, to generated music in a human-like performance way and therefore potentially mistimed notes and different velocity. The user can control by choosing a temperature value to control the randomness of the output melody. As this model is suited for generating continuations, we do not need the mistimed notes in a composition and therefore use a similar, but simpler model by Magenta explained later in this thesis.

Another RNN, proposed by Hadjeres et al. [HN20], uses an anticipation mechanism to enforce unary constraints, defined by the user, to influence the generated sample. This gives users more control, but also requires more knowledge for formulating constraints, which makes it hard for beginner users.

Watson Beat [Cha18] is a combination of Reinforcement Learning and a Restricted Boltzmann Machine (RBM) to generate music, based on imitating the rhythm with some randomness and learned rules in the melody. This model takes hyperparameters as input to generate a whole song in the form of a MIDI file. As we only need smaller parts for an iterative workflow, this model is not used in this thesis.

Behzadhaki et al. [HJ19] used text-based long short-term memory (LSTM) as an unpredictable source of creativity to generate a bassline correlated with a drum sequence, as these two parts of a song are highly correlated in some music styles. We do not use this model here, as it needs a drum sequence as additional input.

Huang et al. [HVU+18] used a transformer network for music generation, to deal with long term structure, as it uses a language-modeling approach to train the model. This was the first successful Transformer with long-term structure in music generation, but the field of Transformer networks in music generation is not as well researched as like RNNs. Since then, other transformer models [Lup21; NHJ21] occured, that show promising potential but struggle with expert level of rhythmic and harmonic consistency.

The open source research group Magenta[2] provides many different models for music generation, some being pre-trained and ready to use. We compared some of the models and chose three monophonic models, based on an RNN architecture, that can predict a continuation of the input melody. We chose the basicRNN, melodyRNN and the improvRNN, as they are controllable with a temperature hyperparameter, which relates to the randomness of the output notes. Although these models are similar, monophonic, and already pre-trained, which is one reason why we chose them, they differ in some small attributes as some use addition chord progressions or limit the output notes. A full comparison and the reasons for the usage decision is shown in Appendix Tables A.1 and A.2.

We tested the subjective quality of the selected models to get an impression of the output melody, depending on the model and the hyperparameters. We tested all models and an older version of the improv model with different temperatures and melodies, but only used one sample per case and rated the sample by the sound of the melody.

We found that independent from the model, a low temperature resulted in mostly the same notes used in the input melody and therefore the melody often sounded similar. As all models, except the old version of the improvRNN, showed different potentials, for example the melodyRNN showed high potential continuing complex melodies, we decided to use all three and let the user choose. Still, this short comparison is not as significant, due to the limited sample size and therefore models could potentially produce better results in other cases. More information and analysis can be found in the appendix sorted for models in Appendix Tables A.3 to A.6.

[2] https://magenta.tensorflow.org/

2.3 Visualization in Music and Event-based Visualization

A survey by Khulusi et al. [KKM+20] presented different visualization techniques for different task related to music. Therefore they divided the tasks into groups: music work, musical collection, musicians, and instruments. Music work is the most interesting group for this thesis, as it includes composing music. The two kinds of visualizations most used in music work are the piano roll view and different glyphs, which are both used in this thesis to support the composer. Still, other visualization techniques like charts, graphs, and timelines were rarely used.

As the concept of a piano roll view remains the same, different versions were developed in 3D [SW97] or 2D [TWM19], to show more information, like different instruments. Piano rolls are used today in known digital workstations for music composition or visualizations, like Ableton Live[3] or Synthesia[4]. Often the same problem occurs, that overlapping notes result in indistinguishable timings through the visualization, which is also one point we want to address later.

Wattenberg [Wat01] used glyphs to visualize the structure of a song. Arcs connect same parts of a song, where the number of arcs and the placement along a timeline should shape the song with a glyph. Glyphs of two songs can be used to compare repetitions or the general number of themes used in a song.

To visualize multiple event-sequences, Bruckner et al. [BM10] used a timeline were events at the same time are stacked vertically. Therefore this shows a graph, using the x-axis as a timeline, where the user can choose different paths, as multiple events at the same time function as options.

Similar to this graph visualization, Wongasuphasawat et al. [WGP+11] converted the event timeline into a tree of sequences and visualized it, using an icicle plot as inspiration. On top of an icicle plot visualization for event-sequences, Lui et al. [LKD+17] added a node-link visualization of the tree, as it is more familiar at first, as they stated. Although such node-link diagrams support the user depending on the complexity of the visualization [AGB21], we use them and the icicle plot, to show the hierarchy of generated music samples at different times.

We also use a similarity-preserving scatterplot to show clusters of similar predictions in order to maneuver through samples, although such visualizations show negligible help for accuracy prediction of classification [GBSW21], as our main goal is not the accuracy prediction but cluster representation itself.

2.4 Interactive Music Creation

There are many playful applications, using interaction for music creation. Most of these applications have limited control, like changing the pace, the randomness, or given seed patterns, to create small samples of music, which is used to interact with an AI in a playful way [Par18; Pas18]. While these can be fun to use, they are not destined to compose complex music.

[3] https://www.ableton.com/en/live/
[4] https://synthesiagame.com/

In comparison, other application use more complex interactions to generate music. A approach proposed by Frid et al. [FGJ20] uses a video and an example song, to inspire the AI, which generates multiple tracks, that should fit the video and similarity towards the example song. The user can then mix the generated tracks, to create a copyright free, background song for the video. Although the user can influence the output song, this is not a approach for music composition but for music generation.

Huang et al. [HDG16] want to assist the user, who wants to compose music, by suggesting different chord progressions, on given chords. The AI suggests multiple options, that can change some chords or add fitting chords at the end, where the user can listen to the options with an user interface. As the approach is kept simple, is does not use further visualization to support the user.

A different approach to interact with an AI is presented by Zhang et al. [ZXLD21], where the user has a conversation with the AI, to describe his imagination of the melody. As the AI generates a melody based on the constraints, the user can afterwards change section with the given conversation technique. Due limited control using the word-based approach, we do not use such an approach, but want to use visualization for better communication and interaction.

Agostini et al. [AG15] provide a library to compose music with the possibility to link an AI model as support. For the composing process, the user can edit notes and generated sequences with interactive visualizations using mouse clicks, keyboard, or messages. They use the modern staff music notation, which can be harder for beginners. Although this library provides the possibility to add an AI, it lacks in control as it is not developed for AI specifically.

A study about Human-AI co-creation by Huang et al. [HKN+20] gives some insights about user interfaces and functions, needed for a good cooperative workflow with the AI. They stated, that the participants needed control through interfaces to interact with the AI and the generated notes, creative freedom, and multiple functions to generate new samples, fine tune sections, or interpolate between sequences. Results showed, that users often generated multiple samples after another until they found fitting ones, compared them and selected the one, as most of the generated samples were not the expected outputs. To address this workflow, we use this as basis and generate multiple samples at once and support the user selecting fitting samples with interactive visualizations.

Music sketchnet is a approach, proposed by Chen et al. [CWBD20], to fill-in missing parts of a composition. To control the ai, the user is able to roughly sketch the melody by providing a rhythm, notes or other conditions. The the AI tries to interpolate between the given sequences, while fulfilling the constraints given, the user is not able to change the notes afterwards and is therefore forced to generate the same part multiple times. While the idea to sketch a part as controlling technique sounds interesting, the approach has limited possibilities when composing music. Similar problems occur in NONOTO, proposed by Bazin et al. [BH19], as this approach also provides an interactive interface to fill space between two sequences. Therefore we want to support a fill-in function as well, but generate multiple options at the same time instead of one at a time.

Another approach, called Cococo by Louie et al. [LCH+20], uses a steerable AI to harmonize a given melody. This means the AI suggests multiple options for different voices, based on chosen hyperparameters, where to user can choose from. The user can control the AI with sliders, suggestions for example a happy harmonization, and adjust single notes afterwards. With that amount of control the user gets more authorship than by just generating music. Our approach is influenced by this

concept, but we extend on the control of the user, as we allow the user to generate more options and support the user with visualization for a better selection process. As Cococo harmonizes a given melody, our approach can be used to compose that melody.

3 Concept

Firstly, we explain the users and tasks of our approach in Section 3.1, as we have the composer and AI analyst as user with different goals, In Section 3.2, the difference in the workflow between common systems, generating music purely through AI, and our AI-assisted user-centered approach is shown. Then we explain the data structure, used for the visualizations, in Section 3.3, as it explains the occurring tree or graph structure of melodies. All visualizations and design choices are then shown and explained in Section 3.4. Firstly, our version of a piano roll is explained, followed by both tree/graph based visualizations, the icicle plot and the node-link diagram. Then we show the similarity-preserving scatterplot (SPS) with all different glyphs, encoding different attributes of the melody samples. Some information about the implementation of the approach can be found in Section 3.5.

3.1 Users and Tasks

There are two different types of users in different scenarios: On the one hand an amateur composer and on the other hand an AI analyst. The composer is interested in composing a new melody for his own song but has limited knowledge in music theory or lacks in initial ideas for the melody. The AI analyst is interested in new technology like AI in music and wants to learn how hyperparameters could impact the result given by the AI and therefore the produced music. A single person can share characteristics of both types of users and can be interested in all of the tasks and goals at the same time.

3.1.1 Composer

Imagine an amateur composer, whose knowledge about music theory is limited or who cannot use the knowledge properly, but still wants to compose a melody for an own music piece. Due to the amateur status, the user might need help when converting ideas into whole melodies, continuing the ideas or finding a seed for a melody. The usage of AI can help with all these problems when the user has the possibility to interact properly. Therefore, the user desires full control over the steps of the composition but still wants to create the music fast and easily. The user gets supported by the AI in the form of suggestions, which are then visualized to make the process of finding and deciding on a good suggestion more efficient. Therefore, the user can use different visualizations to compare different samples. With the possibility to input and adjust notes, generate suggestions on demand and get supported in decisions by visualizations, all needs of the users to compose music with the help of AI are met.

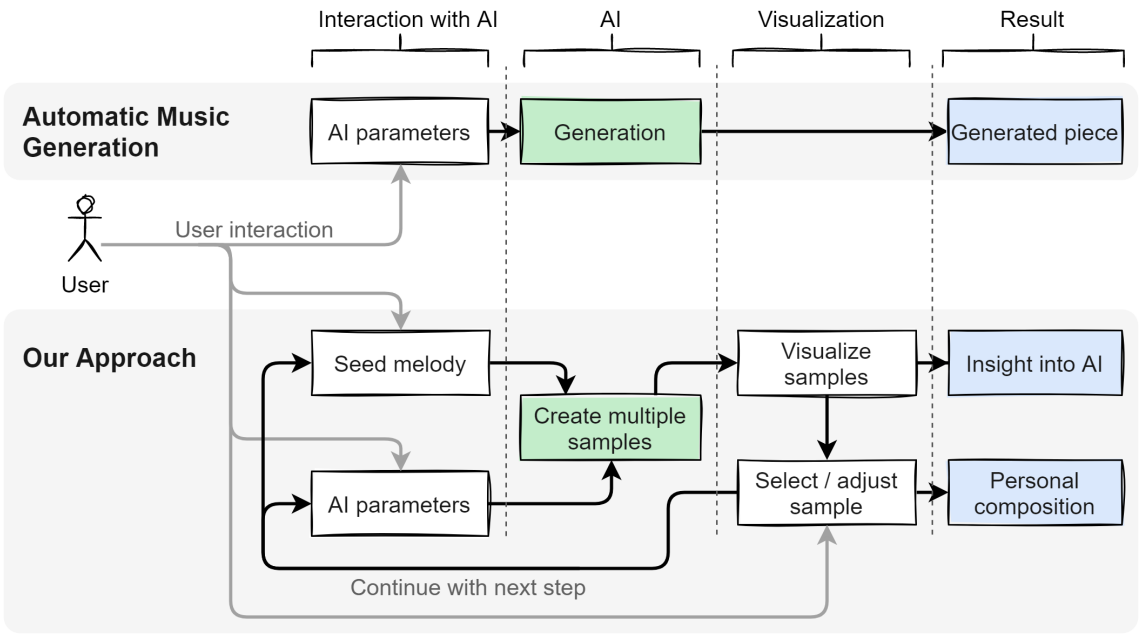


Figure 3.1: Our approach gives users control and artistic freedom through iterative choice and adaption of AI-suggested melodies. Users gain insight into the AI's behavior.

3.1.2 AI Analyst

An AI analyst is interested in learning how AI can work in music and how it can be used to full potential in combination with a human. Therefore, the user wants to gain insights into the impact of different hyperparameters on the musical output. How the AI reacts to different example melodies or to different steering methods and values are crucial questions for this user. The insights are gained by different visualizations, showing multiple relationships of data and context between hyperparameters and AI output. The user can use these insights later to improve efficiency when composing in a collaborative manner with AI.

3.2 Workflow

In this section, the workflow between a general automated music generation task and our Human-AI collaborative approach is compared. The general idea is that a user wants to create an own song with the help of the AI. In an automated approach, the user controls the ML model with hyperparameters which then generates a whole song and outputs the generated music piece, where the user has little authorship. The approach of this thesis not only uses hyperparameters as a way to control the output, but the user can also provide an example melody to steer the AI into a certain style of music. The AI then generates multiple short samples as continuations for the example melody. These samples are then visualized and the user is able to interact by selecting or adjusting a sample. Then the user can decide to compose a next part by repeating this step and composes a melody in an iterative manner. The output is then a personal composition, where the user has high authorship.

3.3 Data

The smallest building block of the used data is a single note. Each note consists of a pitch, the start time step, and the end time step. Multiple notes can be combined to a melody part of the chosen length. Melody parts are stored independently and are not combined apart from the current composition. Therefore, the data consists of multiple melodies or parts, which were either inputted by the user or generated by the AI.

The AI uses a given melody as example to generate a short sample of notes. This sample is a continuation of the example sample and therefore later in time but also in a similar style. The user can manipulate the data by adjusting single notes or influence the AI via hyperparameters. Often, one generated continuation does not meet the expectation and therefore one solution for the user was to repeat the generation multiple times. To prevent this from happening the user wants to generate multiple continuation samples at the same time to not miss out on the interesting ones. Each of these samples can then have multiple continuations as well.

The user can repeat the steps to generate samples in advance. This results in a tree structure of samples where the the example melodies correspond as parents for the generated continuations. The relationship represented in the tree shows the schedule of samples as well as the input and output of the AI generation. The root of the tree corresponds to the current composition of the user.

Imagine the user wants to go back to a part of his composition, the root, and recompose the chosen part. To not miss out on good suggestions by the AI, the user wants to generate multiple samples at the same time as well. These samples are called fill-in samples from now on. These fill-in samples work just like all other continuations in terms of them being a child of the root in the given tree structure. The only difference to the other children is, that the notes of a fill-in samples start and end before the end time of the composition. Therefore it results in a graph structure instead of a tree and will be called melody graph from now on. This difference is important for the visualizations later.

3.4 Design

In this section, different types of visualizations are shown and the design choices are explained. The visualizations are designed to support the different types of users in fulfilling their tasks. Therefore, the main focus of a visualization can vary and different tasks are supported. The following visualizations are explained in this chapter. First a Piano Roll visualization is shown in Section 3.4.1. Then the mentioned tree structure is visualized with the help of an icicle tree structure in Section 3.4.2. A node-link diagram, a flow visualization, which can also show the tree structure is explained in Section 3.4.3. Addressing different tasks and problems, a similarity scatterplot is more suited than the previous ones and is explained in Section 3.4.4.

3.4.1 Piano Roll

As mentioned in Section 2.1, the way to visualize notes of a melody chosen in this thesis is the concept of a piano roll. The piano roll with its explanations are shown in Figure 3.2. In the following part, the word note refers to the rectangle visualizing the note in the piano roll.

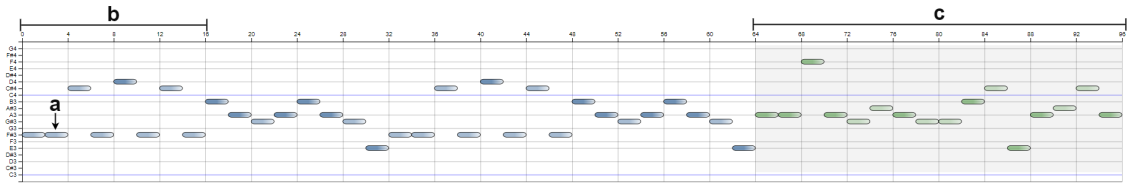


Figure 3.2: A piano roll visualizing a sequences of notes. Pitch, start time, and duration of a note are encoded by the position and the length of the respective rectangle. Y-axis shows the pitches and x-axis shows a linear timeline. Color of notes encode the belonging to a melody sample. The ticks of C are highlighted, while ticks of sharp notes are missing and the representation of sharp notes are more transparent. **a)** An eighth note. **b)** One bar (16 time steps). **c)** AI generated part, highlighted with gray background.

When visualizing several notes inside a piano roll, different aspects for understandability by the user have to be taken care of. First, each note can be represented with a simple rectangle, that can also be dragged by the user to adjust the timing, duration, and pitch of that note. With this start, the problem occurred that for example two consecutive quarter notes with the same pitch looked the same as one half note at the same pitch. So the user was not able to tell the difference between melodies in certain cases.

The boundary of a note has to be clear for the user in order to avoid misunderstandings of the length of a note. This problem is addressed with multiple solutions. Our first solution was that each note has a small black border so the length of a note is clearly visualized. The second solution is that the corners of the rectangle are curved so it is much easier to see the difference in the example at the first glance. Another idea is to use the color, specifically the gradient of the color. The gradient starts at a high opacity of the color and gets lower to the end of the note. This also leads to an easier view of the length of the note. All these ideas and solutions where used in this thesis, as shown in Figure 3.2, to make sure the notes can be seen well and the length of a note is clearly shown.

As support for determining the pitch of a note more easily, the ticks of the y-axis are not drawn for each pitch. Only ticks for natural notes are shown so it is much easier to find a specific pitch. To emphasize this, the color is also used to reflect the difference between natural and sharp notes. For visualization, the opacity of the color is used, so sharp notes have a lower opacity than natural notes. These two aspects aim for the same target and should help the user to recognize the pitch of a note faster. To support the readability of a pitch the baseline of an octave can be helpful. Therefore, the tick of each key C is highlighted with a thicker, blue line.

An idea to improve the identification of the pitch of a note, the specific color was chosen based on the pitch. For example, a note C would be marked with the color blue and a note D with the color red. Due to the data structure, as explained in Section 3.3, different parts of the melody have to be visually encoded. A good way to show different parts in a melody is by color and therefore the previous idea was discarded. The color now represents the affiliation of a note for a part. As shown in Figure 3.2, all blue notes belong to the same part while all green notes belong to another part.

The chosen colors where selected carefully with the intent to show differences well, as the various parts of the melody should be clearly distinguishable. To show the difference between several parts, a chosen color should be different to all other colors, so all selected colors need to be pair-wise

Figure 3.3: Color palette Tableau 10 specialized on showing differences and a sophisticated look.

different. Due to humans' limited ability to distinguish between multiple colors, a maximum of 10 colors is chosen. Because of these requirements to show differences, we chose Tableau 10 [Sto16] as our color palette, shown in Figure 3.3.

A different problem other piano roll visualizations had was that overlapping notes were difficult to detect and the length of the notes could be misunderstood. Imagine a quarter note is drawn underneath a half note of the same pitch, the quarter note cannot be seen by the user due to the smaller length and the drawing level. Addressing this problem, the idea is to lower the opacity of a note so the user is able to detect multiple notes at the same spot. Due to additive colors of the notes, the user can detect if overlapping notes have different colors and therefore belong to different parts. Although this is not a perfect solution and there are still cases where this does not lead to the right conclusion, this idea is a step into the right direction and solves one part of the given problem.

The question of authorship often occurs when working together with an AI. Therefore, it can be helpful for the user to visually be reminded of which parts where purely generated by the AI and which parts where either actively composed, adjusted, or approved by the user. As shown in Figure 3.2, the time area where only AI generated notes are underlaid with a gray rectangle. This indicates that all notes inside this rectangle are purely AI generated and not yet approved by the user.

In order to use a specific part as a seed melody for predictions, the user is able to brush a part and then generate new melody samples as continuations for the selected part. If the user wants to go back to a part of his composition to replace it with some new suggestions to improve that part, the same concept as generating continuations applies. To not miss out on any samples the user can generate multiple fill-in samples at the same time by using the brush for selection and the AI to generate fill-in samples for that selected part. The user is able to listen to all samples or only to a selected part via the brush. A line functions as playback indicator and shows the current time of the sample while playing.

3.4.2 Icicle

An icicle plot [KL83] shows a hierarchical order where a node is next to its parent node and the height is determined by the sum of heights for all children in one level. We visualize our melody graph with an icicle plot, where nodes show a piano roll, containing the notes of the corresponding sample (Figure 3.4). These piano rolls share a common time axis from left to right which also indicates the relationship between samples. Nevertheless, each piano roll has its own pitch axis from bottom to top.

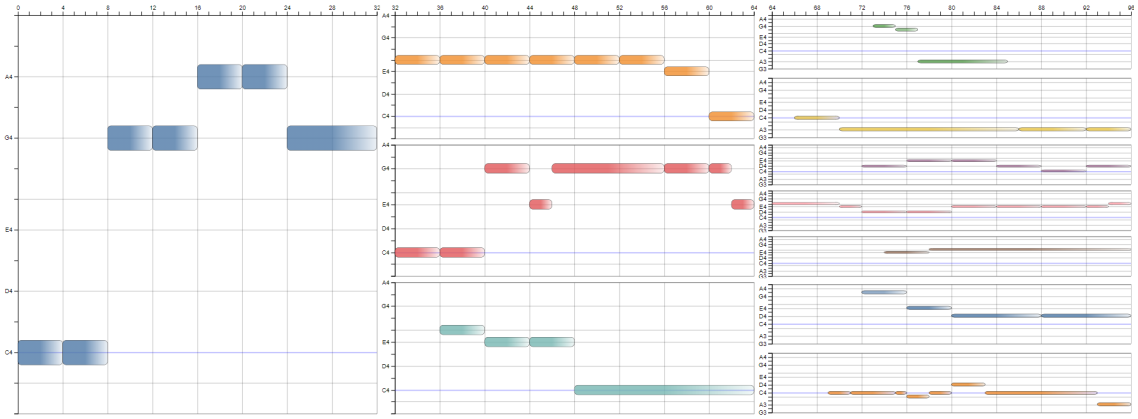


Figure 3.4: We visualize a graph of melody samples similar to an icicle plot with piano rolls in all nodes. Y-axis is separate for each Node while x-axis is a common timeline. The visualization has 3 levels.

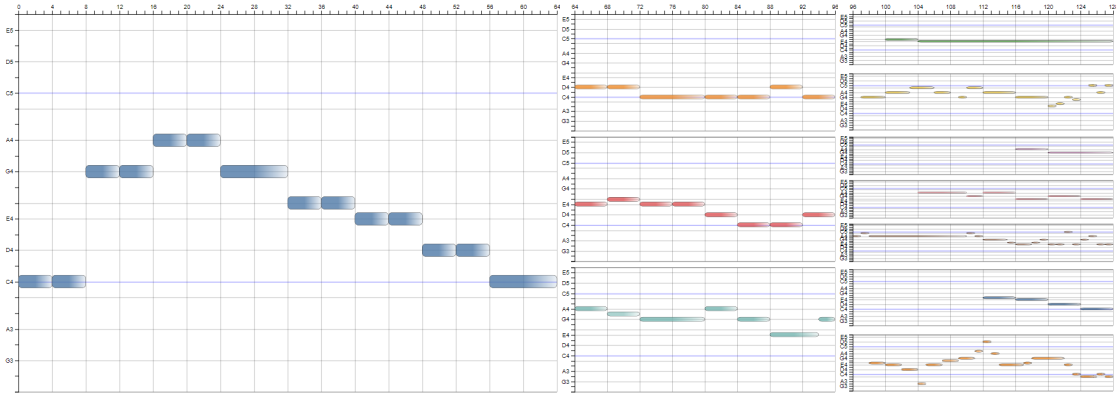
We allow the user to interact with the melody samples in the icicle plot by selecting a sample. The selection includes the whole path to the root and shows all the notes in a separate piano roll. Here, the user can play the whole melody or just a selected part, via a brush. A playback indicator helps when following the current played notes. With a simple mouse click the user can add the selection to his composition.

Because each node has its own axis from bottom to top, indicating the range of pitches shown in the piano roll, the selection of the right scale is important for different views. We tried three different types of scale selection, each with advantages and disadvantages regarding the understanding of the visualization (Figure 3.5).

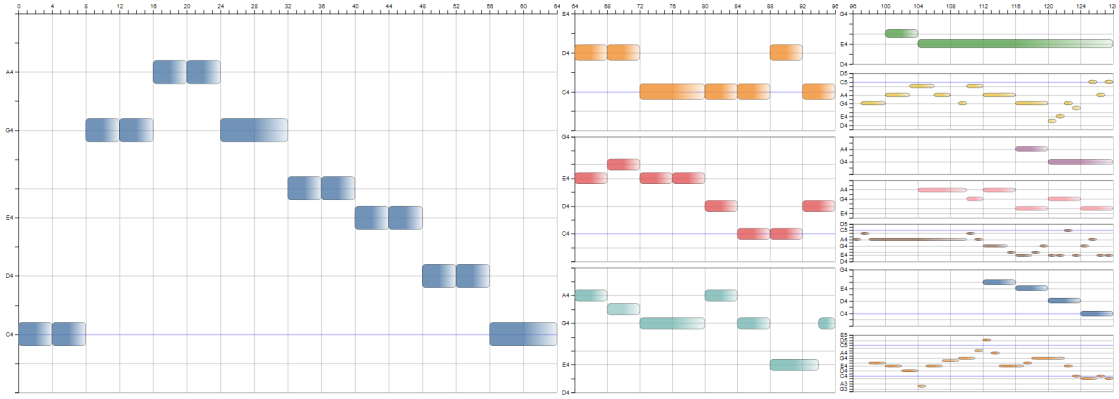
The first idea was to use a global scale, so each node uses the range on the axis. Therefore, the maximum and minimum pitch over all samples where calculated. This led to a higher amount of white space, if some outlier notes produced a high range, where the range is too large for most samples. On the other hand, the context between samples and the total position of a note is kept more intuitively. For example a note shown in the middle of a node, always shows the same pitch. These phenomenon can be seen in Figure 3.5a.

To fix the problem of the high amount of white space in each node, we tried a local scale for each node. Therefore the maximum and minimum pitch are calculated for each node independently. Here outlier notes can produce a high range for the scale but it only impacts the own node instead of all the other nodes. This reduces the white space due to the adapted axis. A drawback of this method is, that the context between samples is not kept and it is therefore harder to compare some of the melodies. This means, two melodies could consist of one note but different pitch, which would lead to a similar representations in the icicle plot with different meanings. Therefore, a note in the middle of a node could represent different pitches. This approach is shown in Figure 3.5b.

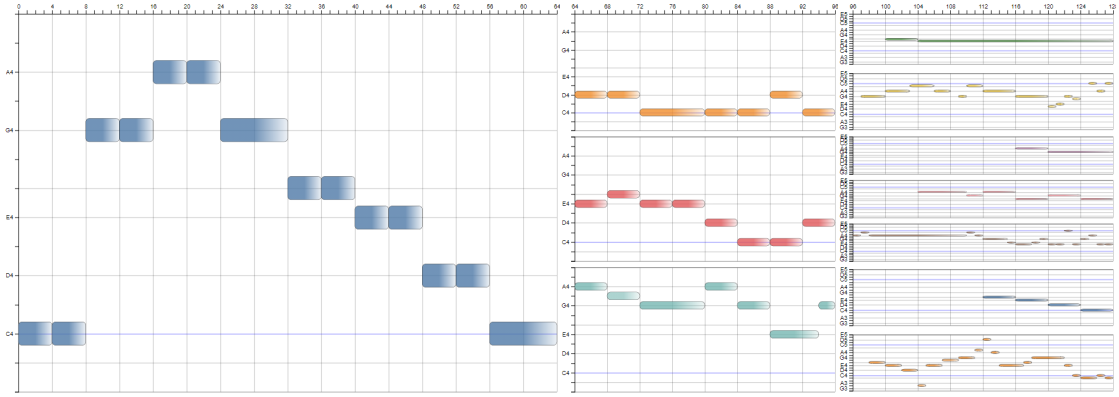
As a compromise for the previous approaches we tried a combination, where the scales are calculated independently for each level. Therefore, the white space is reduced for each level but context inside a level is kept. The problem of outlier notes can still occur but only impacts the respective level. This approach is shown in Figure 3.5c. The user is able to switch between the scales depending on the task and his preference.



(a) Global y-axis.



(b) Local y-axis.



(c) Same y-axis per level.

Figure 3.5: Our icicle plot visualization of the same tree-like structured data with different y-axis types. **a)** All elements share the same range. **b)** Local calculated range for each node. **c)** All elements of the same level share the same range.

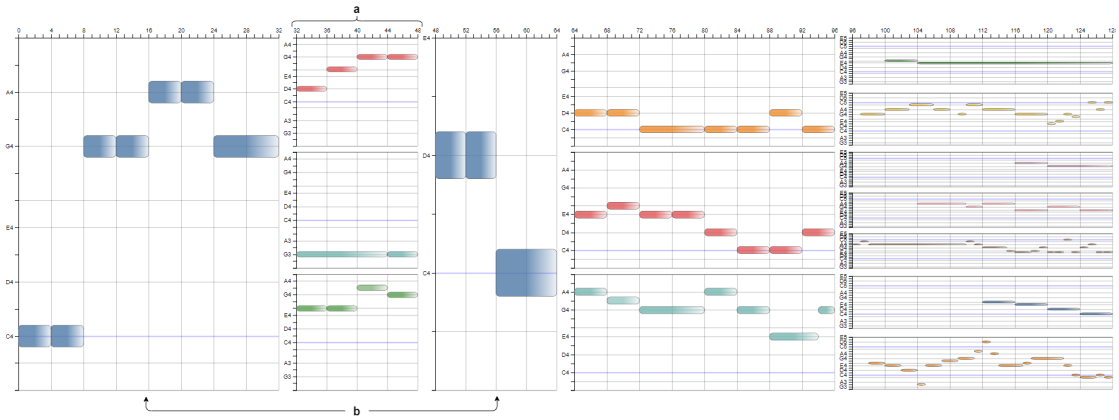


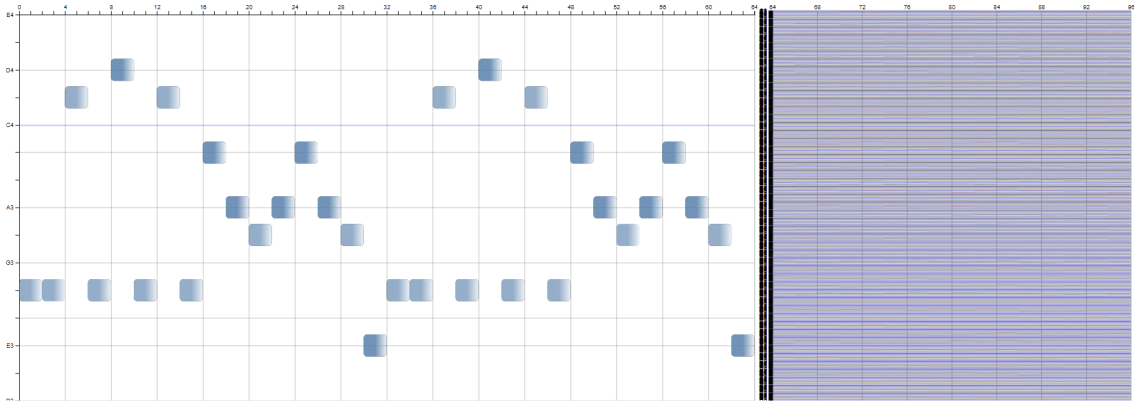
Figure 3.6: Our icicle plot, showing the graph-like data structure with fill-in samples. **a)** Fill-in samples. **b)** The root composition is split into two halves due to the timing of the notes.

As addressed in Section 3.3, the user is able to go back to a part of his composition to fill-in multiple samples as replacement suggestions for the selected part. This results in a melody graph with the corresponding icicle plot shown in Figure 3.6. Here, the root note is split into two parts, both taking the full height, while the fill-in samples are placed between them in the same manner other children are placed.

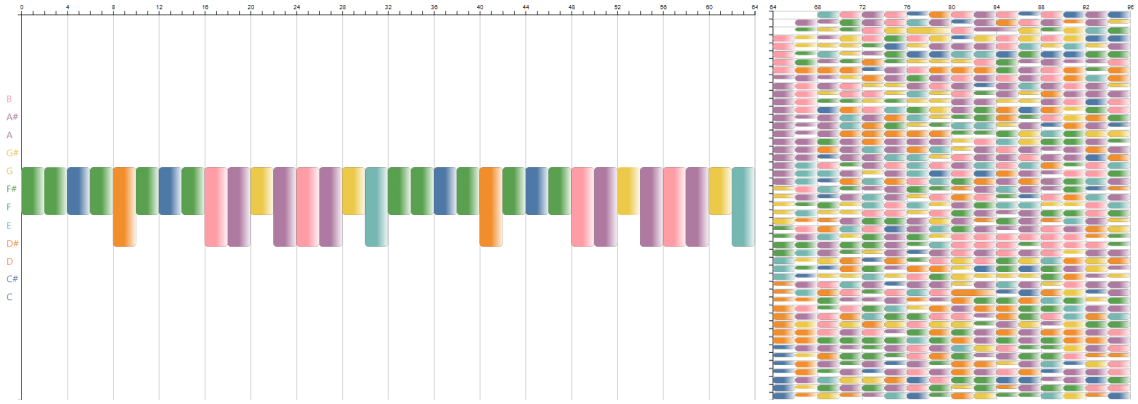
When generating a large amount of sample melodies as continuations ,for example 50 samples, the normal icicle plot representation struggles with displaying all nodes and their information. Due to the limited space on a level the space each child gets is too small. As shown in Figure 3.7a, the nodes are too small, resulting in a clutter where the information is not recognizable. To solve this problem we chose a different visualization to encode notes of a melody sample. As shown in Figure 3.7b, we decided to replace the piano rolls with a line, where the notes are placed on. The position of a rectangle on the x-axis corresponds to the start timing and the duration of the note as encoded in a normal piano roll. A note's pitch is encoded by the color and the height, where the color corresponds to the pitch named in the legend on the left and the smaller height indicates a # note. Important to say is that the octave number for a pitch is omitted and therefore a C note could show a C4, as well as a C5 note. To help the user with selecting samples by their starting note, samples are sorted within a level and the same parent by the start note of the melody, showing samples starting with C at the bottom and samples starting with a pause at the top.

3.4.3 Node-Link Diagram

To further improve sorting as support for efficient sample selection, we chose different sorting metrics, which are shown later. Sorting all nodes inside a single level of the icicle by a chosen metric often results in the problem that nodes could not be shown besides their parents. Therefore an icicle plot was not sufficient to show relationships and sorting the nodes of a whole level. To solve this problem we added links between the nodes to show a relationship independent from the position of the nodes. The links additionally allow to encode the value of the chosen metric inside their width. So for example when sorting by *temperature* the link width encodes the value of temperature, showing wider links for higher temperature values.



(a) Piano rolls, pitch encoded by y-position.



(b) Pitch encoded by size and color.

Figure 3.7: The icicle visualization with 50 generated continuations. **a)** All 50 piano rolls share a limited space, resulting in clutter. **b)** Pitch is encoded with color as seen in the legend on the y-axis but octaves are omitted. For example a blue note can be C4 or C5 or any other C. Sharp notes have smaller height like on a piano where black keys are shorter than white keys. Melody samples are sorted by starting notes, C at the bottom and empty at the top.

We decided to fill the nodes with simpler versions of piano rolls to reduce clutter and improve visibility. The simplier versions of piano rolls do not show axis but only the respective notes in their position. Therefore, the melodic structure is still visible but the context and the actual pitch of the notes are lost. We decided to trade the missing pitch for better visibility to reduced clutter, because the melodic structure already can indicate more interesting melodies. The actual pitch of a note is not as helpful, especially for novice users, as a visual representation of a melody's structure.

In order to be able to see the actual pitch of notes from a chosen melody, we allow the user to view all nodes of a selected path in the separate, detailed piano roll mentioned in Section 3.4.2 by mouse clicking on the respective node in our node-link diagram (Figure 3.8). We also allow the user to listen to a single melody sample or the whole path to the root directly in the node-link diagram, by hovering over a node and mouse clicking on the respective button. A playback indicator shows which node and which note is currently played. The user is also able to directly add a node and the

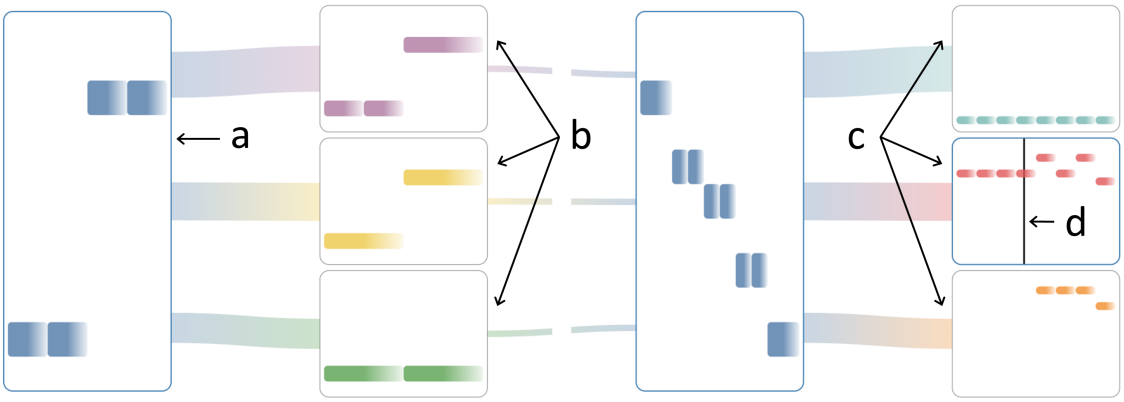


Figure 3.8: Our node-link diagram visualization of a melody sample graph. Nodes represent samples and display them through piano rolls, links connect related nodes and encode the current sorting metric's values in their width. **a)** The full-height nodes are part of the current composition. **b)** Nodes in the middle show options for a fill-in. **c)** The right-most nodes show possible continuations. **d)** A time cursor shows that the center right node is currently played back as audio.

whole path to the composition. We decided to automatically generate the next set of continuations when using the adding method from the node-link diagram, because it automatically updates the visualization and enables a fast way to compose music in an iterative way. The user has full control over the selection of samples and could compose a longer melody in a short amount of time.

We chose four different metrics to sort by in order to improve sample selection depending on a given idea. For example, the user could sort the samples by *variance of intervals* between notes to filter more exciting melodies (Figure 3.9 **Bottom**). Other metrics we chose were the temperature used for generation of the sample and similarity to the parent. When sorting by *similarity to the parent*, the user should be able to find small variations or completely different samples depending on the intention of the user.

If the user searches for a continuation with a high similarity but small changes compared to the seed melody sample, he should be able to sort by *similarity to the parent* in a descending way. The used similarity function is explained in the next Section 3.4.4 and shown in more detail in Listing 3.1.

The metric temperature intents to support the user when searching for more or less randomness inside the melody samples (Figure 3.9 **Middle**). Melody samples generated with smaller temperature contain a smaller amount of random notes. This can also be interesting for the AI analyst user and support the investigation of melody samples depending on temperature. We chose these three metrics because it enables the user to search for exciting or calm melodies as well as for continuations, that are small variations or completely different melodies. The temperature metric was chosen by us, because it allows both the composer as well as the AI analyst to search for randomness in the melody.

We also added the normal relationship, also used in the icicle, as sorting to achieve a better overview of all samples and to show parents and children besides each other without link crossings (Figure 3.9 **Top**). Of course there are many more possible metrics, with different intentions to fulfill other tasks, that could replace the current ones or be added.

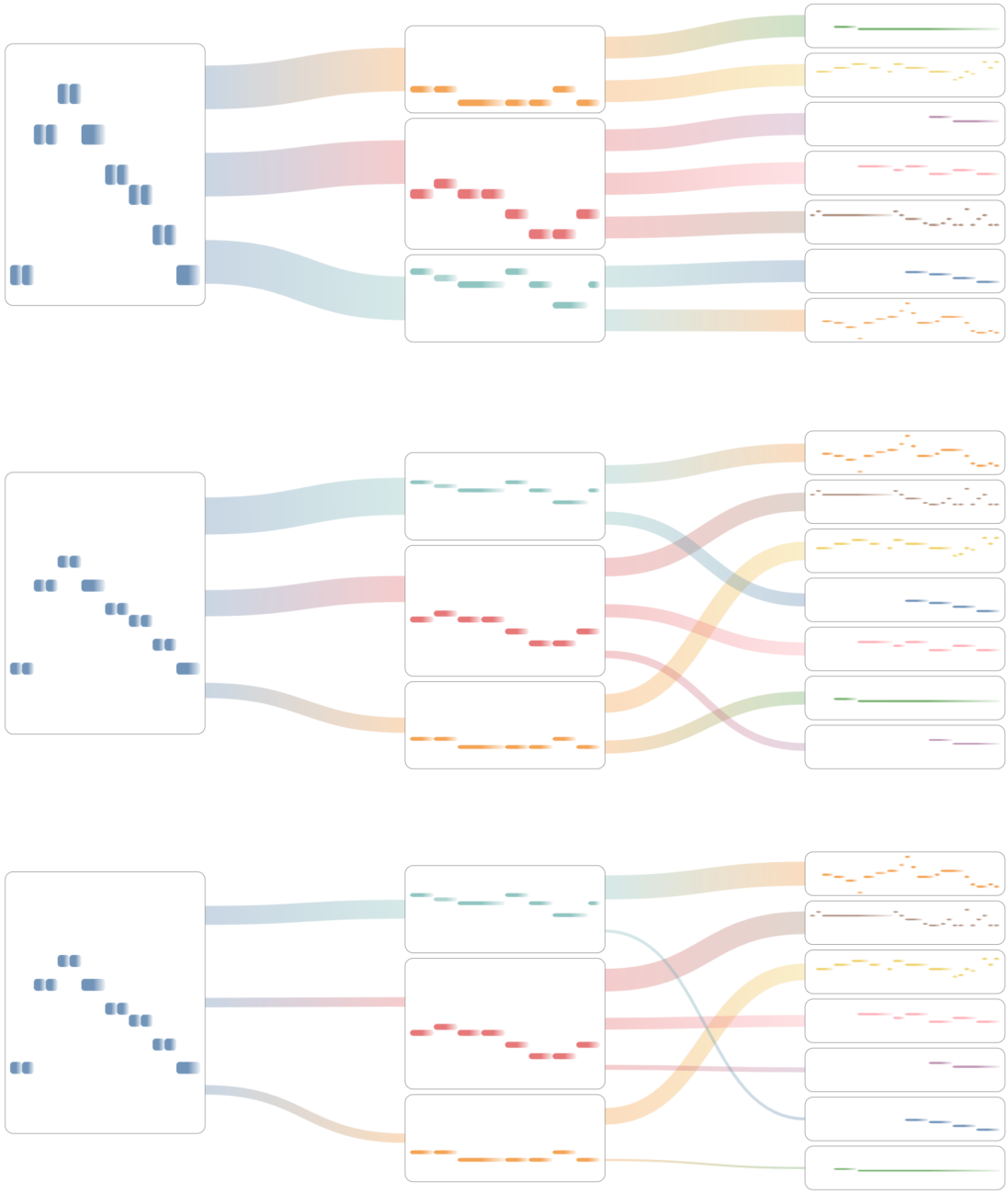


Figure 3.9: Three node-link diagrams with the same data but different sortings. All nodes on the same level are sorted by the chosen metric for better selection. Links show relationship while also encoding the chosen sorting metric's values in their width. **Top)** Normal tree relationships without any sorting. **Middle)** Nodes of the same level are sorted in descending order according to temperature. Bigger links correspond to a higher temperature. **Bottom)** Nodes of the same level are sorted in descending order according to the variance of all pitch differences between notes. For example the pitch difference between C4 and E4, which converts to MIDI values of 60 and 64, is four. Bigger links correspond to a higher variance value.

3.4.4 Similarity Scatterplot

In order to better understand the AI model and its suggestions, the user can generate tens or hundreds of melody samples with different model hyperparameters. Since our previous visualizations, the icicle plot (Section 3.4.2) and the node-link diagram (Section 3.4.3), do not scale with this amount of data, we chose a different design for this use case. To solve this problem, we introduce a third visualization to show an overview of all samples by visually grouping them based on the similarity of the melodies in a so called similarity-preserving scatter plot (SPS).

Similarity-Preserving Scatterplot

Similarity-preserving scatterplots are often used in different areas to give an overview before filtering and investigating interesting looking groups or single samples. As proposed in a work from Sedlmair et al. [SMT13], 2D scatter plots to reveal clusters in the output of dimensionality reduction algorithms are a promising approach and a reason why we chose an SPS as an overview of all samples. They are used in video analysis to find interesting parts in a video, as proposed by Achberger et al. [ACTS20], in order to increase efficiency compared to traditional analysis. Finding songs of similar styles in a music collection is also a use case of SPS, as proposed by Gomez et al. [GGKG20]. Le et al. [LND21] used an SPS to show differences between instruments for given audio samples. In the previous works using an SPS, as well as other works like [SSS+19], the similarity is calculated in wave form with Mel-Frequency Cepstral Coefficients (MFCC), which extract features of samples. In this thesis, this function is not applicable due to the different types of data.

Symbolic melody similarity is needed when using our representation of notes. DeHaas et al. [DWV13] proposed a symbolic harmonic similarity to calculate the similarity based on chords. Due to the chosen limitation of monophonic samples in this thesis, this similarity function is not usable in our case. Urbano et al. [ULMS10] proposed a different approach using the melodic shape in the form of a curve interpolation to measure similarities between the curves. This approach was especially designed for polyphony and harmonics and is therefore too complex for our use case. Another similarity function, proposed by Park et al. [PKL+19], uses the edit distance between two melodies for pitch difference and rhythm separately and in a multi-scaled way. A common problem for these techniques is a time consuming calculation. In order to improve usability we chose to trade-off accuracy of similarity to reduce calculation time. Because similarities are calculated pair-wise between each sample, the number of calculations can add up quickly when generating hundreds of samples.

Our similarity function is used in the following SPS Section 3.4.4 and for sorting samples as in Section 3.4.3. This function calculates a similarity between two melody samples and provides a value between 0 and 1, where 0 corresponds to totally different and 1 highly similar samples. We decided to calculate a rhythmic and a melodic similarity independent from each other and combine them with a weight, inspired by [PKL+19]. This allows the user to decide which part is more interesting and should be weighted more. A simplified version of the similarity function for monophonic melodies is shown in Listing 3.1. We also tested the string based Levenshtein-distance as similarity function. Therefore, we converted monophonic melodies to a string containing the notes at each timestep. With that string conversion the rhythm, especially the note duration, was not taken into account and we therefore chose our combined function.

Listing 3.1 Code to compute the similarity of two monophonic melodies. Similarity is calculated by weighting rhythmic and melodic similarity, which are calculated separately. For the melodic similarity, only the differences between two neighbor notes are taken into a count, instead of actual pitch, to account for shifted melodies with the same structure.
Choosing the weight: 0 = rhythmic similarity, 1 = melodic similarity.

```
def similarity(melo1,melo2,weight):                    #musical length in quantized steps.
    rhythm = 0
    melody = 0
    for i in range(min(musicalLength(melo1),musicalLength(melo2)):
        note1 = findNote(melo1,i)                      #find note at timestep i
        note2 = findNote(melo2,i)
        if duration(note1)==duration(note2):           #note duration and timestep equal?
            rhythm++
        if note1.diffPrevious == note2.diffPrevious:   #difference to previous note equal?
            melody++
    total = max(musicalLength(melo1),musicalLength(melo2))
    return (1-weight)*(rhythm/total)+weight*(melody/total)

def findNote(melo,i):                                  #return first note found at timestep i
    for note in melo:
        if note.quantizedStartStep<=i && note.quantizedEndStep>i:
            return note
    return null
```

To calculate the SPS, shown in Figure 3.10, we first calculate the pair-wise similarity between all samples. We use dimensionality reduction, especially multidimensional scaling (MDS) [Kru64a; Kru64b], to project the similarities between samples to 2D points. These points are then drawn into a scatter plot (Figure 3.10), where each circle represents a melody sample from our dataset.

In our SPS, melody samples are represented as circles that encode different information in their color and radius. In Figure 3.10, the color encodes the hyperparameter temperature, where lower temperature corresponds to blue and higher temperature to red. The radius of a circle is fixed here but could be used to encode the number of notes or other properties.

We allow the user to use a resizable circular brush to select multiple samples or groups of samples for further investigation. These samples are shown in different types of aggregations (Figure 3.16), which are explained later.

Due to the limited amount of information, that could be encoded into a circle, we tried different glyph ideas to represent the melody of a sample [WCHU08]. These glyphs should help the user to get information about the sample inside of the SPS without actively selecting samples. When using glyphs, we found that overlapping glyphs in an SPS can result in clutter, where the user is unable to see any information due to occlusion. As a solution to that problem, we gridified the SPS in order to avoid collision of glyphs.

In a gridified scatterplot the space is divided into cells of the same size and the points are placed into separate cells, so each cell contains up to one representation of a sample. With that grid, the visibility of all glyphs is guaranteed, but similarities might not be shown correctly. This problem can occur when two points are close together and a third point is placed with higher distance towards the

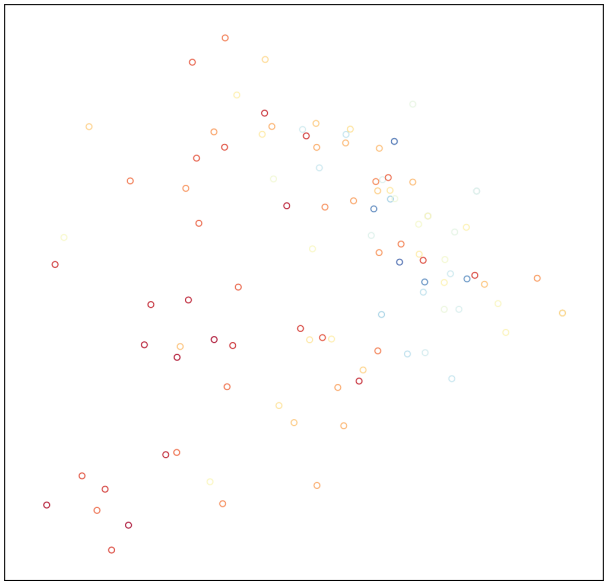


Figure 3.10: A similarity-preserving scatterplot (SPS) representing melody samples with a circle. The color encodes the temperature used to generate the melody, where blue indicates a low temperature and red a high temperature. Users can use a circular brush (gray circle) to select samples for different visualizations shown in Figure 3.16.

first two points. When using the grid in this scenario, all three points could end up in neighboring cells, which does not represent the true similarities. Nevertheless, we rate visibility higher than the fully correct accuracy so we chose to gridify the SPS when using glyphs instead of circles.

Pie Chart Glyph

A first idea is to represent each melody sample with a pie chart, showing the occurrences of pitches in the melody. To calculate the occurrences, the octave of a pitch is omitted, so C4 and C5 both count as a C in this occasion. The pitches are placed, starting at the top, clockwise in the circle and are encoded with a color from Tableau 10 (Figure 3.3) to identify the difference between sections. Since our color palette only has 10 colors but 12 pitches are used, we reuse the first two colors, as shown in the legend at the bottom of Figure 3.11. With that decision there is a possibility that two neighboring sections at the top could have the same color. Imagining a line from top to bottom, these sections can be can be separated, as shown in Figure 3.11 at the right. We also tried to encode the temperature of the sample as a colored dot in the middle, which was too confusing with the different colors.

With that representation of melody samples, the time aspect is omitted and the occurrences of pitches, independent from octaves, are presented. Some examples are shown in Figure 3.11. For example the left glyph shows that the melody mostly uses F#, G#, and E as pitches. These should help the user when searching or comparing melody samples by occurring pitches.

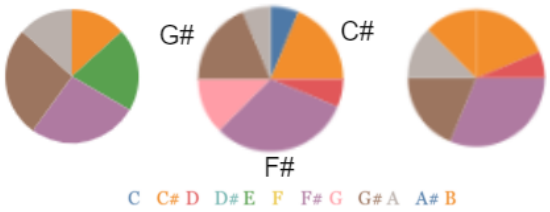


Figure 3.11: Melody samples are represented by a pie chart, where the occurring pitches are represented. A pitch is encoded by a color as shown in the legend at the bottom. The pie chart starts at the top and shows clockwise the occurrences of pitches in ascending order. For example the middle pie chart shows high occurrences for the pitches **C#**, **F#**, and **G#**, as represented by the larger, colored slices.

Starglyph

Because the pie chart glyph only showed one attribute of the melody sample we chose a different representation, to show multiple attributes at the same time. Therefore we chose a star plot as glyph [KHW09] to encode four attributes of the sample: *number of notes*, *similarity to the parent*, *variance of intervals between pitches*, and *mean length of notes*, arranged in the same order starting at the top and going clockwise. Some examples of these starglyphs are shown in Figure 3.12. The chosen attributes can be changed or a new one could be added, but we decided to use these four for different reasons. First, we want to achieve a higher visibility and avoid misunderstanding of the glyph, which is why we only chose a limited amount of attributes.

In order to find more complex or simplier melodies, the number of notes can be used to indicate that. A complex melody most likely consists of a higher number of notes due to either a faster or longer melody. The attribute mean length of notes can also indicate more exciting or calm melodies. Therefore, the combination of those two attributes can be used to indicate the following properties of a melody with a high chance:

- fast melody (high number of notes, low mean length of notes)
- slow melody (low number of notes, high mean length of notes)
- short melody (low number of notes, low mean length of notes)
- long melody (high number of notes, high mean length of notes)

A fast melody for example, consists of many short notes. On the other hand, a long melody could have many notes but also a high mean length of notes, although a long melody could also have many notes and a small mean length of notes. In this case the melody would also be fast which is indicated by the mean length of notes. So these attributes indicate likely properties of the melody, as mentioned previously, but can fail in some special cases.

The attribute *variance of intervals* between pitches should indicate, whether a melody mostly uses pitches around the same pitch or uses jumps between notes. This can indicate more exciting melodies as they would use a wider range of pitches. We chose similarity to the parent as an indicator on which melodies are small variations of the seed melody and which melodies are different ones. Due to the tree structure mentioned previously, a melody sample is not necessarily a direct continuation

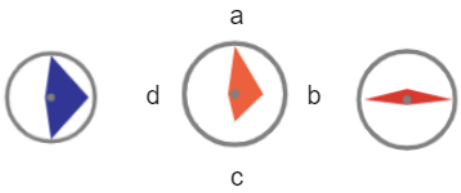


Figure 3.12: Starglyphs as representation for melody samples, encoding different metrics as contents. We chose four contents, which are ordered in the following way starting at the top and going clockwise: **a)** number of notes, **b)** similarity to parent, **c)** variance of pitch difference, and **d)** mean duration of notes. The color of the star corresponds to the temperature of the sample, where blue represents a low temperature and red a high temperature. For reference on the values the gray dot symbolizes the center and the gray circles symbolizes the range of values of the starglpyh.

of the composition. So we chose these attributes to indicate different aspects of a melody, but there could also be other different attributes that could help finding samples depending on different aspects.

The glyph itself shows a gray dot to indicate the middle as reference for the star and a gray circle to show the range of values and indicating the maximum. The color of the star also encodes the temperature of the sample: blue represents a lower temperature and red a higher temperature.

Reviewing the examples in Figure 3.12, the left starglyph could likely represents an exciting and fast melody, that was generated with lower temperature and is a small variation to the parent. On the other hand the right starglyph is likely to represent a rather slow and calm music due to a few long notes and small intervals between their pitches.

Histogram Glyph

Although starglyphs can encode more attributes than pie charts, we wanted to show more information about the melodic structure, which was missing in both previous glyphs. Therefore, we wanted to encode the occurrence of intervals between notes for a given melody sample as an aggregation of the melodic structure. We use a histogram, shown in Figure 3.13, to visualize the occurrences. An interval of -12 pitch steps is shown on the left of the histogram and is encoded with a blue color, while the opposite, +12 interval, is shown on the right in red color. The bar in the middle shows the number of 0 intervals in gray. The value of occurrences is shown by the height of a bar. In order to show the values in comparison to all melody samples, we first chose a global scale with the maximum height being the maximum occurrence over all samples. Especially for comparing samples, this scale helps due to the total value, as a higher bar always has a higher occurrence in comparison to a lower bar. A sample that is calculated with that scale is shown in Figure 3.13 on the right. Although a global scale enables better comparison, a single outlier can produce a non-optimal maximum which results in all other samples only showing small bars, which makes it harder to compare them due to visibility. To solve this problem we use a local scale were the maximum is calculated over all occurrences inside the sample. These local scales are shown in the examples in Figure 3.13 on the left and in the middle. This scale increases the visibility of each sample and their values, but reduces the comparability, due to lost context.

Figure 3.13: A single histogram glyph to represent melody samples showing the occurrence of pitch distance between the notes. The pitch distance is shown from left to right indicating a distance of -12 to +12, which corresponds to an octave. The color encodes the value of distance where blue is -12, gray is 0, and red is +12. **Left, Middle:** Local y-axis scale to reduce white space inside the histograms and higher visibility. **Right:** Global scale to show big spikes across all histograms.

Figure 3.14: Melody samples are represented by a double histogram showing the occurrence of pitch distance between the notes. Pitch distance is shown from left to right indicating a distance of 0 to -12/+12, which corresponds to an octave. The positive distance values (0 to +12) are shown on the top, while the negative distance values (-1 to -12) are shown on the bottom. The color encodes the value of distance where blue is -12, gray is 0, and red is +12. The y-axis scale is a local scale specialized on reducing white space inside the histograms.

Therefore, a higher bar in one sample could encode a lower value than a smaller bar in another sample. Investigating the example in Figure 3.13 on the left, the user is able to tell that there are few intervals between notes in both positive and negative direction, but most intervals between notes are 0 which results in the same note being played several times. This could likely refer to a more monotone melody, whereas the example in the middle, where the intervals are either high in both directions or 0, could refer to a melody with more changes.

Due to the layout of the histograms, the width of a bar is small and sometimes not easy to detect. To improve the visibility and the comparability of positive and negative intervals, we decided to combine two histograms. These double histograms are shown in Figure 3.14. The top histogram shows the occurrences of intervals with the value 0 to +12, while the bottom histogram shows the values -1 to -12 both from left to right. This also improves the understanding, as small intervals are shown to the left and large intervals to the right. Important to notice is that the intervals with the same absolute value, for example +8 and -8, are at the same position on the x-axis. So it is easier to identify the occurrences of intervals compared to intervals from the opposite side (positive or negative). Both scales mentioned previously are usable here, but only the local scale is shown in Figure 3.14 with regard to better visibility.

Figure 3.15: A piano roll glyph representing a melody sample with its notes to show the melodic structure of the sample. Piano rolls are represented in a simple way without axis to reduce clutter, similar to the representation in Section 3.4.3. The background color of a piano roll encodes the temperature used to generate the corresponding melody sample, while blue equals to a low temperature, orange to a medium temperature, and red to a high temperature. Note color is chosen by the brightness of the background color to achieve a higher contrast.

Piano Roll Glyph

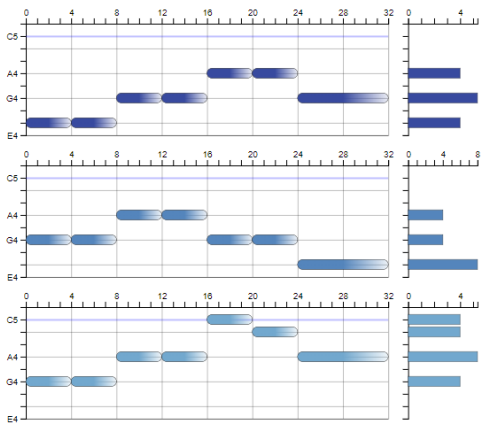
Finally, we considered to visualize the melody itself in the form of piano rolls, as shown in Figure 3.15. The user is already familiar with this concept and can use it to investigate patterns in the melodic structure, especially when comparing different melody samples. Piano rolls may also help the user to get a good feeling for the melody, which can improve the selection of samples. The notes are drawn on top of a colored rectangle and use a local scale to use the maximum space in order to reduce white space and improve visibility. The background color of a piano roll encodes the temperature used to generate the specific melody sample. With the changing brightness of the background color, the color of the notes also changes to stand out from the background. For example in Figure 3.15, three samples are shown with low, medium and high temperature (left to right) each of them showing a different melodic structure, while the sample on the left and the right show a similar start of the melody.

Statistical Aggregations

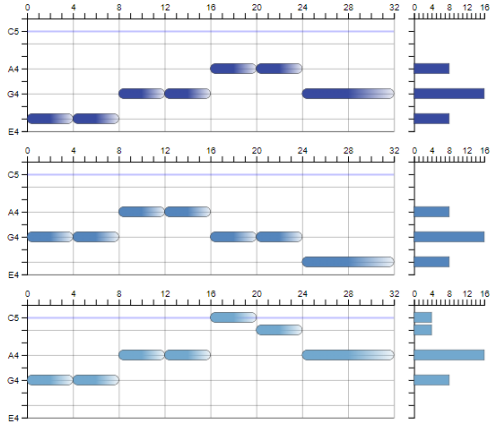
As mentioned previously, we allow the user to select samples with a circular brush in the SPS. These samples are shown in different visualizations at different levels of aggregations (Figure 3.16) to support investigating the group of samples and finding patterns. The data used is different for each of the shown visualizations (except for **a)** and **b)** to show diverse, interesting cases.

a),b Each selected sample is represented in a separate piano roll. The histograms on the right of the piano rolls aggregate the notes for a faster overview of the sample with the possibility to use different x-axis. This visualization should help the user to investigate single samples of the group.

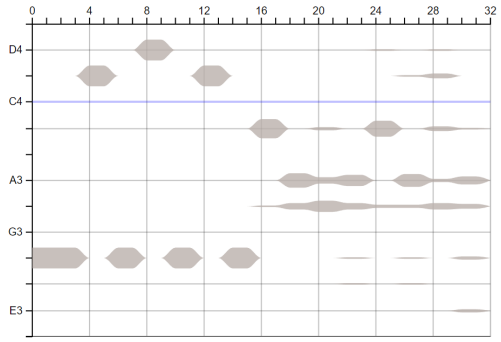
c) An aggregation of the notes of all selected samples showing the density for pitch and time. A similar visualization is shown in a paper by Heyen et al. [HS20]. The number of notes for a certain pitch at the same time are added up for the density. We chose this type of visualization to support the user when investigating the used notes of all selected samples, for example to get insights into how the AI generates samples. A finding in this example would be that all selected melody samples start with the same note sequence for the first 16 steps.



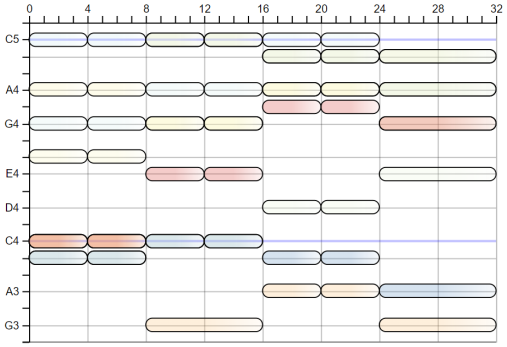
(a) Histogram showing the mean duration of notes for a certain pitch.



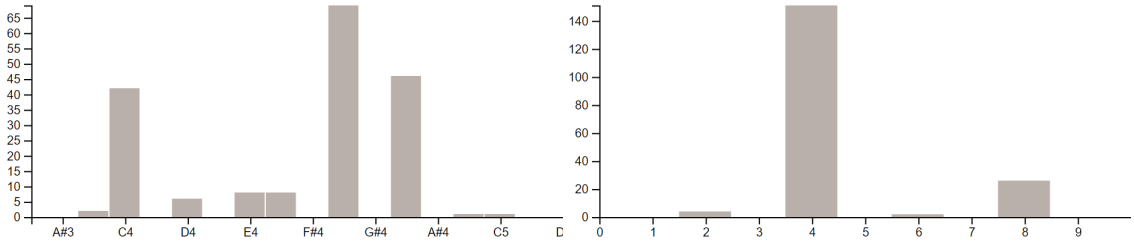
(b) Histogram showing the combined duration of notes for a certain pitch.



(c) Combined density for certain pitch and time.



(d) Notes of multiple samples in one piano roll.



(e) Histograms showing the occurrence of pitches or note duration for all selected samples. **Left:** Occurrence of pitches. **Right:** Occurrence of note duration.

Figure 3.16: Different visualizations at different levels of aggregations of the data for selected samples. **a), b)** Each sample is represented in a separate piano roll. The histograms on the right of the piano rolls aggregate the data for a faster overview, using different x-axis scales. **c)** An aggregation of the notes of all selected samples showing the density for pitch and time. **d)** All notes of the selected samples are shown in the same piano roll. **e)** Two histograms with the occurrence of pitches (left) and occurrence of note's duration (right).

d) An aggregation of the notes of all selected samples showing all notes in the same piano roll. Notes are represented more transparent to show overlapping. This visualization is similar to the density visualization but emphasizes on the length of notes instead of the number of notes at the same time step. Here, the length of the notes is visible in comparison to the density visualization. A finding in this example would be that all selected samples end on a half note.

e) More aggregation than c) and d) shows two histograms with the occurrence of pitches on the left and occurrence of note's duration on the right for the selected samples. This can help to identify often used notes, which can be part of a specific key or chord. The duration gives a rough indication about how exciting or calm the melodies might be. A realization in this example would be that C, G, and A are the most used pitches and most of the notes are quarter notes with some half notes. Therefore, the selected samples are rather slower and calm.

3.5 Implementation

The presented approach with all visualizations is implemented as a web application and can be used in a browser like Google Chrome, which was used during testing and development. Different browsers were not tested and could therefore produce issues during run time. The intended screen size is $1920x1080$ pixels while smaller resolutions or different aspect ratio could lead to misaligned visualizations and therefore worse usability.

Sharing this approach and making it available for many users is one of the main reasons for the decision to implement it as a web application. JavaScript is one of the most popular languages for web application implementation. We chose JavaScript as a language because it supports many different popular libraries for user interfaces (UI), server run time, visualization, utility tools, as well as music generation through AI.

React (reactjs.org) is a popular library for building user interfaces in JavaScript. Views consist of multiple components that can work independently from each other and manage their own states. Therefore, more complex and nested user interfaces are possible. State management of a component is used to store data and trigger re-render methods when the state changes. Therefore, changes for example triggered by the user, update the view for an interactive application. In this thesis, the version 17.0.2 of React is used.

D3 (d3js.org) is a library in JavaScript used for data-driven visualizations. With D3 it is possible to create shapes, style them, and combine them for powerful visualizations. Visualizations are data-driven and therefore transformable depending on the used data. The version 8.0.0 of D3 is used here.

Magenta.js[1] [RHS18] is a JavaScript suite providing an API for music generation and art through ML models provided by Magenta. Different pre-trained models can be accessed this way and used for predictions, for example to generate music. In this thesis Magenta.js is used for communication between the user and the AI to get multiple predictions for music generation. The version 1.21.0 is used.

[1] https://github.com/magenta/magenta-js/tree/master/music

DruidJS[2] [CKS20] is a JavaScript library for dimensionality reduction. It provides many techniques, especially MDS, t-SNE, and UMAP. MDS is the main technique used in this thesis for the similarity-preserving scatter plot and visualizing distances from a higher dimensional space in 2D. The version used here is 0.3.16.

Musical Instrument Digital Interface (MIDI) [Moo86] is a file format, designed to store notes and the properties of a music piece, as well as information about different voices and instruments. It is specialized on storing music, so manipulating single notes or the instrument playing a sequence is possible. Therefore, MIDI is often used in digital music composition, while we use web MIDI as input for a starting melody. In this thesis, the composition can be exported as MIDI to provide compatibility to other applications.

3.5.1 Data Structure

Musical notes are stored as objects with a pitch, a start time and an end time. The pitch of a note is stored with the MIDI value, which is between 0 and 127 and corresponds to the key of a note. Therefore, the value 0 is equal to C0 and 60 to C4. The start and end time is stored in a quantized way, where time is represented as steps with equal duration instead of seconds. In this thesis, one step equals the duration of a single sixteenth note. Therefore, a quarter note's duration is equal to four steps. A melody is stored as an array containing the note objects belonging to that melody as shown in Listing 3.2.

Listing 3.2 A melody consisting of a C4 quarter note followed by an E4 half note, stored in an JSON-like structure.

```
melody = [{
    pitch:60,
    quantizedStartStep:0,
    quantizedEndStep:4
},{
    pitch:64,
    quantizedStartStep:4,
    quantizedEndStep:12
}]
```

[2] https://github.com/saehm/DruidJS

4 Evaluation

We evaluated the system and its visualizations with case studies and a small pair analytics study. In the case studies we looked at specific cases and the resulting visualizations to find interesting patterns, missing features and tested the scalability of our approach. We had two participants in our pair analytics study, where they tested the approach and gave us feedback on the usability and the positive and negative aspects, as well as their feelings towards composing with our approach.

4.1 Technical Evaluation

In this section we evaluated the scalability of our approach, looking at run time and visualizations separately. The run time should indicate the usability, in terms of reaction to actions taken by the user, depending on different tasks and number of samples. Visual scalability should indicate the usability of the visualizations, depending on the number of generated melody samples.

4.1.1 Run Time

We evaluated the run time of our approach by measuring the time for different tasks with different amount of melody samples. The testings were made in the browser Google Chrome (Version 95.0.4638.69) on a computer with 8GB DDR3 RAM and an Intel i5-4690 CPU processor.

We chose different numbers of samples, reaching from 3 to 1000, but we focused on smaller amounts first. The chosen values are 3, 10, 50, 100, 200, and 1000. The values 3 and 10 were chosen to represent a small amount used for the icicle and the node-link diagram, while 50, 100, and 200 were chosen as possible use cases for the similarity-preserving scatterplot (SPS). The value 1000 should indicate the run time for a large amount of data.

Samples	3	10	50	100	200	1000
Times (seconds)	1.3	3.2	14	38	130	1350
Seconds per sample	0.43	0.32	0.28	0.38	0.65	1.35

Table 4.1: Run time of generating melody samples and visualize them.

At first, we measured the time for generating a certain number of samples and then calculating all visualizations depending on that data. The values (Table 4.1) show that a small amount of samples up to 100 samples can be generated and visualized in under one minute. Afterwards, the times increase fast, where it takes several minutes to generate many samples.

After generating samples, the user is able to interact with the samples and its notes in different ways. One way is to drag notes in order to adjust the melody. For this task we measured small numbers for all amounts of samples, which indicated real time. It is important to know, that after dragging a note, the visualizations are refreshed, which can then result in extra time to show visualizations.

Another task we measured was adding a sample to the composition. Here, the maximum time measured was 2.2 seconds for 200 samples, but all other cases of below 100 samples showed a near real time task finish with under one second.

Measuring the sorting metrics from the node-link diagram, we needed a maximum of two seconds for 50 samples. Due to a large amount of samples inside one level, the diagram is not shown for 100, 200, and 1000 samples and therefore is not used in this measuring.

The overall response time to clicks and interactive usage is real time for under 200 samples, but increases a bit for 200 samples. When using 1000 samples, the response time greatly increases and results in long waiting times.

It is important to notice that the system is not optimized for run time, which results in longer waiting times for specific tasks. Overall, the measured times showed smooth usage of the approach for a maximum of 100 sample. Between 100 and 200 samples, the response time is a bit longer but still usable with short waiting times. At 1000 samples, the approach has high response times and therefore takes long to use.

4.1.2 Scalability of Visualizations

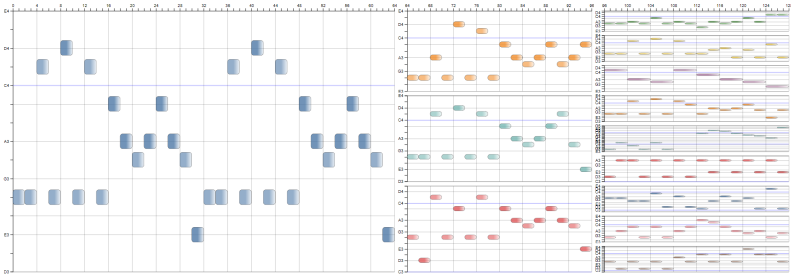
After measuring the time of different tasks for a run time evaluation, we took a look at our visualizations and evaluated the scalability of these with different amount of samples or levels.

Icicle Plot

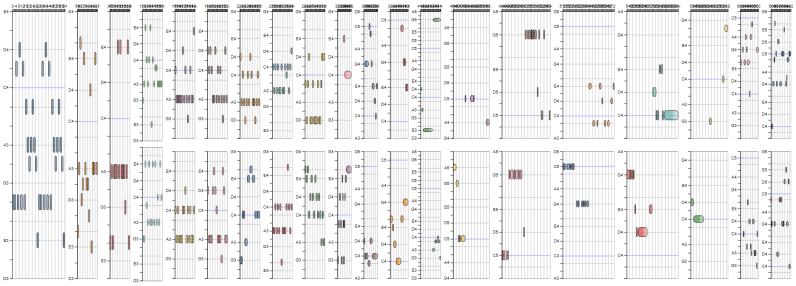
We evaluated the icicle plot using different amount of levels, for scalability in x-direction, and different amounts of samples, for scalability in y-direction. In Figure 4.1a a case is shown were the icicle works as intended and described in Section 3.4.2. It is important to note that even if some small clutter occurs, the basic functionality is still given.

When generating multiple levels and visualizing them in the icicle plot, the visibility is dependent on the length of the samples. In Figure 4.1b we generated 20 levels with the first sample being 64 time steps long and all other samples being 32 time steps long. With these settings, the x-axis labeling overlaps and results in clutter, so identifying the exact time step of a note is not possible for most time steps. Due to the limited space, all notes are compressed which can result in a note only being a single line. Some of these cases are shown in Figure 4.1b. Without zooming in, short notes can be hard to identify when many short notes occur in quick succession. Overall, the icicle plot is readable up until 20 levels, depending on the length and melodic structure of the melody samples.

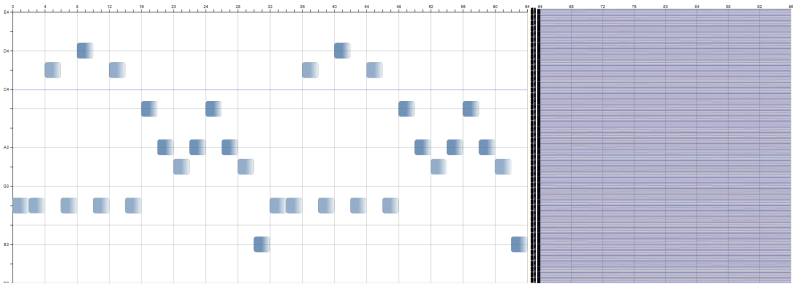
With many samples inside a single level, the limited space gets smaller for each sample. With 50 generated samples in a single level, as shown in Figure 4.1c, the icicle plot produces clutter, where no notes are visible because the space for each node is too small and overlaps. Therefore, the scalability is limited to a few samples in the same level. The visibility of samples is dependent on the range of



(a) Icicle plot with 3 levels and up to 9 samples on a level.



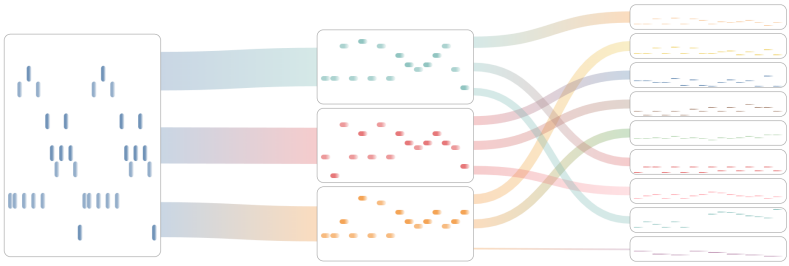
(b) Icicle plot with 20 levels.



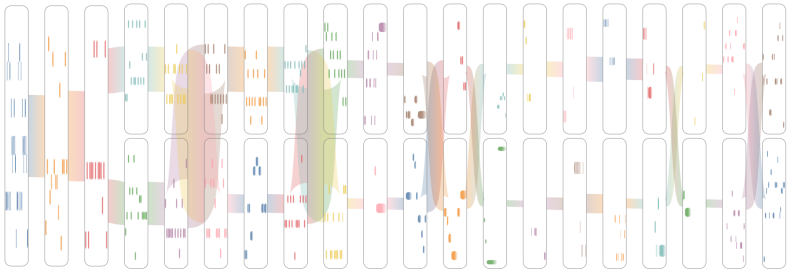
(c) Icicle plot with 50 samples on the same level.

Figure 4.1: Icicle plot with different amount of levels and samples, to test the visibility of the visualization. **a)** All information are visible. **b)** Notes get compressed in x-direction, making it hard to differentiate short notes. **c)** Results in clutter due to limited height of nodes.

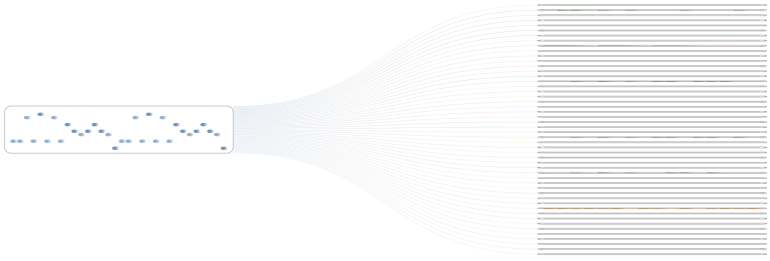
pitches and chosen type of scale used for the axes. Overall, the icicle plot is readable up until 15 samples on the same level, but with more than 15 samples notes are no longer distinguishable. As already stated in Section 3.4.2, we presented an alternative idea (Figure 3.7) to increase scalability depending on the number of samples. Although this alternative worked for 50 samples, it struggled when using 100 samples in the same level.



(a) Node-link diagram with 3 levels and up to 9 samples on the same level.



(b) Node-link diagram with 20 levels.



(c) Node-link diagram with 50 samples on the same level.

Figure 4.2: Node-link diagram with different amount of levels and samples to test scalability of visualization. **a)** All information are visible. **b)** Notes get compressed in x-direction, making it hard to differentiate short notes. **c)** Results in clutter due to limited height of nodes. No information is obtainable.

Node-Link Diagram

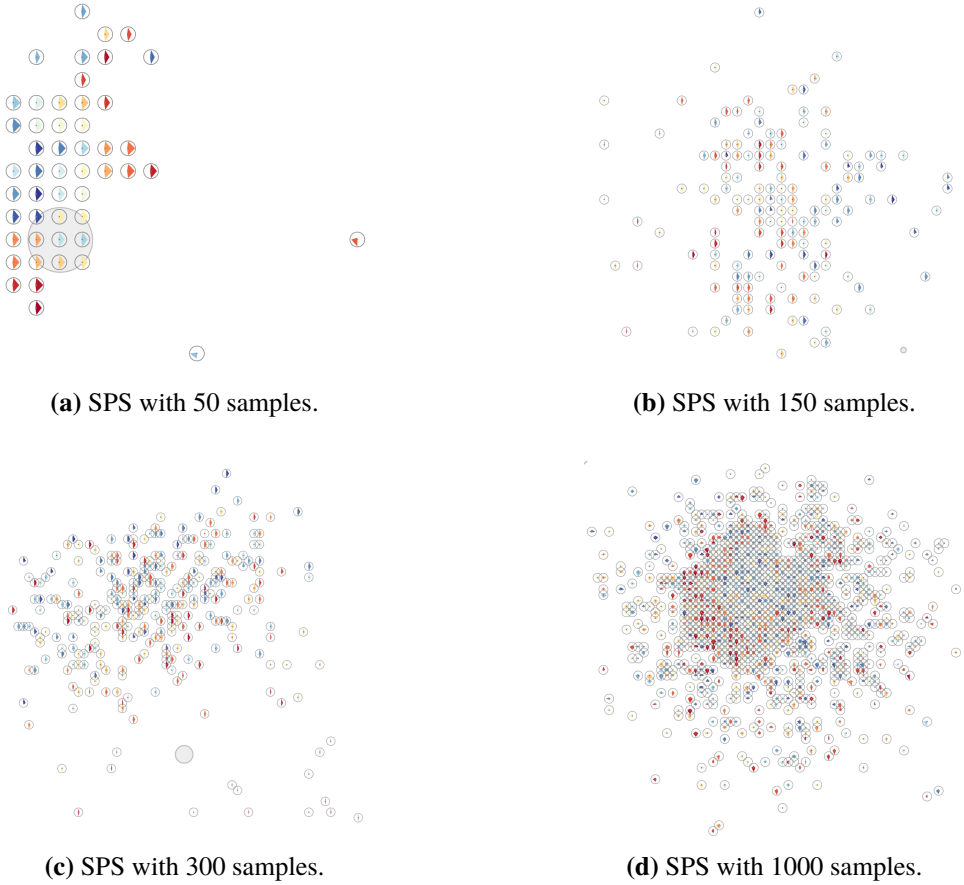
Compared to the icicle plot, the node-link diagram shows no axes and therefore reduce clutter when scaling. Instead the added links have to be evaluated.

In Figure 4.2a shows an example, where the node-link diagram works as intended and visualizes the links with different widths and sorted nodes, as described in Section 3.4.3. Here, all links, nodes, and their corresponding notes are visible.

We again generated 20 levels to test the scalability in x-direction (Figure 4.2b). Due to limited space between nodes, bigger link crossings result in overlapping of links and nodes, as seen between the levels 5 & 6, and 8 & 9. Therefore, the readability of notes in the affected nodes decreases and notes can be misinterpreted, but should not occur in practice, as users are unlike to generate this many

levels before deciding on a sample or branch. The limited space also affects the nodes, which are visually compressed, resulting in even more compressed notes. Short notes in quick succession can then be hard to identify, which is shown at the left at the start of the melody. Again, the visibility of notes is also dependent on the music structure, as well as the length of a melody sample visualized in a node. Due to the links between nodes, the scalability in x-direction is lower than the scalability of the icicle plot, but the node-link diagram is still usable with at least 10 levels, depending on the length and structure of the respective melodic samples and the link widths.

To evaluate the scalability in y-direction, we generated 50 samples in the same level, as shown in Figure 4.2c. The same problem as in the icicle plot occurs, where the space for each node is too limited and notes are not visible or identifiable. Same scalability realizations apply for the node-link diagram and the icicle, because links do not affect the scalability in y-direction. So the node-link diagram is only usable for a few amount of samples per level.



(a) SPS with 50 samples.

(b) SPS with 150 samples.

(c) SPS with 300 samples.

(d) SPS with 1000 samples.

Figure 4.3: Similarity-preserving scatterplot showing starglyphs in a grid with different number of samples. **a)** Large glyphs due small number of samples. **b)** All glyphs are smaller due to finer grid. **c)** Overlapping due to smaller grid. Local information can get hard to read. **d)** More overlapping due to even smaller grid. Local information is lost, while global clusters are visible through color. For example, area of blue glyphs in the middle with red glyphs around them.

(a) SPS with 300 samples.

(b) SPS with 1000 samples.

Figure 4.4: Similarity-preserving scatterplot showing starglyphs with different higher number of samples. Glyph size adapts to the grid size. **a)** Showing small glyphs due to fine grid without overlapping. **b)** Even finer grid and therefore smaller glyphs without overlapping.

Similarity-Preserving Scatterplot

We evaluate the scalability of the similarity-preserving scatterplot (SPS) and the glyph size, by taking a look at the representation of 50, 150, 300, and 1000 melody samples. For 50 samples the glyphs were big enough for good visibility and had no overlapping when using the gridified version (Figure 4.3a). The size of glyphs decreases when using more samples as the grid gets finer. The glyphs are still recognizable, as shown in Figure 4.3b. So the SPS works as intended, as described in Section 3.4.4 for a maximum of 255 samples.

First problems occur using more than 255 samples, as shown in Figure 4.3c, where the grid gets even smaller but the size of the glyphs is not decreasing. This results in overlapping glyphs and therefore a worse visibility, as it is harder to analyze the glyphs.

This problem gets worse with more samples as the grid gets finer. As shown in Figure 4.3d, where 1000 glyphs are placed in a fine grid, the glyphs overlap even more making it hard to evaluate the glyphs. Visibility is not given when using this large number of samples. Therefore, the scalability of the SPS allows a maximum of 255 samples without overlapping glyphs and 511 with smaller overlapping glyphs, where details aren't readable, but coarse overall patterns are still visible, such as clusters of red color in Figure 4.3d.

We addressed this problem and adapted the size of a glyph to the space of a cell in the grid. Therefore, no overlapping glyphs occur when using the gridified version. As exchange, the size of the glyphs might get too small to read all information, especially when the encoded values are low. We tested these changes with 300 and 1000 samples (Figure 4.4), because these values caused overlapping previously.

The aggregation visualizations do scale well, even with 1000 samples, as they only change scales on the axis. As we only show the top five separate piano rolls beside the SPS at a time, the amount of samples exceeding five does not limit the visualization, as the user can scroll through the rest.

Showing the notes of all selected samples in the piano roll, many notes overlap or overwrite each other, resulting in difficulties to see patterns. The histograms and the density piano roll only change the scale of the occurrences, which isn't changing the actual view of these visualizations.

4.2 Case Studies

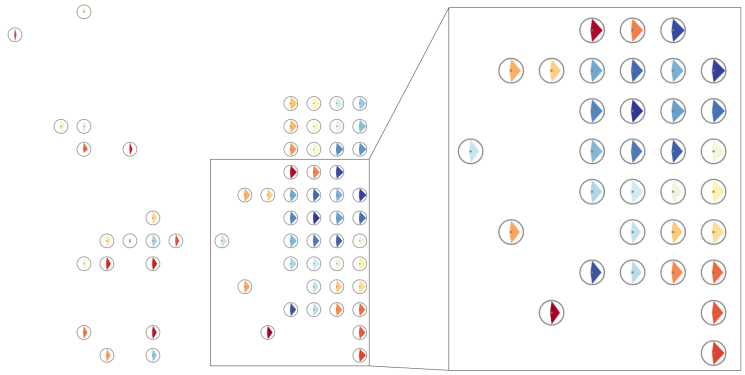
In this section, we evaluated different cases and inspected the different glyphs in the SPS. We especially took a look at groups showing similar glyphs that indicate small variations of a similar melodic sample. Other interesting samples were glyphs, showing different information than other glyphs in the SPS. We evaluated these glyphs and looked for interesting patterns and information inside the SPS.

4.2.1 Starglyphs

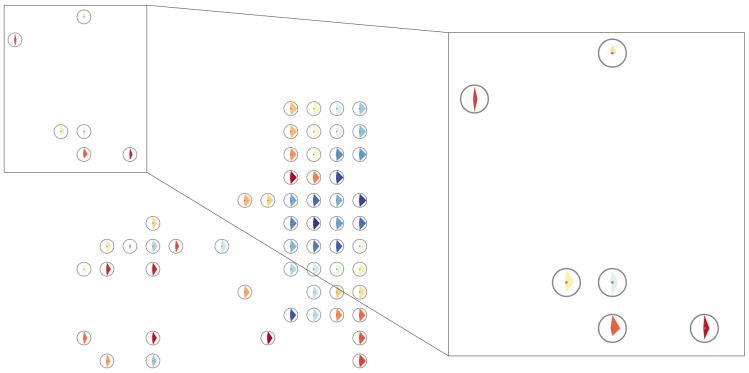
First, we took a look at the starglyphs, encoding the four attributes *number of notes*, *mean duration of notes*, *variance of intervals*, and *similarity to the parent*. In Figure 4.5a, we took a closer look at a group of similar samples, which is highlighted. The similarity of the samples should be recognizable through the distances between the samples but can be confirmed quickly by the shape of the starglyphs. Analyzing the starglyph, all these melodies contain a high number of short notes with high *variance of intervals* and a high similarity to the parent. This indicates exciting and lively melodies, like the used parent melody. Therefore, a high similarity to the parent can be confirmed. It is important to recognize, that although the shape of the glyphs seem identical at first, they can vary slightly when taking a closer look. The glyphs do not all encode the same similarity to parent value, as the deflection to the right varies. This can be an indicator towards some of the samples being variations of the parent melody.

As the colors indicate the temperature of the ML model, most of the similar samples are generated with a lower temperature and therefore less randomness. This information can be used by the AI analyst to recognize that a lower temperature produces a more similar melody sample. Most of the blue samples are closer together than the red samples, which could emphasize the previous realization. Still, this could also be an artifact the used grid, where distances might not be shown correctly. The user could verify similarity by showing the samples in detailed visualizations.

When taking a look at samples that are further away from the previous group, as highlighted in Figure 4.5b, the shape of the glyphs starts to vary much more. Especially the two samples at the top of the cutout show a low similarity to the parent, indicating completely different melodies. Still, all of these shown glyphs indicate a high number of short notes, so the influence of the seed melody can be stated with these attributes. This findings can help the AI analyst when looking for relationships between the generated samples and the seed melody. Important to recognize is that most of the outlier samples with different melodies are generated with higher temperature (red color). The starglyph are especially helpful at showing multiple attributes at the same time and enables comparing different samples by these attributes.



(a) Group of similar samples.

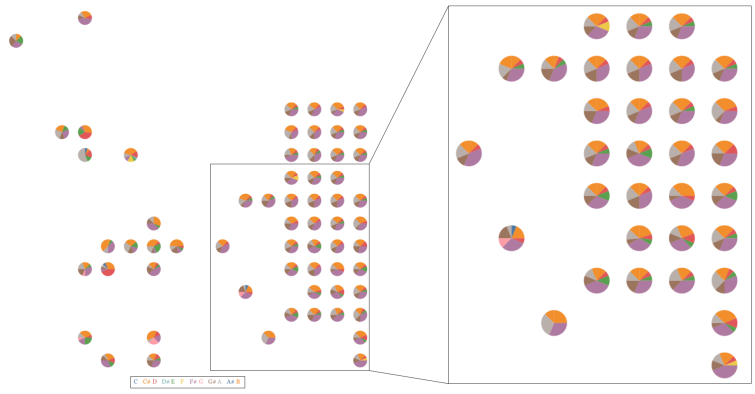


(b) Different samples.

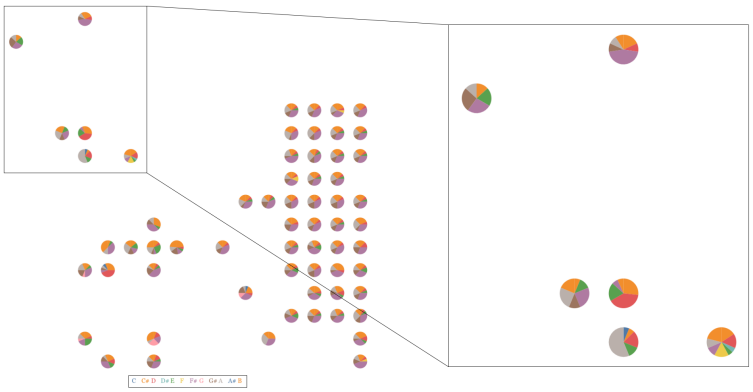
Figure 4.5: Highlighted part shows interesting sample groups, represented by starglyphs. **a)** These melody samples all contain a large number of short notes with high variance and a high similarity to the parent. In this case, the color indicates that a higher temperature leads to more variation as red glyphs are more at the edges or separately. **b)** Those glyphs indicates different samples with some similarity towards neighbors but large differences towards the cluster from a). Here, the color indicates that a higher temperature produced more different samples compared to the seed melody.

4.2.2 Pie Chart Glyphs

Next, we analyzed the pie chart glyph inside the SPS, using the same starting position as with the starglyphs. First, we analyze the group of similar samples, highlighted in Figure 4.6a, by taking a look at the pie charts and their occurrences. Most of the pie charts show a similar sectioning of the colors, which correspond to the occurring pitches. The main colors shown are orange, purple, and gray with some more colors occurring sometimes or as small parts. So the main notes are C#, F#, and A, which are also the most used in the seed melody, showing the similarity. With the pie chart glyph, it is easier to see that some of the melodies vary due to small changes in the sections and small differences on the occurring pitches. For example, some samples in the middle of the cutout



(a) Group of similar samples.



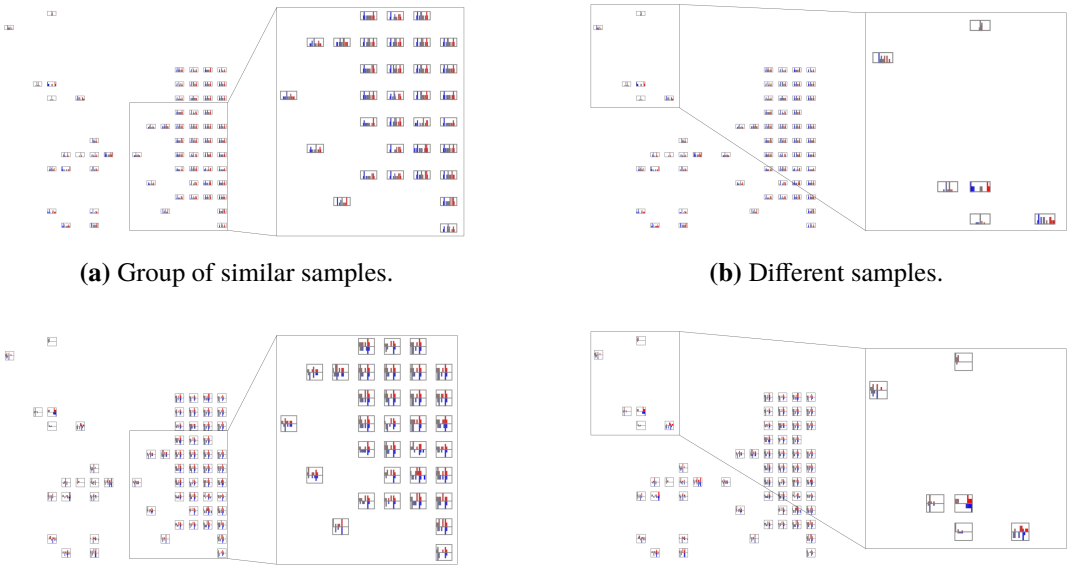
(b) Different samples.

Figure 4.6: Highlighted part shows interesting sample groups, represented by pie chart glyphs. **a)** The pie charts show similar distributions of pitches. Most pie charts show the colors orange, purple, brown, and gray but differences and additional pitches can be seen in different pie charts. **b)** These samples show different distributions of pitches compared to each other. The different occurrences of pitches indicate different melody samples in terms of used pitches.

have an additional green section, while some others own a bigger red or purple part. It is interesting to see that the samples closer to the edge can show an additional color like blue or yellow, which indicates some randomness.

Taking a look at the glyphs with higher distance towards the similar grouped samples, the colors vary much more (Figure 4.6b). These samples show a completely different sectioning compared to the previous samples. Especially the samples at the bottom of the cutout show multiple differently colored parts, indicating a higher number of different pitches in the melody.

Although the pie chart shows the occurring pitches of a melody sample, it is harder to read information from this glyph. Most of the information that can be extracted, are small variations between melodies in terms of small changes of the pitch occurrences. The color supports noticing the size and occurrences of sections and allows to do this more quickly.



(a) Group of similar samples.

(b) Different samples.

(c) Group of similar samples.

(d) Different samples.

Figure 4.7: Highlighted part shows interesting sample groups, represented by our two versions of histogram glyphs. **a,c)** The samples show similar occurrences of pitch distances. Most of the highlighted samples show a similar distributions of blue, gray, and red bars. **b,d)** Some samples show large distances (blue and red bars at the edges of a histogram) while others show mainly small to no pitch distances between notes (gray bars in the middle of a histogram).

4.2.3 Histogram Glyphs

Using the histogram glyph (Section 3.4.4) the user can analyze the occurrences of intervals between notes, which could indicate exciting and lively melodies. In Figure 4.7a and Figure 4.7c, a group of similar samples is highlighted. The histograms show a similar distribution of occurrences of intervals, with small changes, indicating small variations. For this example, we used the local axis for better visibility, so all bars are shown and not too small. Analyzing the visible bars, most of the samples show bigger spikes around the 0 and +/-7 step intervals, which indicates an exciting and lively melody like the seed melody. The parent melody starts with a sequence of alternating notes, where the interval is around 7 steps large, which is represented well by the spikes in the histograms, showing the similarity between the samples and the parent.

Comparing the two versions of histograms, it is easier to see the previous realization in the double histogram due to the alignment of positive and negative bars. The single histograms save space due to smaller size but visibility decreases in this scenario. Looking at different samples, some bars are overall smaller due to a higher maximum in the sample and others showing a few high bars, for example the two samples closest to the bottom left. The values encoded in the bars can be more similar or different than appearing in the visualization because of the local scale, but it is important to notice that the coarse view can indicate interesting patterns.

Analyzing outliers, which are further away from the previous group as highlighted in Figure 4.7b and Figure 4.7d, different findings can be made. Some of the samples highlighted consist of bigger intervals and some mostly have no or small intervals. For example the double histogram in the middle shows bigger bars in red and blue on the right of the glyph, indicating the big intervals and a potentially exciting and lively melody. In contrast, looking at the two samples in the middle row at the top and bottom, the histograms show big bars around the interval size 0 and a few small bars for larger intervals. Therefore it is likely that the melodies are calm and more monotone, as they only have a few larger intervals, but mostly consist of neighboring notes, which we confirmed by listening to them. The interval 0 occurs in a sequence of notes with the same pitch, which sounds monotone. This finding is easier to make with the single histogram glyph, as the relevant intervals are placed in the middle of the glyph.

In conclusion, the usage of the single histograms supports looking for the occurrences of smaller intervals and therefore calmer melody samples. The double histograms help with finding more lively melodies as it's easier to compare positive and negative intervals. Large intervals are identifiable through their colors and therefore easier to detect, making it overall easier to find lively melodies compared to monotone melodies.

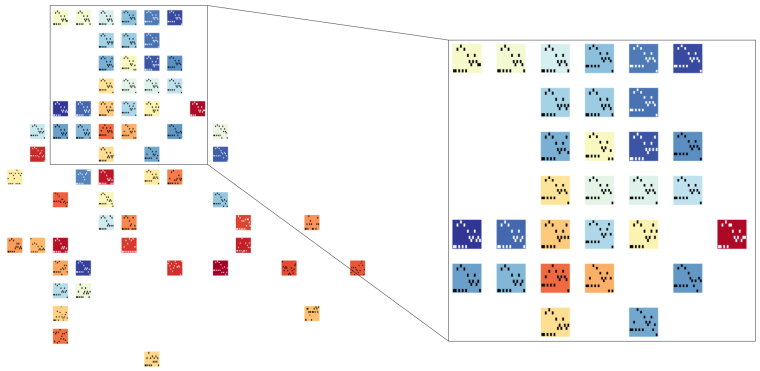
4.2.4 Piano Roll Glyphs

At last, we presented a simple piano roll as glyph to directly analyze the melodic structure of the samples or searching for interesting looking melody samples. In Figure 4.8a, a group of similar samples is highlighted, showing a similar melodic structure compared to the seed melody[1] as well. Taking a closer look at the melodic structure and the notes of the highlighted samples, we found that almost all samples start with a similar sequence of alternating notes. As mentioned in previous glyph analyses, the seed melody has the same melodic structure.

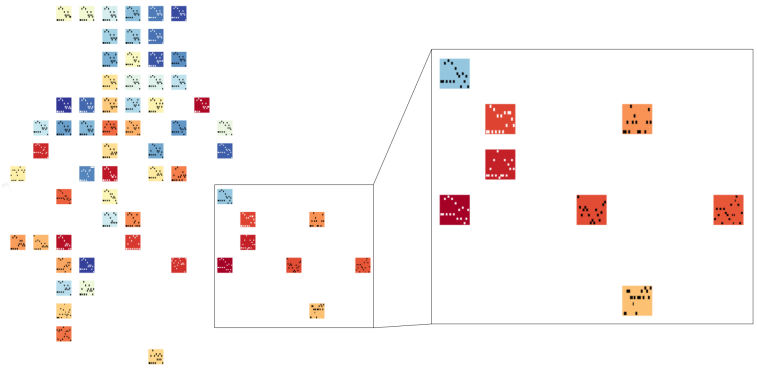
Comparing different samples from the cutout, small differences especially at the end of the sample indicate small variations. For example, comparing the two samples in the top left corner, we saw that the two samples are the same except for the last note, which also indicates a high similarity between those two. This also supports the functionality of the SPS, showing similar samples near each other. The background color of a sample indicates the temperature, which was used to generate the sample. As most backgrounds are blue or yellow, the AI analyst can see that most similar samples are generated with lower temperature, validating the assumption that lower temperature leads to less randomness.

Taking a look at outlier samples, as highlighted in Figure 4.8b, some of these show completely different melodies. For example, the samples closest to the bottom right corner of the cutout show a different melodic structure. In comparison to that, the melody samples close to the top left corner show a similar structure at the start (alternating notes) as the previous shown similar samples. Therefore, the position is justified, being closer to the other samples with higher similarity. Here, it is important to note that most outliers have a red background, indicating a higher temperature and therefore more randomness in the melodies.

[1] Intro of Crazy Train by Ozzy Osbourne



(a) Group of similar samples.



(b) Different samples.

Figure 4.8: Highlighted part shows interesting sample groups, represented by piano roll glyphs. **a)** Most of the samples show a similar melodic structure, where a few notes vary resulting in small variations of the parent melody. **b)** Outliers are mostly high temperature samples and show completely different melodic structures, resulting in different melodies.

In conclusion, the different glyphs can indicate different attributes of the melody to support the user in comparing these samples. When analyzing the samples and the similarities of the glyphs, we saw that our similarity function and the arrangement in the SPS show the similarities well. Although some samples might not be placed perfectly or the similarity could show weaknesses, the position of most samples seem to correspond to similarities well in our cases. It is important to know, that we did not evaluate the arrangement in the scatterplot and the similarity function to an extend, where we could state that they work for a high amount of cases.

4.3 Pair Analytics Study

To get feedback from actual users and evaluate usability and ease of understanding we evaluated the usability of the system, usage of visualizations and sorting metrics for different tasks, and the understanding of the visualizations and glyphs with the help of a pair analytics study. It is important to notice that the system needs some time to learn, but with the limited time we had this was not part of the study, as it would take multiple weeks or months of using. Because of that we decided to conduct the pair analytics study, as learnability of our approach is not part of the study.

The study was conducted with two participants with limited music knowledge for one hour and is therefore not as representative due to a low sample size and missing experts. We let the participants work with our approach and told them to compose a short melody and to analyze how the AI generates samples depending on the hyperparameters. Following these testings, we interviewed the participants about their feelings and findings when working, the overall usability, and possible improvements.

Both participants were not involved in the design and have little experience with music theory. One of the two participants had a generally good understanding of AI but not in a musical setting, while the other participant had limited knowledge about machine learning models and their functionality. So both participants fit into the idea of an amateur composer, while regarding the AI analysis, different starting points of knowledge about machine learning were given, but both participants had little knowledge about music generation.

4.3.1 Findings regarding our Visualizations

Both studies we followed the same procedure with both participants, as we firstly explained the general idea and some controls to the participants. Next, the participants generated melody samples and continuations and inspected them with the help of our visualizations and sorting functions. Afterwards, they also used the fill-in function to refine some parts, that they did not like as much.

In the beginning, both participants took some time to get used to the controls and possibilities, but after a small amount of time they were more willing to experiment. One of the user was eager to explore the different functions as quickly as possible and clicked on several buttons to test their functionality. This can also be a consequence of the shortened introduction, but showed that our approach can be engaging.

After generating the first continuation samples, both participants used the listening functions more than other visualizations and sorting metrics to get a feeling for the generated melodies. They especially used the brush function to save time instead of listening to the whole path multiple times.

After getting used to the visualizations and the sound of generated melodies, the participants focused on using visual information as well. Both participants used the different sorting metrics in the node-link visualization to find samples. First, they sorted by *temperature* and the similarity to the parent. When using the *temperature* sorting, they quickly recognized a relationship between the chosen temperature and the subjective quality of a melody, as the melodies with lower temperature showed better potential when looking for samples that are similar to the previous melody. In

comparison, the participants were sometimes surprised, in positive and negative ways, by melodies generated with a higher temperature and stated a relationship between the randomness and the high temperature.

Later, both participants used the *variance of interval* sorting metric to find lively melodies. They stated that the sorting metrics helped them with quick comparisons and a faster selection compared to listening to all samples. Especially the visualization of the melody helped with filtering boring looking melodies, while the sorting metrics helped them find a matching sample for their imagination. Both stated that the *temperature* metric helped most when analyzing the AI, while the *similarity* helped the most when looking for a continuation with small variations. As previously said, they found the variance metric most helpful for comparing lively melodies.

Both participants used the node-link diagram more often for sample selection, compared to the icicle plot, due to the possibility to sort samples and the simpler look. The icicle was not as interesting to them, but both stated that it can be more useful, as it is more detailed and the actual timings and pitches are displayed. Especially for a user not as familiar with music theory, the look of a melody and the feeling for the image was more helpful and therefore the node-link diagram was also used due to the less detailed look.

Later, we asked the participants to take a look at the similarity-preserving scatterplot (SPS) and shortly explained the different glyphs were shortly explained. Both were overstrained with the SPS at first, but as we did not want to evaluate learnability, this is only a side note. The participants stated, that the SPS helped with analyzing the relationship between temperature and the generated samples and their attributes. Using the glyphs, for example the starglyph, multiple attributes of the samples can be shown at once, which was well received by our users to extend on the sorting functions from the node-link diagram. Both participants also mentioned that using glyphs and the SPS needed a bit of imagination, especially when swapping glyphs. One participant found no use for the histogram glyph, as they could not find any purpose analyzing the occurrences of intervals for his selection of samples. In comparison to that, they especially liked the starglyphs due to the multiple encoded attributes and the piano roll glyph, where they could get a good feeling of the melody, similar to the usage of the node-link diagram.

Composing music and using the SPS as support for sample selection felt a bit laborious and not as intuitive for the participants, but they also stated that it could improve the workflow when being more familiar with the approach.

4.3.2 Findings regarding the User Workflow

Evaluating the workflow and usage of our approach, we asked the participants about their feelings when working. Both participants missed certain feature as for example missing undo function in some cases or the missing functionality of removing pauses in a melody. Still, overall they stated that our approach was fun to use, supported engaging with composing melodies, and helped beginners make music. They also liked the possibility to quickly compose melodies, as well as going more into detail and taking the time for selecting samples and fine tuning them.

We were especially surprised by the different workflows of the participants and usage of our approach. One of the participants compared his workflow to drawing a picture in terms of an iterative workflow. He generated samples, selected a few bars and then went to fine tune his composition before moving

on to the next bars. Especially the usage of the fill-in, which was the most important approach for fine tuning, surprised us, as we thought the user would select a few bars and generate new samples for those. Instead, the participant used the fill-in to get suggestions for only a few notes. This could be a result of the lacking music knowledge, so the user doesn't feel confident enough to adjust single notes by themselves but uses the suggestion approach to generate and select a fitting note.

After both participants composed a small melody, they used the MIDI export function to save their first composed music and were excited for presenting it to others.

Overall, both participants used the node-link diagram and its sorting function to begin with sample selection and to get a feeling for the generated melody. As it is currently only possible to sort for a single attribute, both stated that the SPS and the glyphs were better when looking for multiple attributes of a melody. For example, when looking for a similar continuation, they looked for samples with a high similarity to the parent and a low temperature, to increase the chance of finding a sample that meets their expectations. They especially found the support of our approach useful for getting ideas and using generations as an initial spark to improve on.

To evaluate the quality of the AI prediction, we asked the participants about their findings and experiences throughout the study. They stated that some melody samples where surprisingly good, but there were more uninteresting or random samples than good samples in their opinion. This also emphasizes on our idea to generate multiple samples at the same time and our plan to include support for different models in the future.

We also asked for some suggestions for improvements and one participant, the one with more knowledge about machine learning, demanded more control over the models, for example by controlling the length of samples or suggesting notes or a rough sketch of the melody directly. The other participant thought about an additional AI that suggests beginner users, which parts of the composition hold high potential and which should be improved in order to improve the whole composition. They also liked the statistical aggregations of samples, while the users did not use it as much in this study, they most likely get more interesting the longer the system is used. One participant requested statistics for the composition, for example *mean note duration*, *variance of intervals*, and the *number of notes*.

In conclusion, both participants found our approach engaging, were able to use it quickly, used our sortings for different tasks and mentioned our approach to be supportive in composing music, especially for beginners.

5 Limitations and Discussion

In this chapter, we take a closer look at the limitations that we encountered during the evaluation and discuss their impact and possible solutions.

The basic idea of our approach is to use visualizations and sorting metrics to help choosing melody samples faster than by listening to all samples one after another. All of our visualizations show samples beside each other in order to compare them, which can be hard sometimes, as differences are still hard to see. To address the limitation, we could add visual indications, to highlight differences between samples directly, in order to improve comparing samples. As highlighting differences between samples mostly applies to a direct comparison of two samples, it might struggle when using many samples. Nevertheless, it can improve the comparison between a few selected samples by showing the differences in all other samples compared to a selected one melody sample.

As our approach should support users in composing music, it can be hard for beginners to fine tune selected samples. One of the participants requested an AI as guidance to highlight parts in the composition, where the user should take a look at for changes, as the section could have a higher potential of better quality. While this might support beginners, this might not support composers with some knowledge as much but that would need more evaluation through surveys and different knowledge groups of the composer. In addition to that, this seems like a hard task, as the quality of a section is depending on the imagination of the user and is therefore subjective.

The pair analytics study revealed interesting insights into the visualizations and usage of these in different workflows. As our study was only conducted with two beginner participants, who had limited knowledge of music theory and composing in general, the study lacks a representative sample size and domain experts, which would be an improvement when doing a larger study.

Using case studies can be a good way to show takeaways from different scenarios and visualizations, they only show a few examples. Some of the gathered information could not be as relevant for the user or do not occur in some situations at all. This is also shown by one participant, who was not able to get information out of the histogram glyphs. Therefore, it would be better to have a larger study with more participants, as more situations and also the importance for the user can be evaluated. As we could not test learnability of our system and the glyphs with our study, this would be a point to address with a larger study.

As already addressed in the scalability evaluation and in the pair analytics study, there are further limitations regarding our implementation. For example, the current version of our approach only supports monophonic melodies. Some of our functions like the similarity function or some used ML models might not work as intended with polyphonic samples. Changing the used ML models and the similarity function, designed for polyphonic music, should help adding polyphony to our approach.

As shown in Table 4.1, the run time scalability of sample generation and calculations for the visualizations struggles with generating many samples, as the values show a quadratic or exponential run time compared to the number of samples.

When talking about the scalability, we also evaluated difficulties with some of our visualizations when using many samples. While the icicle and the node-link diagram work well with multiple levels containing a few samples, both get unusable when working with, for example 50 or more samples, as they clutter and the space is too tiny to display all samples. To reduce the amount of clutter, we could display only the most interesting samples or scroll through samples to reduce to the number of samples, shown at the same time. This solution improves the visibility, but also results in some information loss due to the pre-selection of samples and missing context.

The number of samples limits the visibility of glyphs, as the grid gets smaller and the glyphs overlap. One possibility would be to decrease the size of glyphs, but we wanted to keep the visibility, as smaller glyph are harder to read. A bad readability of the piano roll glyph can occur, when cases where the contrast between background and foreground colors is low. We addressed this problem by changing the color of the foreground depending on the brightness of the background, but this can fail sometimes.

Our use case study showed interesting patterns and the functionality of the SPS and glyphs, but these were only a small number of scenarios and when evaluating more scenarios, some SPS did not show good patterns, but some exceptional side effects. Therefore, the shown patterns in the SPS might be unreliable, depending on the similarity function and the occurring samples. The chosen similarity function and dimensionality reduction technique is crucial for the expressiveness of the SPS, but hard to evaluate to find the best parameters and functions.

The user should have control over the music composition, but the user also has to choose some parameters like the weighting of the similarity parts, rhythm, and melody. This can be helpful when the user has a certain imagination of the sample he is looking for, but can leave a user, who just wants to use the total similarity, overstrained with choosing a weighting value. So the user has to choose certain parameters with maybe limited knowledge about them.

Similar to that is the scale of the axes in the icicle plot, where the user can choose between different scales. The user might be overstrained with which scale to use, but we wanted to give the user the possibility to improve visibility in certain cases.

Another limitation regarding the similarity or sorting metrics, is that we only use statistical metrics and data to calculate those. Some characteristics of the melody, like the musical temperature or the feeling a melody can trigger, would be an addition to improve the usability for more music orientated users. We decided to only use the statistical metrics for now because of the objectiveness they provide, while some musical characteristics can be subjective.

Our participants mentioned in the evaluation that melody samples, generated by the AI, can often sound unsatisfying. Although the concept of using AI to get inspired by different samples, the user expects different samples with high quality. To achieve this, the usage of different, maybe higher quality models could be a solution, but also more control of the user over the AI, as demanded by one participant in our study, can help generating more satisfying samples. Some of the ideas for example could be to have more data requirements, like note length, or a musical approach, to suggest a feeling like happiness towards the AI. It is important to notice that the quality of the AI is not directly a limitation for our approach, as our goals are the interactive visualization.

If we take a look at not just the composition process of a music track, but also playing the notes with real instruments, some health concerns rose, when for example the AI suggests large jumps in a melody, that cannot be played in that way on a real instrument without hurting the hand or wrist. As we describe our composition electronic music and the user can play the music with applications, this problem does not directly occurs in our approach, but can be a problem when being used afterwards with real instruments. Still this is not considered as one of our goal and therefore it is not taken into account.

Since the approach has some limitations and the implementation is missing features, there is space for improvement in different directions regarding the idea, implementation, user control, and AI. However the approach helps users compose music with the help of AI, while the user has control over the composition.

6 Conclusion and Future Work

We propose a user-centered approach for human-AI co-composition based on visualization of multiple generated continuation suggestions. First, we used the resulting tree structure of continuation samples to visualize multiple samples and the relationships at the same time in an icicle plot of piano rolls. To improve sample selection by the user, we added sorting by different metrics. As the icicle plot shows the relationships by positions and sorting function would destroy these, we added links between the nodes. The node-link diagram shows the relationship and the nodes visualize the melodies via piano rolls. Nodes could be sorted by different properties of the melodies like *similarity to the parent*, the hyperparameter *temperature*, and *variance of intervals* between notes. For the first metric, we implemented a similarity function, which uses a weighting between similarity of rhythm and melody. We also used this similarity in a similarity-preserving scatterplot (SPS), visualizing multiple samples and encoding their properties in different glyphs. This allowed for easier comparison between hundreds of samples at the same time.

We evaluated our design with a use case study and a pair analytics study. Although the icicle plot and the node-link diagram are limited to a few samples per level to preserve visibility, they help the user selecting out of the few samples by sorting. Users liked to generate samples, listen to the melodies, and composing music in general. They also learned how the hyperparameter temperature impacts the generated result. For working with more samples, the SPS and the combination of the different glyphs helped the user to look for interesting melodies.

6.1 Future Work

In the future, we want to incorporate more domain knowledge and music theory to improve supporting especially beginner user. This also includes to add more control over the AI, for example by adding feel suggestions. With this, the user would be able to add happiness or a different feeling as requirement for the generated samples. Choosing a feeling would imply the usage of music theory to choose a key that fits that feeling.

We further plan to test different models [HVU+18; RER+18] for the different composition tasks — continuation, fill-in, harmonization — and expand the functionality to either directly generate polyphonic music or harmonize melodies as a second step. As our current similarity function struggles with polyphonic melodies, we want to design different similarity functions that support polyphony and incorporate music theory. Additionally, we want to use models for different genres and instruments, like a rock guitar, a classic piano, or directly provide a plugin API, where the user can host his own model and use it in our system.

To emphasize on supporting the user, we could estimate and indicate the quality of each suggestion and therefore guide the user in choosing the best continuation. This could be done using model outputs, a regressor, or using the confidence of the model. Improving learnability should give the user an easier start using the system, for example by using more intuitive glyphs.

To visualize the found continuations, we want to test our visualizations with showing interesting melodies larger than others. Another possibility to only show interesting samples in the visualizations with the possibility to switch to another mode, where all samples are used. Future work also needs to empirically validate our ideas in a longitudinal study including more users.

Bibliography

[ACTS20] A. Achberger, R. Cutura, O. Türksoy, M. Sedlmair. "Caarvida: Visual Analytics for Test Drive Videos". In: *Proc. International Conf. Advanced Visual Interfaces (AVI)*. 2020 (cit. on p. 32).

[AG15] A. Agostini, D. Ghisi. "A Max Library for Musical Notation and Computer-aided Composition". In: *Computer Music Journal* (2015), pp. 11–27 (cit. on p. 19).

[AGB21] A. Arunkumar, S. Ginjpalli, C. Bryan. "Bayesian Modelling of Alluvial Diagram Complexity". In: *arXiv preprint arXiv:2108.06023* (2021) (cit. on p. 18).

[BDG+20] G. Benevento, R. De Prisco, A. Guarino, N. Lettieri, D. Malandrino, R. Zaccagnino. "Human-machine Teaming in Music: Anchored Narrative-graph Visualization and Machine Learning". In: *Proc. International Conf. Information Visualisation (IV)*. 2020, pp. 559–564 (cit. on p. 13).

[BH19] T. Bazin, G. Hadjeres. "NONOTO: A Model-agnostic Web Interface for Interactive Music Composition by Inpainting". In: *Proc. International Conf. Computational Creativity (ICCC)*. 2019, pp. 89–91 (cit. on p. 19).

[BHP20] J.-P. Briot, G. Hadjeres, F.-D. Pachet. *Deep Learning Techniques for Music Generation*. Springer, 2020 (cit. on p. 13).

[BM10] S. Bruckner, T. Möller. "Result-Driven Exploration of Simulation Parameter Spaces for Visual Effects Design". In: *IEEE Trans. Visualization and Computer Graphics (TVCG)* (2010), pp. 1468–1476 (cit. on p. 18).

[Cha18] A. Chaney. *The Watson Beat: Using Machine Learning to Inspire Musical Creativity*. 2018. URL: https://medium.com/@anna_seg/the-watson-beat-d7497406a202 (visited on 05/31/2021) (cit. on p. 17).

[CKS20] R. Cutura, C. Kralj, M. Sedlmair. "DRUID$_{JS}$ — A JavaScript Library for Dimensionality Reduction". In: *Proc. IEEE Visualization Conf. (VIS)*. 2020, pp. 111–115 (cit. on p. 41).

[CWBD20] K. Chen, C.-i. Wang, T. Berg-Kirkpatrick, S. Dubnov. "Music SketchNet: Controllable Music Generation via Factorized Representations of Pitch and Rhythm". In: *Proc. International Society for Music Information Retrieval Conf. (ISMIR)*. 2020, pp. 77–84 (cit. on p. 19).

[DF20] R. T. Dean, J. Forth. "Towards a Deep Improviser: A Prototype Deep Learning Post-tonal Free Music Generator". In: *Neural Computing and Applications* (2020), pp. 969–979 (cit. on p. 16).

[DWV13] W. B. De Haas, F. Wiering, R. C. Veltkamp. "A Geometrical Distance Measure for Determining the Similarity of Musical Harmony". In: *International Journal of Multimedia Information Retrieval (IJMIR)* (2013), pp. 189–202 (cit. on p. 32).

[FGJ20] E. Frid, C. Gomes, Z. Jin. “Music Creation by Example”. In: *Proc. Conf. Human Factors in Computing Systems (CHI)*. 2020, pp. 1–13 (cit. on pp. 13, 19).

[GB21] J. Gillick, D. Bamman. “What to Play and How to Play it: Guiding Generative Music Models with Multiple Demonstrations”. In: *Proc. International Conf. New Interfaces for Musical Expression (NIME)*. 2021 (cit. on p. 13).

[GBSW21] N. Grossmann, J. Bernard, M. Sedlmair, M. Waldner. “Does the Layout Really Matter? A Study on Visual Model Accuracy Estimation”. In: *arXiv preprint arXiv:2110.07188* (2021) (cit. on p. 18).

[GGKG20] O. Gomez, K. K. Ganguli, L. Kuzmenko, C. Guedes. “Exploring Music Collections: An Interactive, Dimensionality Reduction Approach to Visualizing Songbanks”. In: *Proc. International Conf. Intelligent User Interfaces (IUI)*. 2020, pp. 138–139 (cit. on p. 32).

[HDG16] C.-Z. A. Huang, D. Duvenaud, K. Z. Gajos. “ChordRipple: Recommending Chords to Help Novice Composers Go Beyond the Ordinary”. In: *Proc. International Conf. Intelligent User Interfaces (IUI)*. 2016, pp. 241–250 (cit. on p. 19).

[HJ19] B. Haki, S. Jorda. “A Bassline Generation System Based on Sequence-to-Sequence Learning.” In: *Proc. International Conf. New Interfaces for Musical Expression (NIME)*. 2019, pp. 204–209 (cit. on p. 17).

[HKN+20] C.-Z. A. Huang, H. V. Koops, E. Newton-Rex, M. Dinculescu, C. J. Cai. “AI Song Contest: Human-ai Co-creation in Songwriting”. In: *Proc. International Society for Music Information Retrieval Conf. (ISMIR)*. 2020, pp. 708–716 (cit. on p. 19).

[HN20] G. Hadjeres, F. Nielsen. “Anticipation-rnn: Enforcing Unary Constraints in Sequence Generation, with Application to Interactive Music Generation”. In: *Neural Computing and Applications* (2020), pp. 995–1005 (cit. on p. 16).

[HS20] F. Heyen, M. Sedlmair. “Supporting Music Education Through Visualizations of Midi Recordings”. In: *Posters IEEE Visualization Conf. (VIS)*. 2020 (cit. on p. 38).

[HVU+18] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, C. Hawthorne, A. M. Dai, M. D. Hoffman, D. Eck. “An Improved Relative Self-Attention Mechanism for Transformer with Application to Music Generation”. In: *arXiv preprint arXiv:1809.04281* (2018). arXiv: 1809.04281 (cit. on pp. 13, 17, 63).

[KC20] S. Knotts, N. Collins. “A Survey on the Uptake of Music Ai Software”. In: *Proc. International Conf. New Interfaces for Musical Expression (NIME)*. 2020, pp. 499–504 (cit. on p. 13).

[KDW18] E. S. Koh, S. Dubnov, D. Wright. “Rethinking Recurrent Latent Variable Model for Music Composition”. In: *Proc. IEEE International Workshop Multimedia Signal Processing (MMSP)*. 2018 (cit. on pp. 13, 16).

[KHW09] A. Klippel, F. Hardisty, C. Weaver. “Star Plots: How Shape Characteristics Influence Classification Tasks”. In: *Cartography and Geographic Information Science (CaGIS)* (2009), pp. 149–163 (cit. on p. 35).

[KKM+20] R. Khulusi, J. Kusnick, C. Meinecke, C. Gillmann, J. Focht, S. Jänicke. “A Survey on Visualizations for Musical Data”. In: *Computer Graphics Forum (CGF)* (2020), pp. 82–110 (cit. on p. 18).

[KL83] J. B. Kruskal, J. M. Landwehr. “Icicle Plots: Better Displays for Hierarchical Clustering”. In: *American Statistician* (1983), pp. 162–168 (cit. on p. 25).

[Kru64a] J. B. Kruskal. “Multidimensional Scaling by Optimizing Goodness of Fit to a Nonmetric Hypothesis”. In: *Psychometrika* (1964), pp. 1–27 (cit. on p. 33).

[Kru64b] J. B. Kruskal. “Nonmetric Multidimensional Scaling: A Numerical Method”. In: *Psychometrika* (1964), pp. 115–129 (cit. on p. 33).

[LCH+20] R. Louie, A. Coenen, C. Z. Huang, M. Terry, C. J. Cai. “Novice-AI Music Co-Creation via AI-Steering Tools for Deep Generative Models”. In: *Proc. Conf. Human Factors in Computing Systems (CHI)*. 2020, pp. 1–13 (cit. on pp. 13, 19).

[LKD+17] Z. Liu, B. Kerr, M. Dontcheva, J. Grover, M. Hoffman, A. Wilson. “CoreFlow: Extracting and Visualizing Branching Patterns from Event Sequences”. In: *Computer Graphics Forum (CGF)* (2017), pp. 527–538 (cit. on p. 18).

[LND21] N. Le, N. V. Nguyen, T. Dang. “Real-Time Sound Visualization via Multidimensional Clustering and Projections”. In: *International Conf. Advances in Information Technology (IAIT)*. 2021 (cit. on p. 32).

[LSM18] O. Lopez-Rincon, O. Starostenko, G. A.-S. Martín. “Algoritmic Music Composition Based on Artificial Intelligence: A Survey”. In: *Proc. International Conf. Electronics, Communications and Computers (CONIELECOMP)*. 2018, pp. 187–193 (cit. on p. 13).

[Lup21] J. A. Lupker. “Score-Transformer: A Deep Learning Aid for Music Composition”. In: *International Conf. New Interfaces for Musical Expression (NIME)*. 2021 (cit. on p. 17).

[MKG+17] S. Mehri, K. Kumar, I. Gulrajani, R. Kumar, S. Jain, J. Sotelo, A. Courville, Y. Bengio. “Samplernn: An Unconditional End-to-end Neural Audio Generation Model”. In: *International Conf. on Learning Representations (ICLR)*. 2017 (cit. on pp. 13, 16).

[Moo86] R. A. Moog. “MIDI: Musical Instrument Digital Interface”. In: *Journal of the Audio Engineering Society* (1986), pp. 394–404 (cit. on p. 41).

[NHJ21] T. Nuttall, B. Haki, S. Jorda. “Transformer Neural Networks for Automated Rhythm Generation”. In: *Proc. International Conf. New Interfaces for Musical Expression (NIME)*. 2021 (cit. on p. 17).

[Par18] T. Parviainen. “Musical Deep Neural Networks in the Browser”. In: *Proc. International Web Audio Conf. (WAC)*. 2018 (cit. on p. 18).

[Pas18] L. Passing. “Generative Music, Playful Visualizations and Where to Find Them”. In: *Proc. International Web Audio Conf. (WAC)*. 2018 (cit. on p. 18).

[PKL+19] S. Park, T. Kwon, J. Lee, J. Kim, J. Nam. “A Cross-scape Plot Representation for Visualizing Symbolic Melodic Similarity”. In: *Proc. International Society for Music Information Retrieval Conf. (ISMIR)*. 2019, pp. 423–430 (cit. on p. 32).

[RER+18] A. Roberts, J. Engel, C. Raffel, C. Hawthorne, D. Eck. “A Hierarchical Latent Vector Model for Learning Long-term Structure in Music”. In: *Proc. International Conf. Machine Learning (PMLR)*. 2018, pp. 4364–4373 (cit. on pp. 13, 16, 63).

[RHS18] A. Roberts, C. Hawthorne, I. Simon. “Magenta.js: A JavaScript API for Augmenting Creativity with Deep Learning”. In: *Joint Workshop on Machine Learning for Music (ICML)* (2018), pp. 2–4 (cit. on p. 40).

[Roa85] C. Roads. “Research in Music and Artificial Intelligence”. In: *ACM Computing Surveys (CSUR)* (1985), pp. 163–190 (cit. on p. 15).

[SMT13] M. Sedlmair, T. Munzner, M. Tory. “Empirical Guidance on Scatterplot and Dimension Reduction Technique Choices”. In: *IEEE Trans. Visualization and Computer Graphics (TVCG)* (2013), pp. 2634–2643 (cit. on p. 32).

[SO17] I. Simon, S. Oore. “Performance RNN: Generating Music with Expressive Timing and Dynamics”. In: *Magenta Blog* (2017) (cit. on p. 16).

[SSS+19] J. Smith, M. Street, M. Street, B. Magerko, C. Street. “Combining Collaborative and Content Filtering in a Recommendation System for a Web-based DAW”. In: *Proc. International Web Audio Conf. (WAC)* (2019) (cit. on p. 32).

[Sto16] M. Stone. *How We Designed the New Color Palettes in Tableau 10*. 2016. URL: https://www.tableau.com/about/blog/2016/7/colors-upgrade-tableau-10-56782 (visited on 08/31/2021) (cit. on p. 25).

[SW97] S. Smith, G. Williams. “A visualization of Music”. In: *Proc. Visualization ’97 (Cat. No. 97CB36155)*. 1997, pp. 499–503 (cit. on p. 18).

[SYTC21] M. Suh, E. Youngblom, M. Terry, C. J. Cai. “AI as Social Glue: Uncovering the Roles of Deep Generative AI during Social Music Composition”. In: *Proc. Conf. Human Factors in Computing Systems (CHI)*. 2021 (cit. on p. 13).

[TWM19] M. Taenzer, B. C. Wünsche, S. Müller. “Analysis and Visualisation of Music”. In: *Proc. International Conf. Electronics, Information, and Communication (ICEIC)*. 2019 (cit. on p. 18).

[ULMS10] J. Urbano, J. Lloréns, J. Morato, S. Sánchez-Cuadrado. “Melodic Similarity Through Shape Similarity”. In: *International Symp. Computer Music Multidisciplinary Research (CMMR)* (2010), pp. 338–355 (cit. on p. 32).

[Wat01] M. Wattenberg. *The Shape of Song*. 2001. URL: http://www.turbulence.org/Works/song/mono.html (cit. on p. 18).

[WATS19] A. Weber, L. N. Alegre, J. Torresen, B. C. da Silva. “Parameterized Melody Generation with Autoencoders and Temporally-Consistent Noise”. In: *Proc. International Conf. New Interfaces for Musical Expression (NIME)*. 2019, pp. 174–179 (cit. on p. 16).

[WCHU08] M. Ward, C.-h. Chen, W. K. Härdle, A. Unwin. “Multivariate Data Glyphs: Principles and Practice”. In: *Handbook of Data Visualization.* 2008, pp. 179–198 (cit. on p. 33).

[WGP+11] K. Wongsuphasawat, J. A. Guerra Gómez, C. Plaisant, T. D. Wang, M. Taieb-Maimon, B. Shneiderman. “LifeFlow: Visualizing an Overview of Event Sequences”. In: *Proc. Conf. Human Factors in Computing Systems (CHI)*. 2011, pp. 1747–1756 (cit. on p. 18).

[ZXLD21] Y. Zhang, G. Xia, M. Levy, S. Dixon. “COSMIC: A Conversational Interface for Human-AI Music Co-Creation”. In: *Proc. International Conf. New Interfaces for Musical Expression (NIME)*. 2021 (cit. on p. 19).

A Magenta Model Comparison

We tested some models for music generation provided by Magenta and show the information and our findings in the following tables. In Appendix Table A.1, we sum up the important characteristics of different Magenta models. Therefore, we take a look at the architecture, inputs, outputs, musical texture, and the availability as pre-trained model. As we only used a few of these presented models, we named some reasons on why we decided to either use or do not use the model (Appendix Table A.2).

To get a feeling for the subjective quality of generated samples by the chosen models, we tested different seed melodies and temperature using the models and documented our subjective results for each sample and then decided whether a neural network is interesting depending only on the one sample. Therefore, they are subjective opinions and not a general findings. We grouped our subjective findings were by the used models: basicRNN (Appendix Table A.3), melodyRNN (Appendix Table A.4), improvRNN (Appendix Table A.5), and the older version of the improvRNN (Appendix Table A.6).

The quality of melody samples and therefore the support through suggestions when composing depends on model choice for different tasks. The used RNN models for example use the seed melody and temperature to generate continuations, which is good for using at the end of compositions. For the fill-in task, the RNN models do not fit as well as for example VAE models, which are specialized on interpolating between two melodies and therefore fit the fill-in task better. We decided to use the same, simple model for an easier understanding, as more models require more knowledge of the user. Therefore, the usage of different ML models for different task would further increase the quality of the output samples.

NN Name	Architecture	Hyperparameters	Task	Musical Texture	Used	Pre-trained
Basic RNN	RNN	Temperature, (chord progression), example melody	Predict melody continuation in limited keyrange	M	√	√
Melody RNN	RNN	Temperature, (chord progression), example melody	Predict melody continuation	M	√	√
Improv RNN	RNN	Temperature, chord progression, example melody	Predict melody continuation	M	√	√
Performance RNN	RNN	Example melody, key conditions	Generate performance (dynamics, natural timing)	P	×	×
Coconet	CNN	Input melody	Harmonize input melody	P	×	×
GANSynth	GAN	(Latentspace point)	Predict input, random sampling (direct audio)	P	×	×
mel Music VAE	VAE	Temperature	Generate new melody, interpolate between melodies	M	×	√
trio Music VAE	VAE	Temperature	Sampling melody, bass, and drums	P	×	√
multitrack Music VAE	VAE	Temperature	Sampling multitracks (chord conditioned)	P	×	√
PianoGenie	En-/decoder RNN	Latentspace butten numbers, (key list, temperature, seed)	Generate next keys note in real time	M/P	×	×

Table A.1: Comparison of available Magenta ML models. Musical texture: monophonic (M) or polyphonic (P).

NN Name	Used	Reason for decision
Basic RNN	√	BasicRNN is monophonic and continues a given start melody. The output melody is limited to a smaller pitch range which improves visualizations and saves space. The continuations are well suited for the iterative composition style.
Melody RNN	√	MelodyRNN is monophonic and continues a given start melody. The output melody gives a wider range of notes due to no limitations which result in more different possibilities and a wider range of melodies. The continuations are well suited for the iterative composition style.
Improv RNN	√	ImprovRNN is monophonic and continues a given start melody. The output melody is also constrained by a chord progression. Therefore the output melodies are more bound to a certain tone which leads to longer structure. The continuations are well suited for the iterative composition style.
Performance RNN	×	PerformanceRNN is polyphonic and therefore not quite suited for only monophonic melody composition. Also the PerformanceRNN simulates different dynamics and natural timing which are characteristics of performing music and not quite composing a simple melody.
Coconet	×	Coconet harmonizes a given input melody and is therefore polyphonic. The task coconet fulfills is not the same task in this thesis and therefore unsuited.
GANSynth	×	GANSynth predicts a polyphonic sample which is represented in an audio form already. Therefore it is not as usable for composing a simple monophonic melody with the intention to adjust single notes and give the user control over the notes.
mel Music VAE	×	Melody MusicVAE can generate completely new monophonic samples and is therefore suited for giving seed melodys if the user lacks in ideas. But for iterative composition the model is not usable due to the lack of structure between generated samples. The interpolation between two melodies can be used for a fill-in but not for continuation and the iterative composition process
trio Music VAE	×	Trio MusicVAE generates completely new polyphonic samples or can interpolate between two samples but is not suited for a monophonic melody composition. Same aspects as in melody MusicVAE apply.
multitrack Music VAE	×	Trio MusicVAE generates completely new polyphonic samples or can interpolate between two samples but is not suited for a monophonic melody composition. Same aspects as in melody MusicVAE apply.
PianoGenie	×	PianoGenie can respond in real time by outputting a single or multiple notes depending on a single or multiple button presses as input. Therefore this model is made for real time improvising with only a few input buttons. This is not used in this thesis due to the possibility to only generate one melody at a time and the potential on missing out on interesting alternatives.

Table A.2: Comparison of all models and the reasons to use or not use in this thesis.

NN Name	Seed Melody (4 Bars)	Low Temp. (0.4-0.8)	Medium Temp. (1)	High Temp. (1.2-1.6)	Interesting?
Basic RNN	Crazy Train Intro	Not as exciting. Many repetition of the same notes. Almost no sharp notes even if input has many sharps.	More occurrences of notes from seed melody. Sign of melody structure recognizable but with randomness.	Chosen notes seem random. Note duration varies even if seed only has same size. Melody structure has no direct relationship to seed melody	×
Basic RNN	No Notes	Few long notes with similar pitch and long pause at the start. Not as interesting	Pause at the start and then medium long notes. Melody shows a scale structure. Calm melody.	Bigger jumps and more variation in note duration and pitch. Show different types of melody structure.	√
Basic RNN	Twinkle twinkle	Low temperature shows same note and duration. Increasing temperature more variation is shown and the melody structure is slightly recognizable. Pitches are used all used in seed melody as well but at different timings.	Seems more random with newly used pitches and different structure. Note duration can vary, but does not look as interesting.	Different melodies compared to seed melody. Shows different melody structure and different note duration. Can seem interesting and exciting.	×

Table A.3: BasicRNN melody generation evaluation (Temp = temperature).

NN Name	Seed Melody (4 Bars)	Low Temp. (0.4-0.8)	Medium Temp. (1)	High Temp. (1.2-1.6)	Interesting?
Melody RNN	Crazy Train Intro	Same melody as seed with small change.	Shows a highly similar melody to the low temperature ones and the seed ones. Small step changes to a few notes.	Similar structure to the seed with more small adaptations but also randomness with increasing temperature, note duration only changes to even shorter notes and more pitch variations. Structure of alternating big jumps is sometimes preserved	√
Melody RNN	No Notes	Mostly same notes and same duration of notes. Not as interesting.	Only shows small intervals between notes. Four different pitches alternating. Seems not as interesting.	Shows more variety regarding pitch, note duration and intervals between notes. More exiting melody.	×
Melody RNN	Twinkle twinkle	Shows exactly the seed melody with no variation.	Shows signs of similar melody structure but varies in limited pitches. Not as interesting.	Show similar melody structure but with high randomness at the end with short notes and big intervals in pitch. Can be interesting but also far from seed melody.	√

Table A.4: MelodyRNN melody generation evaluation (Temp = temperature).

NN Name	Seed Melody (4 Bars)	Low Temp (0.4-0.8)	Medium Temp (1)	High Temp (1.2-1.6)	Interesting?
Improv RNN	Crazy Train Intro	Note duration is sometimes preserved. The used keys are limited due to the chord progression of the seed melody. Not as interesting as continuation.	Melody structure is not preserved. Note duration is same as in seed. More interesting than low temperature due to bigger intervals but seem not as interesting	Shows randomness in terms of different note lengths and sometimes big intervals. Notes are all bound to the chord progression so usage of restricted randomness.	×
Improv RNN	No Notes	Mostly same notes and same duration of notes. Not as interesting. But can show different keys.	Start more jumps and then alternating between two pitches. Only uses four different pitches.	Uses the same pitches due to chord progression but can show different exciting melodies and variety in note duration. Shows parts where the same pitch occurs in quick succession.	√
Improv RNN	Twinkle twinkle	Shows same pitch and duration the whole time.	Same rhythm as seed melody but simple and less interesting scale melody structure.	Shows similarities to seed melody but also varies more in note length and random pitches. Seem to be more interesting and can be variations and continuations of seed melody.	√

Table A.5: ImprovRNN melody generation evaluation (Temp = temperature).

NN Name	Seed Melody (4 Bars)	Low Temp (0.4-0.8)	Medium Temp (1)	High Temp (1.2-1.6)	Interesting?
Old Improv RNN	Crazy Train Intro	Mainly uses the three pitches of the main chord. Alternating notes and sometimes variation of note duration. Not as interesting.	Varies in note duration in comparison to seed melody. Melody structure is not recognizable and has no large variety of pitches.	Sometimes big intervals between notes. Shows a scale structure instead of a alternating structure. Shows melody over 3 scales instead of one in the seed melody.	×
Old Improv RNN	No Notes	Mostly same pitches and duration of notes. Not as interesting.	Shows scale structure with variety of length and pitch. Not as interesting.	No big intervals between notes resulting in melody with a low pitch range. Not as exciting and interesting.	×
Old Improv RNN	Twinkle twinkle	Shows same note duration the whole time. Limited amount of different pitches are used. Not as interesting.	Shows melody structure but with different pitches and variation. Can be interesting as variation to seed melody.	Shows limited melody structure and overall not as interesting due to the simple structure and the limited variations.	×

Table A.6: Old version of the ImprovRNN melody generation evaluation (Temp = temperature).

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

place, date, signature