Diffusion Models for Symbolic Music Generation

Andreas Kalavas
School of Electrical and Computer Engineering
National Technical University of Athens
Athens, Greece
andreaskalavas@gmail.com

***Abstract*—In recent years, Diffusion Models have been integrated into the field of Machine Learning, demonstrating significant advancements and attaining state-of-the-art performance as generative models, with a notable emphasis on image generation. While these models have garnered considerable attention and success in various domains, one relatively under-explored area pertains to Symbolic Music Generation. This paper presents a new dataset comprising binary representations of piano rolls, accompanied by its use on a model developed in prior research in the field.**

I. Introduction

Computation was barely a theoretical vision when researchers started to muse about the possibility of composing music automatically and algorithmically [4]. Furthermore, this initial fascination with the field has persisted over time. As evidenced by the evolution from variational autoencoders to discrete diffusion models, researchers continue their pursuit to push the boundaries of the available resources and machine learning knowledge. This relentless quest is driven by the aspiration to create more advanced and improved music generation models.

An indication of the relative underdevelopment in this field is the limited number of available datasets, often characterized by suboptimal quality or insufficient size. The evolution of written music spans at least four centuries, traversing various phases. The majority of existing datasets lack specificity with regard to distinct historical periods and frequently encompass samples of questionable musical merit. Moreover, the inherently intricate nature of music poses challenges in its faithful representation. Presently, the predominant format for music datasets is comprised of MIDI files, yet we experiment with a different representation of musical content, and discuss the advantages of each one.

This paper initiates by presenting past endeavors within the domain of symbolic music generation in section II. We elucidate three distinct models, each approaching the task from different perspectives. Subsequently, in section III we introduce a new dataset, or better a family of datasets, consisting binary representations of piano rolls derived from keyboard compositions by Bach. Following this, in section IV we formulate a model grounded extensively in the aforementioned antecedent work, along with a thorough examination of it. Finally, in section V we present our results and in section VI we provide a comprehensive discussion about various facets of the process, including the dataset's intrinsic attributes, the model's comprehension and conception of musical essence, and the quality evaluation of the model-generated samples.

II. Previous Work

In this section, we delineate three models introduced in prior scholarly contributions. These models exhibit several distinctive features, leading to distinct strengths and weaknesses for each of them.

A. Mittal et al. [1]

Within this study, a novel model is proposed, which employs a pre-trained variational autoencoder, specifically the MusicVAE [2], to encode $2-bar$ MIDI monophonic segments into latent representations. Subsequently, a diffusion model is crafted to effectively capture interdependencies among 32 latent variables, thus facilitating the modeling of a comprehensive $64-bar$ context. Notably, a selection process is applied, retaining only the 42 dimensions of the autoencoder with standard deviations less than 1.0 prior to the training of the diffusion model. This step is implemented to eliminate the modeling of high-dimensional noise. The initial phase involves a fully connected layer that projects the $42-dimensional$ latents into a $128-dimensional$ space. Subsequently, the diffusion model utilizes a linear noise schedule spanning 1000 $steps$, with $\beta_1 = 10^{-6}$ and $\beta_N = 0.01$. The model is trained on the Lakh MIDI dataset [3].

This model is employed for the tasks of Unconditional Generation and Infilling. In the former, the objective is to generate samples that exhibit prolonged structural coherence. To achieve this, the model initializes Gaussian noise and employs the reverse process to progressively refine these noise samples into a sequence of embeddings conforming to the data distribution. In the context of infilling, during each sampling step, the model diffuses the fixed regions of the sample s using the forward process $q(s_t|s)$ and employs a mask m to merge the diffused fixed regions with the updated sample x_{t-1}. The ultimate output x0 represents a modified version of s, in which the occluded areas are restored using the reverse process. In both of these tasks, the produced embeddings are input into the MusicVAE decoder, which subsequently generates the final MIDI sequence.

To assess the model's performance, a framewise self-similarity metric is devised to gauge the statistical congruence between the generated output and the original sequences. This metric is systematically evaluated in comparison to a collection

of baseline generators. The procedure involves initially modeling the pitch and rhythm within $4-bar$ segments as normal distributions, each characterized by a respective mean and variance. The assessment involves quantifying the overlapping regions of these distributions, and subsequently, aggregating these values across all segments. For a more comprehensive elucidation of this process, we encourage interested readers to refer the corresponding paper [1].

B. Plasser et al. [4]

This research employs Discrete Diffusion Models, a class of models in which the forward transition probabilities for discrete scalar random variables, are characterized through matrices as introduced by Austin et al. [5]. Among the various matrix types available for this purpose, the Absorbing State matrix type is chosen due to its several advantageous features. In the forward diffusion process, random variables are randomly masked out, and they can solely transition into the absorbing state, but are precluded from transitioning out of it, thus leading to an absorption phenomenon. Conversely, in the reverse diffusion process, variables can only transition if they currently reside within the absorbing state.

The architectural framework of the model called SCHmUBERT can be summarized as follows: Initially, the discrete indices, representing musical notes, are transformed into $128-dimensional$ vectors. These vectors are subsequently consolidated into a single $512-dimensional$ embedding via a one-dimensional Convolutional Neural Network ($1-D$ CNN). Subsequently, a sequence of 24 transformers, each representing a step in the reverse diffusion process, is applied to the compressed embeddings. Finally, a shared one-dimensional CNN is employed to decompress the sequence, restoring it to its original length. Finally, a shared head is utilized to map the sequence of dimensions 1024×128 back to the original sequence of discrete indices.

This model undergoes training using the identical dataset as the previous one, inclusive of trios. It is subsequently assessed in comparison to the prior model, employing the same metric, across the tasks of Unconditional Generation and Infilling. In all these evaluation scenarios, the new model demonstrates superior performance and outperforms its predecessor. Finally, the authors address the limitations inherent to statistical metrics and acknowledge the inherent challenge in objectively assessing the quality of music.

C. Atassi [6]

In this paper, a novel diffusion model is introduced, employing a binomial prior distribution for the generation of piano rolls, along with an efficient training methodology for sample generation. A key distinguishing feature of this model, in contrast to the two already discussed models, is its direct training with binary piano rolls, each having dimensions of 384×56, similar to image data. In the context of the reverse diffusion process, a UNet architecture [7] is utilized for training and the number of $steps$ is set to 100. The model is trained using the Maestro dataset.

The model is versatile in its ability to perform various tasks, encompassing Unconditional Generation, Infilling, and additional functions such as Melody Harmonization, primarily facilitated by its input format. To elaborate, as the input comprises an image representing a piano roll, it enables the selective masking of the lower half of the image. Consequently, this transformation effectively transposes the Melody Harmonization task into an Infilling task along the $y-axis$.

Additional details regarding this model are available in section IV, as it consists the foundational basis for our own model.

D. Training Times

The model developed by Mittal et al. required a training time of 6.5 hours. In contrast, Plasser et al.'s model took 24 hours for the single melody configuration and an extended duration of 50 hours for the trio setting. On the other hand, Atassi's model demanded 48 hours of training.

III. Dataset

Probably the most significant contribution of this paper is the development of an image dataset containing keyboard compositions by Bach, represented as binary piano rolls. The selection of these compositions was deliberate, as Baroque Music is distinguished by a high degree of structural organization, rendering its motifs readily discernible within a musical excerpt. Below we explain the course of developing the dataset.

A. MIDI Files

Initially we gathered a portion of the keyboard compositions by Bach in the form of MIDI files. Subsequently, we systematically processed each piece to make them suitable for the impending conversion, a process explained in the next subsection. Specifically, we refined the compositions and tried to ensure that the smallest note value adhered to the demisemiquaver or the thirty-second note, with any additional note values being multiples thereof. This involved the elimination of trills and triplets (and tuplets in general), subsequently rearticulating them in the most optimal manner using the available values. Certainly, in instances where such an alteration would significantly disrupt the integrity of the composition, as exemplified by Prelude No. 6 in D minor from the first book of The Well-Tempered Clavier, we chose to preserve the original structure. In addition, all pieces have been transposed to the key of C major or C minor for reasons addressed later. Noteworthy works included are the inventions and the sinfonias. This dataset is made available in the GitHub repository[1] as '$midi\_dataset.zip$'.

B. From MIDI to Binary Image

The conversion of MIDI files into binary images necessitates a comprehensive consideration of various factors. Primarily, it is imperative to delineate the elements of the musical

[1]Our implementation and dataset can be found at https://github.com/ntua-el19709/AI_music_generation.git

composition that can be retained in the ensuing representation and those that must be excluded. Given the binary nature of the images, regretfully, all dynamic nuances, tempo indications, and time signatures must be omitted. Additionally, the transposition of all pieces to C major or C minor results in the forfeiture of information pertaining to the original key signature.

We now look into the technical intricacies of the conversion. Contemplating the nature of the images, we recognize that one axis – the pitch axis – will comprise 88 pixels, corresponding to the number of available pitches on a standard keyboard. Simultaneously, the other axis – the time axis – can be configured to incorporate any desired number of pixels. In addition, the temporal extent represented by a single pixel must be determined. Conceptually, a pixel can be regarded as a discrete time tick, and the determination of its duration involves assessing the number of ticks within a demisemiquaver, which constitutes the smallest musical note value in the most of the MIDI files. Through manipulation of these parameters, it becomes feasible to generate distinct image datasets derived from the same MIDI dataset.

Introducing data augmentation techniques serves to enhance the dataset, and we outline three methodologies for achieving this. Firstly, the consideration of overlapping entails determining the extent to which the ending of one image aligns with the beginning of the next one. This approach allows for the appearance of small music segments in various regions of the image, fostering a sense of connectivity between phrases. Additionally, transposition of each MIDI file by distinct intervals facilitates the spread of melodies across a broader range of pitches. Lastly, a reduction in the tempo of musical pieces can be effectuated by doubling each column of the images. This manipulation conditions the model to comprehend the relative nature of tempo, as the duplication of both music note values and tempo yields an equivalent outcome.

In light of the foregoing considerations, we are now able to introduce the generating function of the image dataset.

C. A Family of Datasets

In the GitHub repository two notebooks can also be found. These notebooks implement the required functions to convert the MIDI dataset to a binary image dataset, as well as convert the generated image samples back to MIDI files.

The function responsible for generating the dataset is implemented within the '$midi2binim\_dataset.ipynb$' notebook. Upon calling the '$make\_dataset()$' function, numerous parameters can be specified to customize the output dataset according to individual preferences. This approach enables the generation of image datasets that are similar in nature but exhibit distinctions, originating from the same MIDI dataset. Furthermore, for the purpose of dataset size reduction, without substantially increasing the number of included compositions, it is conceivable to selectively choose a subset of the pieces available in the midi dataset and apply augmentation techniques to them. Additional details are available within the notebook for comprehensive insights.

Similarly, within the notebook '$binim2midi.ipynb$', the function that converts the model-generated sample images back to MIDI files is implemented. Again, additional details are available within the notebook.

IV. Model

As previously mentioned, the foundational framework of our model is derived from the work of Atassi [6]. Essentially, we modify and train that model using our newly developed dataset. In the following subsections, we will swiftly describe the forward process, the sampling algorithms, and the model architecture. Given that this essentially constitutes a replication of the original model, we advise interested readers to refer to the original paper [6] for a more comprehensive analysis. Furthermore, we discuss how we selected the dataset (among various potential others) that will be employed for training.

A. Forward Process

Given that the model employs binomial prior distribution the kernel will be $q(x_t|x_{t-1}) = B(x_t; x_{t-1}(1-\beta_t) + 0.5\beta_t)$, where β_t denotes the diffusion rate, and x_0 represents the input. Additionally, according to [8], it is feasible to establish a forward process that solely depends on the input x_0. This results in efficiently computing the output at step t, without having to compute the preceding outputs [6]. Consequently, we can shuffle the order of noisy training data within minibatches, enabling the model to generate them dynamically, thus requiring less memory. The new kernel will be $q(x_t|x_0) = B(x_t; \bar{a}_t x_0 + (1-\bar{a}_t)0.5)$, where $\bar{a}_t = \prod_{s=1}^{t} a_s$, and $a_t = 1 - b_t$. We also note that the 0.5 in the equation is the success probability of the prior distribution, thus we substitute it with the ratio of ones in the piano rolls.

B. Sampling Algorithms

Inspired by the method presented in [9], the sampling algorithm diverges from the conventional sampling approach. In the typical sampling procedure, where the extraction and addition of noise can result in a point x_{t-1} being considerably distant from x_t, the samples can be unstable. A refined approach involves partially adding the initial noise x_T back into the estimated x_0. The proposed algorithm can be seen below:

Algorithm 1 Generating new samples

Input: A piano roll sampled from a binomial distribution x_T
for $t = T, T-1, \ldots, 1$ **do**
  $\hat{x}_0 = \text{UNet}(x_t)$
  $\delta = x_T \bigoplus \hat{x}_0$
  mask $\sim B(\delta\beta_t)$
  $x_{t-1} = \hat{x}_0 \bigodot (1 - \text{mask}) + x_t \bigodot \text{mask}$
end for
return x_0

This algorithm pertains to unconditional generation and can be easily modified to accommodate the infilling task by making slight adjustments. Specifically, we fix the areas (the

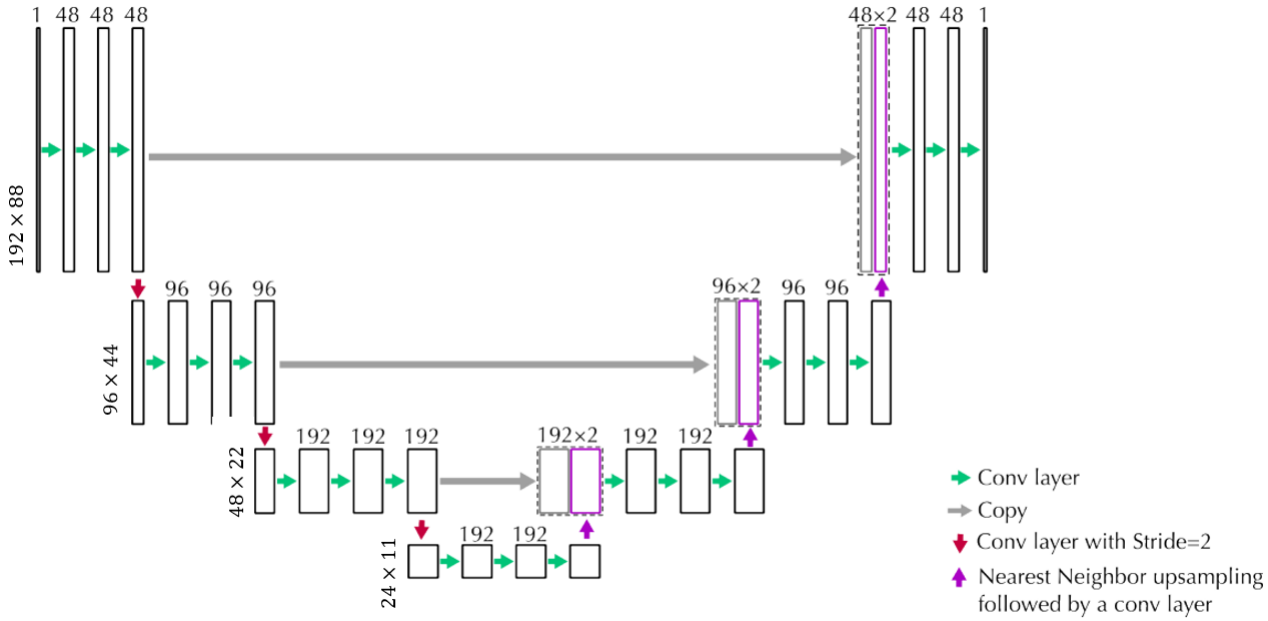

Fig. 1: Taken from [6] this figure depicts the UNet architecture used in the experiments, adjusted for input matrices (representing piano rolls) of size 88×192. Each green arrow shows a conv layer. The rectangles depict the feature maps, with the number above each rectangle showing the depth of the feature-map tensor which is also the number of conv filters in the corresponding layer. The red arrows shows the conv layers with stride of two that reduce the size (rows and columns) of the feature maps by 2. The purple arrows show the upsampling step using the nearest neighbor interpolation, which doubles the size of the feature maps. The grey arrows depict copying a feature map from the left side of the network to the right side. On the right side, the copied feature maps are stacked on the upsampled feature maps, resulting in a feature map with double the depth of the feature map tensor below. The last conv layer, top right, contains a single convolution filter, as a result, the output is a single piano roll. The number of rows and columns of the tensor in each row or level of UNet remain the same due to padding the input when applying convolution filters.

notes) of the image that are given and need to stay intact, before each step and after the last one. This way we reestablish those fixed regions for subsequent iterations. So the infilling algorithm is:

Algorithm 2 Infilling samples

Input: A fixed region r, a piano roll sampled from a binomial distribution x_T
fix r in x_T
for $t = T, T-1, \ldots, 1$ **do**
  $\hat{x}_0 = \text{UNet}(x_t)$
  $\delta = x_T \bigoplus \hat{x}_0$
  mask $\sim B(\delta\beta_t)$
  $x_{t-1} = \hat{x}_0 \bigodot (1-\text{mask}) + x_t \bigodot \text{mask}$
  fix r in x_{t-1}
end for
return x_0

C. Model Architecture

Within the steps of the diffusion model, a UNet model [7] is incorporated. The UNet model used in our work is illustrated in Figure 1. The difference with the typical hourglass model is the existence of horizontal connections that connect the feature maps at the same scale from the first half of the network to the second half. The UNet further employs two primary operations: downsampling, achieved through a convolutional layer with a stride of 2, and upsampling, accomplished by duplicating values within the input feature map. The final output has the same size as the input, thus the UNet preserves the dimensions of the piano roll segments.

D. Training Dataset

In the original study, the underwent training with images of dimensions 384×56, utilizing a dataset with 2044 piano roll segments. This training regimen spanned a duration of two days. Consequently, we must be careful not to significantly exceed the scale of the original dataset to ensure effective model training. Thus, we opted to constrain the training dataset exclusively to the 24 fugues of the first Well-Tempered Clavier book by Bach. The span is set to 192 ticks and augmentation techniques involve transposition by an octave higher, overlap, and one halving of the speed. This yields a dataset of 2262 piano roll segments.

Since the dimensions of our images are 192x88, our dataset's size aligns closely with that employed in the original paper. Furthermore, we highlight that the actual size of the dataset is 100 times larger, as for every piano roll segment,

100 instances are introduced by the diffusion model, each consisting of the original plus a varying amount of noise ranging from none to pure noise.

Several considerations guided us to the selection of this specific dataset and the accompanying augmentation techniques. Primarily, fugues are characterized by repeatedly occurring short and simple musical motifs, leading us to believe that the model is likely to distinguish them more effectively compared to longer and more intricate melodies. Moreover, the decision to solely transpose them by an octave is grounded in the belief that this approach facilitates the model's comprehension of the hierarchical relationships within a musical scale, encompassing elements such as the tonic and the dominant. Furthermore, the choice to overlap and halve the speed once serves the dual purpose of generating additional segments without introducing an excessive volume of new information into the model. This strategic augmentation is predicated on the assumption that such an approach will enhance the model's capacity to encapsulate the nature and inherent characteristics of fugues.

V. Results

This section presents the outcomes attained by the model in the tasks of unconditional generation and infilling. More specifically, the model undergoes testing in three distinct infilling scenarios: firstly, infilling the piano roll in the time domain, secondly, infilling the piano roll from the top voice, and lastly, infilling the piano roll from the middle voice. Prior to delving into these evaluations, however, we briefly discuss about the training process.

A. Training Process

As mentioned above, the dataset included 2262 piano roll segments (therefore 226200 images including those with noise). Due to memory limitations, we set the batch size to 20, but perform a backward pass every 10 batches, so that we imitate a batch size of 200. Moreover, we set the number of epochs to 50. We trained the model on two GPUs, and the training time was around 130 hours.

In the Github repository, there is the notebook called '$train\_model.ipynb$' contains the code, which was heavily based on the original of [6], as well as our trained model '$checkpoint.pth$'.

B. Unconditional Generation

Some samples of Unconditional Generation can be seen in Figure 2. From top to bottom we can witness the ability of the model in generating from single voice to multi voice piano rolls. The first piano roll is like the beginning of a fugue, where the theme is exposed and then other voices are entering. In the second piano roll we can see that the model recognized themes, and repeated them in a higher pitch (the two halves of the piano roll are identical but the second one is a perfect 4th higher). In the next two samples we can observe the movement of the voices, as well as the Baroque-like harmonies that they create. We also notice that in all the samples there is still some noise (that sound like random notes in the midi file). This also happens with the generated regions of the samples of the infilling task.

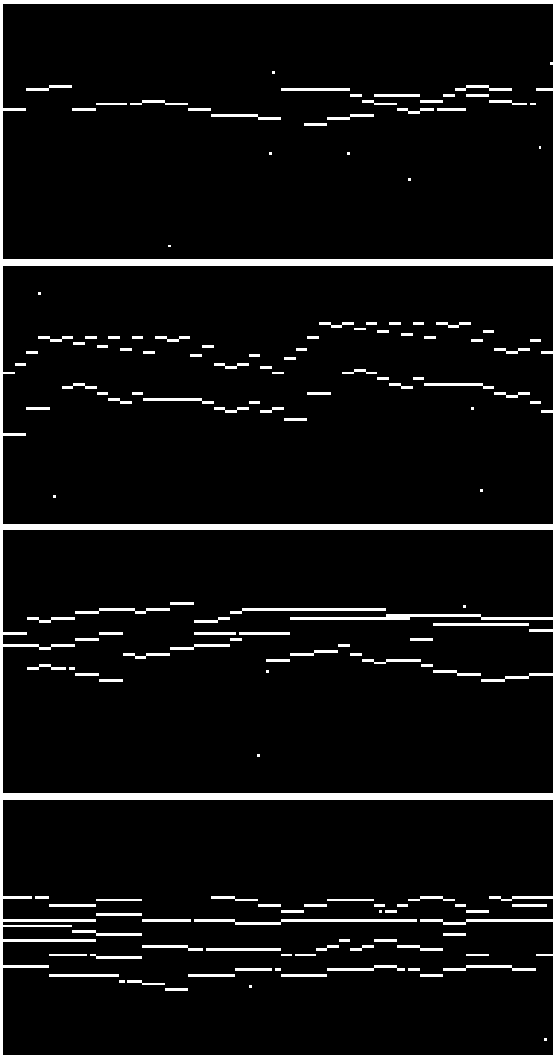

Fig. 2: Samples of Unconditional Generation

These midi samples, along with mp3 files, can be found in the Github repository for the readers to listen. Also, in the github repository there is the notebook '$generate.ipynb$', where our code for unconditional generation as well as infilling is (again based on the code of [6]).

C. Infilling the Time Domain

In this particular task, we fixed the first and the last quarters of the segment, thereby directing our model to generate the intermediate half. The results are depicted in the second row of Figure 3. Notably, the model demonstrates proficient infilling within the central portion, as evidenced by its adherence to the musical key (stays in C major) and its commendable emulation of the textural attributes present in the fixed segments. Interestingly, in the third sample the generated part modulates

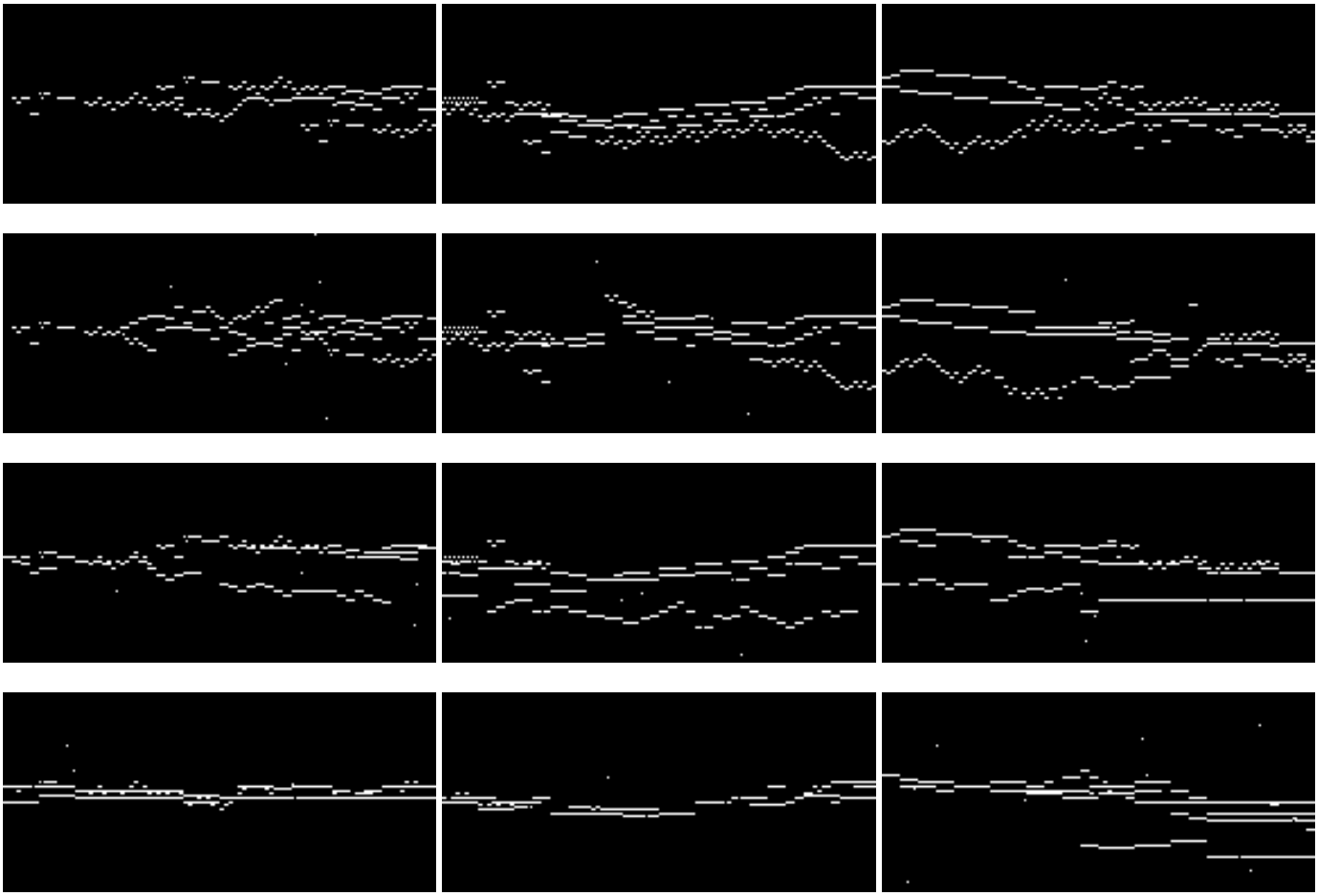

Fig. 3: Samples of Infilling. From top to bottom, on the first row are the original samples, on the second the time infilled, on the third the infilled from the top voice, and on the fourth the infilled from the middle voice.

in D minor (with a $ic - V - i$ chord progression), just before returning back to C major for the fixed ending.

D. Infilling from the Top Voice

For this task we fixed the top voice of the fugue, as well as the region above that (pitch-wise), to make sure that the model wont generate anything higher than the voice. This task could be interpreted as harmonization as well (in different context), as it generated the base lines for a given melody. The generated samples with this method are depicted in the third row of Figure 3. The most interesting out of the three samples is the second one, where the model, given a slowly ascending melody line, generates a coherent harmony along with a contrasting fast paced base line.

E. Infilling from the Middle Voice

Given that we are working with fugues, we thought it would be interesting to assess the capabilities of our model in the construction of a fugue using a single voice, other than the top one. To do that, we only fixed the middle melody line of the fugue, and left the rest of the image up to the model to generate. Unfortunately, as we can witness from the samples in the fourth line of Figure 3, the model almost ignores the given line, and composes melodies on top of it. We think that this happens, because our model might perceive the given voice as noise, and thereby trying to remove it. Having said that, it is interesting to examine the third sample, wherein the middle voice comprises prolonged notes progressing in stepwise motion. In this instance, our model generated a harmonious arrangement, concluding in a perfect cadence ($IV - V - I$), that notably reminds an ending of a fugue.

VI. Discussion

Within this section, we dive into more depth in various aspects of both the dataset and the model. Furthermore, we present a different perspective, by critically evaluating the results and gauging the model's level of comprehension. Moreover, we give guidelines for further research on this specific model, for people to explore its advantages and limitations.

A. Dataset and Model Evaluation

Let's begin by pointing out that assessing the quality of a music sample is really difficult by its nature. Plasser et al. in [4] argue that statistical metrics are poor for this task, and

follow Naem et al. [10] who introduce the concepts of fidelity and diversity to be the aspects of a good model. In addition, Hsiao et al. [11], believe that you can assess these attributes, by listening to the generated samples, an opinion that we highly agree with.

In addition, we believe that when assessing a model, there must be some context to evaluate the quality of the samples. This is the main reason why we chose the strict music of Bach when it came to the dataset, and not something more modern. In Bach, and generally in Baroque Music, there are many rules that apply over the pieces, which tend to result in creating a specific atmosphere and sound. This is what we want to look for when listening the samples. We are happy to say, that the results of our model do indeed have this attribute. Moreover, we believe the model captured the notion of the key, as all the generated samples are in C (although C minor was more common than C major), and include modulations in nearby keys.

However, there are still some things we need to address. We decided to heavily compact the pieces in the time axis, which led to having longer pieces in shorter images, but created the problem of consecutive same notes being merged. That is the reason why in the infilling samples, the given melody is somehow lost, because when setting the neighboring pixels of a note to 1, then the note is just spread. Furthermore, we noticed that the model often generates samples that are almost blank (have only some noise left). Not only that, the model seems to favour less dense segments, and usually generates samples with just two voices. Although this may not be considered as a problem, it can be easily overcome by checking the ratio of 1s in the image, and if it is under a specific threshold the sample is discarded and the model generates a new one. However, here lies one of the biggest flaws of our work. As we said earlier, we used the 24 fugues of the first Well-Tempered Clavier book. Among these 24 pieces, there is only one that is two-voiced - the Fugue no. 10 in E minor, therefore, the model is overfit on that specific piece. This is evident, as when generating segments with 2 voices, the motifs of that fugue are continuously used. Nevertheless, the level of imitating the original piece stays on these motifs and not on full phrases.

Finally, we delve into an examination of the model's level of music comprehension. We argue that the representation of music as images results in a considerable loss of understanding. The distinctive nature of music as an art form, characterized by sounds unfolding over time, does not align seamlessly with the static nature of images. The utilization of image representations in training a model designed for images involves an unnecessary expenditure of sources and energy. The generated binary images lead the model to attend to the entire image, rather than focusing solely on the notes constituting the melody. To surmount this limitation, we advocate for the introduction of a novel method for representing music that captures its inherent temporal artistic nature more effectively. An obvious starting point would be the method of "bag of words", used in Large Language Models, and a "word" would correspond to a note. This way, single melody lines can be easily modeled, but the difficulty would be to generalize it for multiple parallel lines.

B. Guidelines for Further Research

Regarding this model, there are numerous avenues for exploration and experimentation. A straightforward approach involves retraining the model using diverse datasets. These datasets can vary in terms of size, be generated through different augmentation techniques, and encompass entirely distinct musical compositions, such as the Chopin Mazurkas. Subsequently, an evaluation of the model's performance across these variations can illuminate its optimal behaviors and reveal limitations in conceptualizing musical motifs.

Furthermore, leveraging the non-binary nature of images opens the possibility of incorporating dynamics into the model. This avenue could enhance the model's capacity to capture nuanced musical expressions. Additionally, the infilling task can be subjected to further experimentation. For instance, the baseline could be fixed while allowing the model to generate the melody, or alternatively, when providing a middle voice melody line, fixing a region around it rather than solely constraining the individual notes in use.

VII. Conclusion

In this paper, we introduced a novel dataset and outlined a method for generating binary image datasets from MIDI files and vice versa. Subsequently, we employed this dataset to train an existing Diffusion Model with UNet architecture, and then utilizing the trained model to generate samples. The model underwent testing in the tasks of unconditional generation and infilling, yielding results that exhibited similarity to the input dataset. Additionally, we shared our reflections on our work and addressed certain limitations inherent in this approach to music generation. These limitations stem from the inherent constraints of binary piano rolls as images, which prove insufficient in representing various aspects of music composition.

Acknowledgment

I thank Eleftherios Tsonis, Spyros Kantarelis and the rest of the AILS lab team for their guidance, feedback, and encouragement.

References

[1] G. Mittal, J. Engel, C. Hawthorne, and I. Simon, "Symbolic music generation with diffusion models," ISMIR, 2021. arXiv:2103.16091

[2] A. Roberts, J. Engel, C. Raffel, C. Hawthorne, and D. Eck, "A hierarchical latent vector model for learning long-term structure in music," in Proceedings of the 35th International Conference on Machine Learning, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. PMLR, 10–15 Jul 2018, pp. 4364–4373. http://proceedings.mlr.press/v80/roberts18a.html

[3] C. Raffel, "Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching," Ph.D. dissertation, Columbia University, 2016. https://doi.org/10.7916/D8N58MHV

[4] M. Plasser, S. Peter, and G. Widmer, "Discrete Diffusion Probabilistic Models for Symbolic Music Generation," IJCAI, 2023. arXiv:2305.09489

[5] J. Austin, D. Johnson, J. Ho, D. Tarlow, and R. Van Den Berg, "Structured denoising diffusion models in discrete state-spaces," in Advances in Neural Information Processing Systems, volume 34, pages 17981–17993. Curran Associates, Inc., 2021.

[6] L. Atassi, "Generating symbolic music using diffusion models," University of California, 2023. arXiv:2303.08385

[7] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in International Conference on Medical image computing and computer-assisted intervention. Springer, 2015, pp. 234–241.

[8] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models", Advances in Neural Information Processing Systems, vol. 33, pp. 6840–6851, 2020.

[9] A. Bansal, E. Borgnia, H.-M. Chu, J. S. Li, H. Kazemi, F. Huang, M. Goldblum, J. Geiping, and T. Goldstein, "Cold diffusion: Inverting arbitrary image transforms without noise," 2022. arXiv:2208.09392

[10] M. F. Naeem, S. J. Uh, Y. Uh, Y. Choi, and J. Yoo. "Reliable fidelity and diversity metrics for generative models".

[11] W. Y. Hsiao, J. Y. Liu, Y. C. Yeh, and Y. H. Yang. "Compound word transformer: Learning to compose full-song music over dynamic directed hypergraphs". In Proceedings of the AAAI Conference on Artificial Intelligence, volume 35, pages 178–186, 2021.