Reducing Negative Log-Likelihood of Convolutional Music Compositions

Luke Griswold
lukedgriswold@gmail.com
Univercity of Central Florida
Orlando, Florida

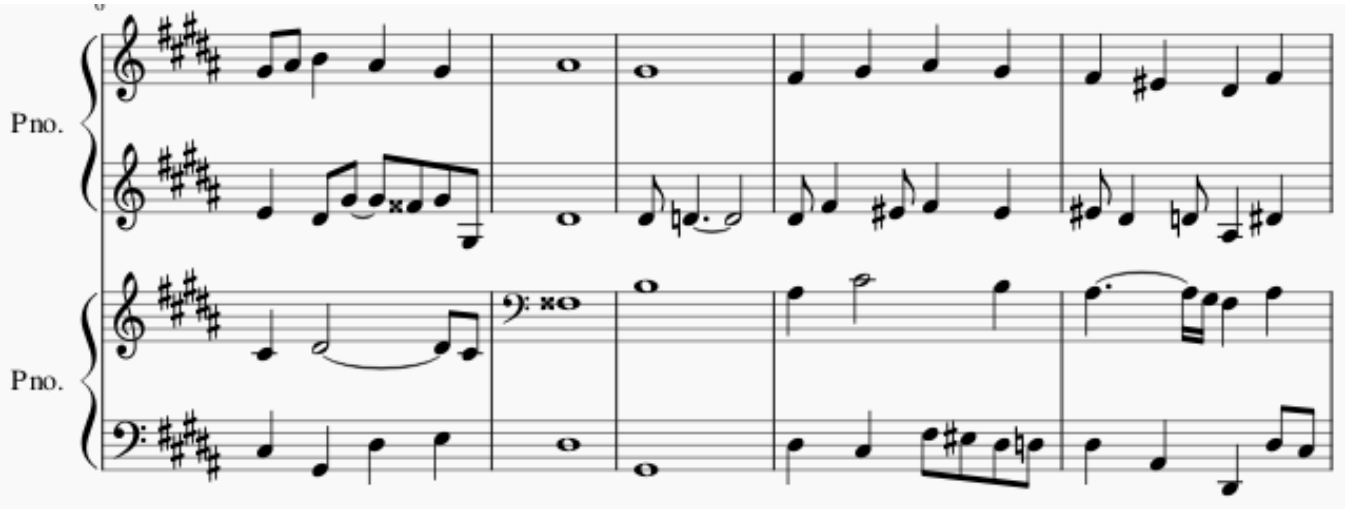

Figure 1: Transposition of BWV364 using Music21 and Piano Rolls

ABSTRACT

One promising approach to creating harmonies for Artificial Intelligence music generation is to use Convolutional Neural Networks (CNNs) in conjunction with a process of re-sampling the generated music. This paper explores one such approach and looks at the specific quantifiable evaluation metric of negative log likelihood in relation to subjective sample quality. It appears to the author that time steps are a major component of both sample quality and pitch-probability convergence, although there remains optimizations in the training process to be explored.

CCS CONCEPTS

• **Computing methodologies** → *Probabilistic reasoning*; *Temporal reasoning*.

KEYWORDS

music, neural networks, gibbs samping

UCF Machine Learning, December, 2019, Orlando, FL

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

ACM Reference Format:
Luke Griswold. 2019. Reducing Negative Log-Likelihood of Convolutional Music Compositions. In *Proceedings of Univercity of Central Florida: Machine Learning Class (UCF Machine Learning).* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Music and harmonization have a long tradition of being targets for Artificial Intelligence systems. Many of these works focus on harmonizing in four parts based on Bach's Chorales [4, 5, 8, 10] due to the excellent example they serve for counterpoint [1] and their relatively simple data structure. It is possible to transform music into a multitude of different data structures to feed as input into ML models, as discussed in section 3.1. Because chord structure is invariant to transposition and cadences occur at various parts of a score temporally, a Convolutional Neural Network is used with a tensor representation of a musical score.

This paper aims at taking a similar approach to [5] in order to build a Convolutional Neural Network and explore the relationship between the temporal resolution of music samples and the average Negative Log Likelihood of replaced notes during Gibbs Sampling. Section 2 goes over historical approaches which often used Recurrent Neural Networks (RNN)s and the problem description that favors Convolution. Section 3 goes over the data preparation for model input, the model architecture, and the model output. Section 4 has a short evaluation of the training of the models under two

different temporal resolutions which leads into the discussion of Gibbs sampling in section 5.

2 RELATED WORK

2.1 Sequence Generation Perspective

As Briot et al. point out in their excellent overview of this problem, music generation by computer systems has its roots in grammar based predictors but has begun transitioning to newer deep learning technieques [2]. Generating music can be seen from a variety of different perspectives which influence how music should be represented as data as well as what kind of network is best for solving the problem.

One such approach is to define music as a task of predicting a note or chord based on the note or chord that preceded it. As Kotecha points out, this approach yields to probabilistic models where the distribution of a sequence is learned from the training set [8]. Kotecha's paper is aimed at tackling this sequence generation task through generating a note's duration and pitch with a Bi-axial Long Short Term Memory (LSTM) network note by note. Another take on this approach has been to use a Generative Adversarial Network (GAN) with LSTM cells in order to achieve better results. C-RNN-GAN was developed by Olof Mogren as an example of such an approach, but one that uses a continuous representation of music vice a discrete, symbolic representation [9]. Finally, an approach developed by Hadjeres et al. is the DeepBach architecture, which uses both the future and the past information of notes in two RNNs with LSTM cells, along with a Neural Network (NN) for notes occurring at the same time, as input for the final network which outputs probabilities for notes [4]. Perhaps most importantly, they discuss the importance of the data representation and the use of "Pseudo-Gibbs sampling" to re sample the generated music.

2.2 Pattern Recognition Perspective

Another approach to music comes from music theory and is based on the idea that relative and not absolute relationships define the tone structure of a phrase [1]. In this sense, music can be seen as a build-up of musical tension, and the release of that tension through a cadence [1]. There are a finite number of these patterns and they can occur in different keys through modulation. The rules of modulation were described, and in Bernstein's opinion perfected, by Bach and should therefore be susceptible to pattern detection in neural networks.

This is the idea behind convolutional approaches to music generation [2]. While some of the above architectures utilized convolutional kernels, most notably the Bach2Bach presented by Kotecha [8], a more traditional CNN is represented by COCONET [5], which powered the AI behind the March 20, 2019 Google Doodle celebrating J.S. Bach's 334th birthday [6]. This model takes a user-generated musical phrase and composes harmony through a combination of a CNN and Gibbs Sampling to iteratively rewrite sections of generated harmony, and a similar model is the focus of this project.

3 METHODS

3.1 Data Preparation

Briot et al. go to great lengths to describe the possible ways to represent music as data [2]. In order to simplify the pre-processing, Music21 was used. Music21 is a python library that simplifies manipulation of MIDI files, can produce MIDI files, and can also work with MusicXML [3]. It also comes with a corpus containing all of the Bach Chorales, which will be the targeted Data Set for this project.

The ultimate aim would be to train a GAN or use a separate RNN for melody generation. Due to time and computing constraints, a single CNN was used to study the effect of temporal resolution on sample quality. The model's architecture is what Briot would describe as a global temporal granularity [2], where pitches and discrete time form a matrix for each voice in the score.

Similar to [5], this project uses the individual voices as a matrix representing discrete time and a one-hot pitch vector over the 128 chromatic midi pitches available. The training and validation data is created by transforming the Music21 corpus files into four of these piano rolls (one for each voice), which requires picking a sampling rate from the score. 1/8th notes and 1/16th notes were used as the two sampling rates for this project. The score is then randomly chopped into 4 measures of contiguous music to keep the time length the same. Finally, this is then transposed up or down by 0-11 chromatic steps. This turns the 167 Chorales used that are in 4:4 time signature to over 7,348 possible 4-measure slices. Because transposition should be handled well by the convolutional filters, not all of these 7,348 possibilities was used for each epoch, but a sample was drawn from them instead. A small subset of the 167 Chorales was held back for validation. As such, a section of a Bach Chorale is represented as a piano roll with dimensions:

$$\mathbf{x} \in \{0,1\}^{IxTxP}$$

where I, T, P represent the instruments, time steps, and pitches. For this model they are 4, 32 or 64, and 128, respectively. To create model input, we select a subset of all the elements in x to include as the context, C. We then append a boolean mask with a 1 where the time step's note is known and a 0 where it has been erased to concatenate with $\mathbf{x}$ and get the input, h:

$$\mathbf{h} \in \{0,1\}^{2IxTxP}$$

3.2 Model Architecture

Two different models had to be trained, one for each temporal resolution, as 4 measures of 4:4 create 64 time steps when sampled at 1/16th note resolution, and 32 time steps at 1/8th note. The model architectures in every other regard are the same. The model takes as input the four piano rolls for the Soprano, Alto, Tenor, and Bass (SATB) as well as their boolean masks. If the notes that remain from the training data can be thought of as the set C, drawn from the entire set of notes, then the output of the model is $p(x_{i,t,p}|x_C)$, where i, t, p represents the instrument, time step, and pitch, and x_C represents all the known pitches.

The model uses 20 convolutional layers with 64 filters of 5x5 kernels. Batch Normalization and Padding are used to keep the input size the same in the time and pitch dimensions for each layer.

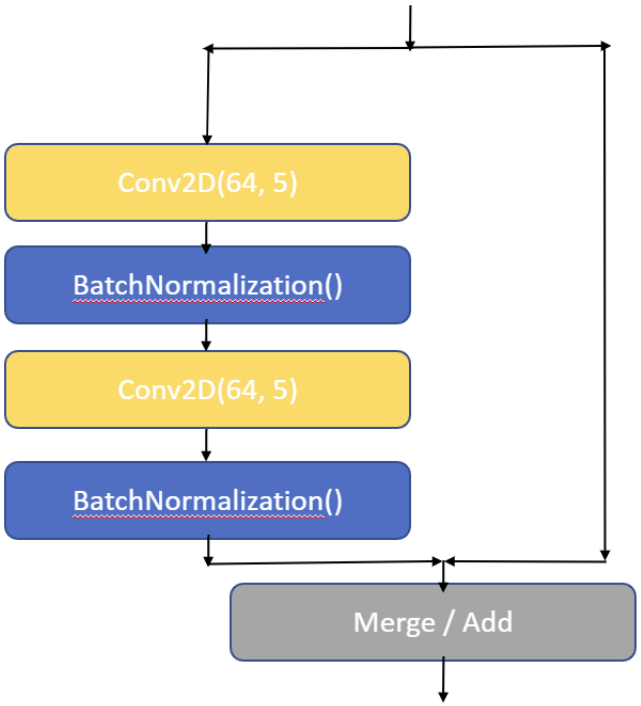

Figure 2: Iterated CNN Architecture

The model iteratively connects layers found in figure 2 where every two convolutional layers includes a skip connection to maximize residual learning as in [7]. The convolutional layers use ReLU activations, with the exception of the final layer, which uses four filters and a softmax activation in order to output the probability distribution over the 128 pitches at each time step.

The loss function is based on Uria et. al's work on something called the Nerual Autogenerative Distribution Estimator [11], where we are minimizing the "average" negative log likelihood of the true pitches for the replaced notes. The average comes from the fact that the gradient depends on all of those terms and on the different possible orderings of sampling form them, so that is why total log likelihood is divided by the number of notes. Huang et. al in [5] translate the loss function into the task of generating music for a set of known notes C, and erased notes $\neg C$ as:

$$\mathbb{L}(\mathbf{x}; C, \theta) = -\sum_{i,t,p \notin C} x_{i,t,p} \log p_\theta(x_{i,t,p}|x_C) \tag{1}$$

where $p_\theta(x_{i,t,p}|x_C)$ is the output of the model with parameter vector θ. The training model then minimizes

$$\mathbb{E}_{x\sim p(x)}\mathbb{E}_{C\sim p(C)}\frac{1}{|\neg C|}\mathbb{L}(\mathbf{x}; C, \theta) \tag{2}$$

The expected values in equation 2 are found from the fact that working in batches of 20 samples, the model trains on the same subset C of known notes, which are generated randomly for each batch of 20. The stochastic gradient descent is then controlled by the Adam optimizer in Keras. Admittedly, this kind of training and evaluation is hard to quantify and based on the results from section 5.1, more time spent evaluating the training process may improve the results. Unfortunately one iteration of training takes considerable amount of time so the focus remained on the sampling process for inferences.

Table 1: Training Loss and Validation Negative Log Likelihood

Temporal Resolution	Min Training Loss	Validation NLL
1/8	.487	1.6
1/16	.168	1.03

4 EVALUATION

The github repository for this project contains sample music to be discussed in the next two sections and can be found at https://github.com/Kickflip89/Convolution-Music-AI. For the validation data, two techniques were used. Real-time, validation was conducted by doing block sampling of the erased notes and determining the average negative log likelihood of the sampled notes WITHOUT re sampling, or comparing the true notes to the sampled notes. By doing this in batches, one can get the framewise Negative Log Likelihood (NLL) for a piano-roll and the results are shown in table 1. Section 5.1 will discuss a different method of evaluating using Gibbs sampling, which was proposed by [12].

Huang et. al were able to acheive better NLLs from validation [5], which may have some bearing on the discussion section below. Training was conducted over five epochs with 347 steps of 20 batches. In both the 1/16th case and the 1/8th case the loss function continued to smoothly decrease to its min value over the steps and epochs, but based on the Gibbs sampling process it appears the model may have been overfit and tuning the training process is probably a likely source to improve the sample quality of the music.

5 DISCUSSION

5.1 Gibbs Sampling to Improve Music Quality

One downside with sampling all erased notes in a block is that they may not have enough context from the remaining notes in C to accurately predict the probabilities. An approach proposed by Uria et. al about NADE was to sample pitch distributions one time step at a time in order to allow the probability distributions to change as one variable after the other is added [11]. This approach was added onto by [5, 12] to include an annealed mask that allows the score to rewrite itself without committing to any notes other than those fixed as input. By combining block sampling at the beginning of the process with independent and individual pitch sampling toward the end, it is more likely to keep the model from simply staying on the same note through time.

The basic idea is to run the input through the model, sample the notes in $\neg C$, and then use a decreasing probability function to randomly select the notes from $\neg C$ to include as input to the next generation. The full probability mask depends on a few hyperparameters and is given as:

$$\alpha_n = \max(\alpha_{\min}, n(\alpha_{\max} - \alpha_{\min})/\eta N) \tag{3}$$

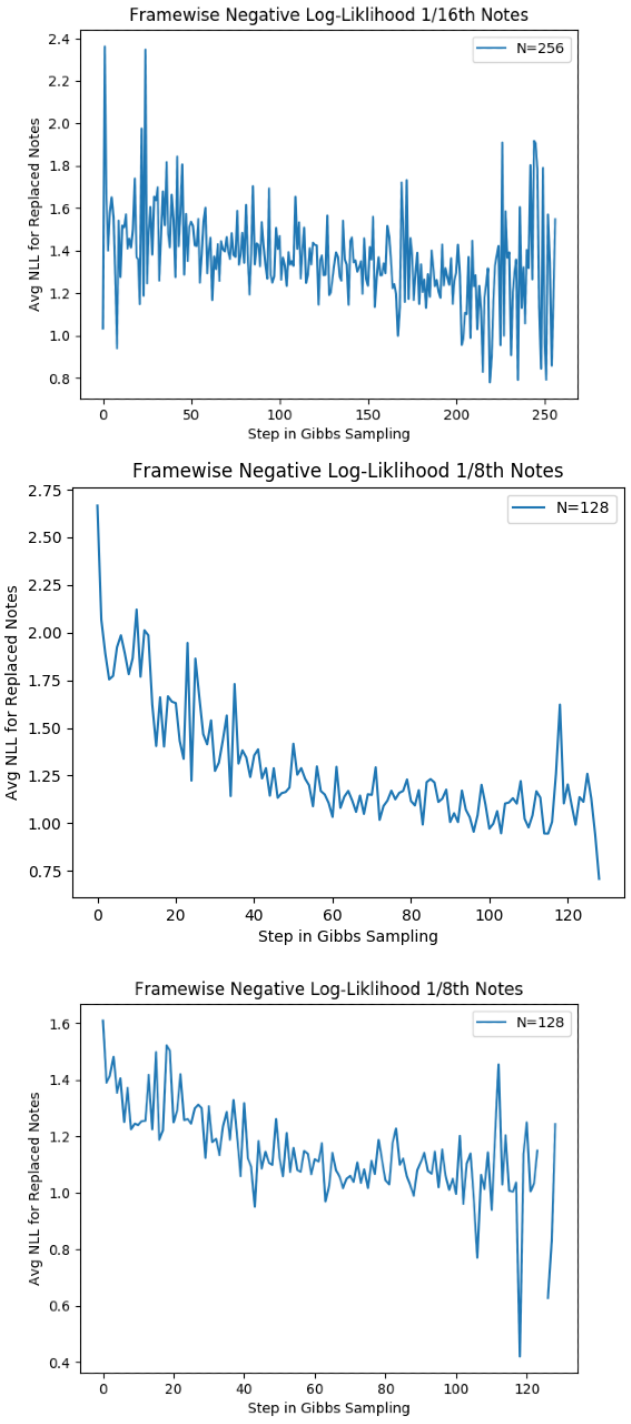

Figure 3: avg NLL of replaced notes at Gibbs steps with: TOP-1/16th notes, blocked erases; MID-1/8th notes, blocked erases; BOT-1/8th notes, random non-melody erases

where η is the fraction of time to be spent above $\alpha_{\min}$, and N are the number of iterations in the sampling process. As described in [5], N was set to $I * T$, or 256 and 128 for 1/16 and 1/8 temporal resolution, respectively. If the model is successfuly taking advantage of Gibbs sampling, then the average NLL for a resampled note should decrease gradually as the masking probability becomes smaller. In this way, as more and more notes are aded to the context, C, the probabilities for the erased notes should converge to become higher for the notes that are most likely. The avg NLLs for different approaches are shown in figure 3 and the algorithm is given, where sampled(S) replaces probabilities with pitches sampled from each time steps probabilty distribution (where the mask is 0).

Algorithm for Gibbs Sampling on input I

$S \leftarrow$model(I)
$S \leftarrow$sampled(S)
for step from 1..N
mask$\leftarrow$ I's original masks
  $p_m \leftarrow \alpha_n$
  for I from 1..4
    for t with mask$_{i,t}$ = 0
      mask$_{i,t} \leftarrow$1, with $p(1 - p_m)$
    end for
  end for
  $S \leftarrow S$*mask (elementwise)
  $S \leftarrow$ Cat(S,mask)
  $S \leftarrow$model(S)
  $S \leftarrow$sampled(S)
end for
return S

Interestingly, although the training loss was lowest for the 1/16th resolution, the Gibbs sampling procedure showed the least reduction in average NLL from blocked sampling. In [5] et. al it is proposed that the model may show decreased average NLL at faster temporal resolution because the previous and next time step tends to give much more information about the current time step. In a quarter note resolution, the next and previous notes are often different, whereas at 1/16th note resolution they are often the same. Thus, 1/4 or 1/8 temporal resolution offers a balance between prioritizing what the other voices are doing, and prioritizing what the melody of that voice is doing in time.

The temporal resolution of 1/8th notes showed a much improved Gibbs sample when large chunks of the sample was erased uniformly. However, when the melody was kept in place and random chunks of the ATB voices were erased, the 1/8th model also didn't show much reduction in avg NLL over the Gibbs steps. The github repository has sample music from all three techniques, labeled "sixteenth", "quarter", and "sample", respectively.

5.2 Subjective Evaluations

Although less formal, listening to samples from the music also yield clues to where the model may be going off track in producing lower quality music samples. In the sixteenth note music, it does appear there is a high premium placed on maintaining the previous or next pitch in a voice. It is not immediately clear how to avoid this for this time step as defining the loss as maximizing the probability for the correct "erased" pitch seems to be the most intuitive solution.

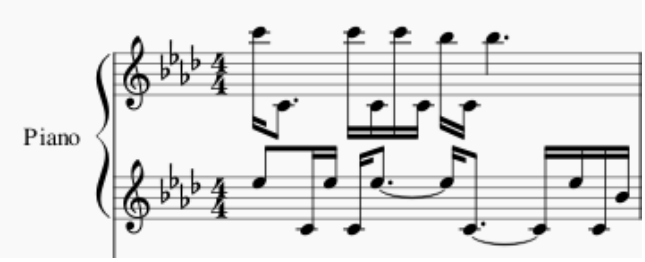

Figure 4: Generated music's SA voices showing high reliance on middle C

Figure 4 also shows a section of music from the "sixteenth" sample at the github repository that shows a huge reliance on C note in the S and A voices (this is also shown in the T voice).

In general the samples from the 1/8th note temporal resolution are of slightly higher subjective quality, although both models are capable of generating pleasant sounding harmonies. The biggest issue is typically that true cadences are rare in the sampled music. While at times there appears to be a melodic structure and even chord structure, it is hard to say exactly what key the piece is in because of the lack of any cadence back to the tonic. In this sense, imposing a sense of structure to the piece by strategically including cadences as part of the context, C, could improve the subjective quality of the music.

6 CONCLUSION

Music remains an inherently human endeavor that combines elements of style, mood, emotive response, structure, and harmony. While many AI models are able to achieve some success in certain areas, it is very difficult to design a system that achieves moderate success in all of them. This project's aim was to achieve some success in developing harmony by iterating over generated conditional probability distributions.

To that end, some success was garnered by following some techniques from NADE and COCONET [5, 11], but there appears to be more work to be done in the training phase and in figuring out the best way to utilize Gibbs Sampling to allow the probability distributions to converge on coherent pitch structures. The 1/16th note model may have been overfit because it was conditioned by the loss function to place higher values on maintaining the same note. The 1/8th model showed more promise by decreasing the avg NLL when large sections of music were erased, although when the melody was retained and note erasures were random, there was not a large reduction in avg NLL through the Gibbs procedure.

REFERENCES

[1] L. Bernstein. 1981. *The Unanswered Question: Six Talks at Harvard (The Charles Eliot Norton Lectures).* Harvard University Press.
[2] J. Briot, G. Hadjeres, and F. Pachet. 2019. *Deep Learning Techniques for Music Generation, Computational Synthesis and Creative Systems.* Springer.
[3] M. Cuthbert and C. Ariza. 2010. music21: A Toolkit for Computer-Aided Musicology and Symbolic Music Data. In *Proceedings of the International Society for Music Information Retrieval.*
[4] G. Hadjeres, F. Pachet, and F. Nielsen. 2010. DeepBach: a Steerable Model for Bach Chorales Generation. In *Proceedings of the 34th International Conference on Machine Learning.*
[5] C. Huang, T. Cooijmans, A. Roberts, A. Courville, and D. Eck. 2017. Counterpoint by Convolution.. In *Proceedings of the 18th International Society for Music Information Retrieval Conference.*
[6] C. Huang and et. al. 2019. *Coconet: the ML model behind today's Bach Doodle.* Retrieved March, 2019 from https://magenta.tensorflow.org/coconet
[7] H. Kaiming, Z. Xiangyu, and R. Shaoqing. 2015. *Deep residual learning for image recognition.* arXiv preprint arXiv:1512.03385.
[8] N. Kotecha. 2019. *Bach2Bach: Generating Music Using a Deep Reinforcement Learning APproach.* arXiv:1812.01060.
[9] O. Mogren. 2016. C-RNN-GAN: Continuous recurrent neural network with adversarial training. Constructive Machine Learning Workshop.
[10] S. Skuli. 2017. *How to Generate Music using a LSTM Neural Network in Keras.* Retrieved December, 2019 from https://towardsdatascience.com/how-to-generatemusic-using-a-lstm-neural-network-in-keras-68786834d4c5
[11] B. Uria, I. Murray, and H. Larochelle. 2014. *A deep and tractable density estimator.*
[12] L. Yao, S. Ozair, K. Cho, and Y. Bengio. 2014. On the equivalence between deep nade and generative stochastic networks. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases.* Springer.