Self-Correcting Non-Chronological Autoregressive Music Generation

Wayne Chi [*1] **Prachi Kumar** [*1] **Suri Yaddanapudi** [1] **Rahul Suresh** [1] **Umut Isik** [1]

Abstract

We describe a novel approach for generating music using a self-correcting, non-chronological, autoregressive model. We represent music as a sequence of edit events, each of which denotes either the addition or removal of a note—even a note previously generated by the model. During inference, we generate one edit event at a time using direct ancestral sampling. Our method allows the model to fix previous mistakes such as incorrectly sampled notes and prevent the accumulation of errors which autoregressive models are prone to have. Another benefit is a finer, note-by-note control during human and AI collaborative composition. We show through human survey evaluation that our approach generates better results than orderless NADE and Gibbs sampling.

1. Introduction

There have been two primary approaches to generating music with deep neural network-based generative models. One treats music generation as an image generation problem (Dong et al., 2018; Yang et al., 2017), while the other treats music as a time series generation problem analogous to autoregressive language modeling (Huang et al., 2018; Child et al., 2019; Dai et al., 2019; Payne, 2019; Wu et al., 2019). We propose a method that uses elements from both the image-based and time series generation techniques. Our method operates on piano roll images with a 2D convolutional neural network, but autoregressively adds or removes notes one at a time in an arbitrary, non-chronological order. We model the conditional distribution of note add or remove events given pre-existing notes. After each sampling iteration, we re-input the resulting piano roll into the model to get the distribution of the next add and remove event.

A benefit of our approach is the model's ability to fix past mistakes. Poor samples due to accumulation of errors is a well-documented problem with autoregressive models (Bengio et al., 2015; Huszár, 2015; Lamb et al., 2016; Venkatraman et al., 2015), especially when directly sampling from the conditional distribution (i.e. direct ancestral sampling). While Gibbs Sampling (Huang et al., 2019a) can help this, we show that direct ancestral sampling is sufficient if the data representation includes removal of past samples. In our use case, users use the model to enhance a musical melody. Since our method generates notes one-by-one and non-chronologically, another benefit is a finer degree of human control which allows closer human-AI collaboration.

Following the introduction of NADE (Larochelle & Murray, 2011; Uria et al., 2016) and orderless NADE (Uria et al., 2014) there have been several works built upon the concept of ordered and unordered autoregressive models. Coconet (Huang et al., 2019a)—the algorithm behind Google's Bach Doodle[1]—uses a convolutional model to generate music by adding counterpoints to existing user input using Gibbs sampling, whereas we use direct ancestral sampling and explicitly model the removal of notes. In a NLP setting, recent works also explore non left-to-right ordering (Stern et al., 2019; Chan et al., 2019) and deletion (Gu et al., 2019). PixelCNN (Oord et al., 2016) models pixels in an image autoregressively in a pre-specified order while our generation is unordered. Many different RNN (Boulanger-Lewandowski et al., 2013; Chu et al., 2016), LSTM (Sturm et al., 2016), Transformer (Huang et al., 2018), and GAN based methods (Dong et al., 2018; Yang et al., 2017; Mogren, 2016) have also shown the ability to generate high quality music.

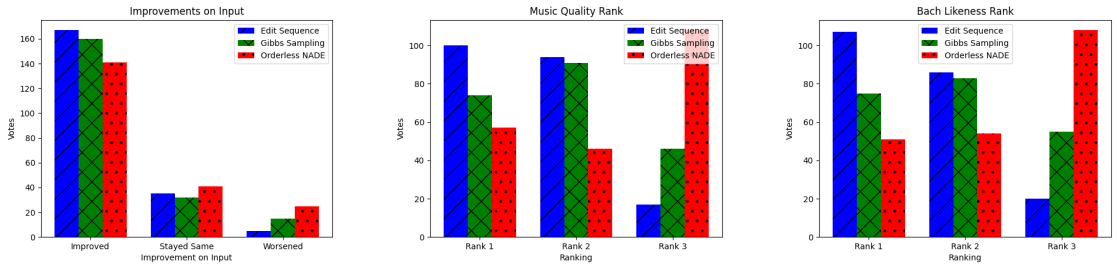
2. Problem Definition

We consider a musical piece $x \in X$ as a point in $\{0,1\}^{T \times P}$ where T is the number of time steps and P is the number of note pitches. This represents a simplified piano roll (PR)—a discrete representation of music as an image matrix across pitch and time. There exists a probability density function $p^{\text{PR}}(x)$ on $\{0,1\}^{T \times P}$ of musical pieces. Instead of modeling $p^{\text{PR}}(x)$ on $\{0,1\}^{T \times P}$ directly, we model the distributions as $p^{\text{ES}}(x)$ on the set of **edit sequences** (ES). An **edit sequence** of length M is a tuple of M-many **edit events** where an **edit event** is a matrix $e^{(t,p)} \in \{0,1\}^{T \times P}$ that has one entry equal to one, and all other entries equal to zero (i.e. a one-hot matrix). We denote the set of all edit

[*]Equal contribution [1]Amazon Web Services. Correspondence to: Wayne Chi <waynchi@amazon.com>, Prachi Kumar <kumprach@amazon.com>, Suri Yaddanapudi <yaddas@amazon.com>, Rahul Suresh <surerahu@amazon.com>, Umut Isik <umutisik@amazon.com>.

Proceedings of the 37^{th} International Conference on Machine Learning, Vienna, Austria, PMLR 108, 2020.

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



(a) (b) (c)

Figure 1. Human Survey Evaluation Ratings: (a) describes whether users thought a sample improved on the input. (b) describes user rankings for music quality. (c) describes user rankings for how similar a sample is to real Bach data.

events by $\mathcal{E}$ and of edit sequences of length M by $\mathcal{E}^M$. The following maps edit sequences to piano rolls:

$$\pi : \bigcup_{M=1}^{\infty} \mathcal{E}^M \to \{0,1\}^{T\times P} \tag{1}$$

$$\pi(e_1,\ldots,e_M) = \sum_{i=1}^{M} e_i \pmod 2. \tag{2}$$

where (2) allows edit events to handle either note addition or removal depending on if a previous edit event exists at the same time and pitch. The mapping between the two joint probability distributions is as follows:

$$\begin{aligned} p^{\mathrm{PR}}(x) &= p^{\mathrm{PR}}(\{(t_1,p_1),\ldots,(t_N,p_N)\}) \\ &= p^{\mathrm{ES}}(x) = \sum_{s\in\pi^{-1}(x)}^{\infty} p^{\mathrm{ES}}(s) \end{aligned} \tag{3}$$

where N is the number of notes in the piano roll, (t_i,p_i) is the time and pitch of a note or edit event, $\pi^{-1}(x)$ is the inverse image of $\pi(x)$, and s is a sequence of edit events $(t_1,p_1)\ldots(t_M,p_M)$ where $M \geq N$. We can further factorize $p^{\mathrm{ES}}(s)$ as:

$$\begin{aligned} p^{\mathrm{ES}}(s) &= p^{\mathrm{ES}}\big((t_1,p_1),\ldots,(t_M,p_M)\big) \\ &= \prod_{i=1}^{M} p^{\mathrm{ES}}\big((t_i,p_i)|(t_1,p_1),\ldots,(t_{i-1},p_{i-1})\big) \end{aligned} \tag{4}$$

We assume that $p^{\mathrm{ES}}((t_i,p_i)|(t_1,p_1),\ldots,(t_{i-1},p_{i-1}))$ is ordering invariant. Our goal is to train a model that maps the distribution of edit sequences $p^{\mathrm{ES}}(x)$ by training on X. By sampling autoregressively from $p^{\mathrm{ES}}(x)$, we will generate a sequence of edit events that can be mapped back into a piano roll representation and then converted to MIDI.

3. Training and Inference

Given an input piano roll, $\mathcal{I}$, and target piano roll, $\mathcal{T}$, we train our model to output the conditional probabilities $p^{\mathrm{ES}}((t_i,p_i)|(t_1,p_1),\ldots,(t_{i-1},p_{i-1}))$ of the next edit event in edit sequences that can recreate $\mathcal{T}$ from $\mathcal{I}$. For each piano roll in the training set, we mask random existing notes and add random extraneous notes to it to generate $\mathcal{I}$. Since we assume ordering invariance in (4), we can also assume that every note difference between $\mathcal{I}$ and $\mathcal{T}$ is equally likely to be the next edit event. Thus, we model the distribution of edit events for the next step as the uniform distribution U supported on the symmetric difference $\mathcal{I}\Delta\mathcal{T}$ between $\mathcal{I}$ and $\mathcal{T}$. We use the Kullback-Liebler divergence between U and the model's output distribution as the loss function:

$$\mathcal{L}(\mathcal{I},\mathcal{T},P) = D_{KL}(P \parallel U), \tag{5}$$

where P is the softmax over the model's output for all times and pitches.

During inference, we sample from the model's output probabilities through direct ancestral sampling. We feed the input melody to the model, sample the next edit event from the P, modify the input based on that edit event, and then feed that melody back into the model. We repeat this for each iteration, each time conditioning on our previous predictions. Since we do not differentiate between adding and removing notes during training, the sampling process is the same for any type of edit event.

4. Empirical Evaluation

We conduct a human opinion survey to compare our approach[2] against orderless NADE and Gibbs sampling using a human survey evaluation. For our comparisons, we build an orderless NADE (Uria et al., 2014) model and use Coconet (Huang et al., 2019a) to represent a Gibbs sampling approach. We use the JSB Chorales dataset[3] which contains 382 midi files of chorales harmonized by J.S. Bach. We generated 8 bar samples conditioned on 15 monophonic input tracks, assuming 4 beats per bar and 16 time steps per bar (i.e. sixteenth-note quantization). For the input tracks, we randomly chose from a dataset of real user inputs from the Bach Doodle Dataset (Huang et al., 2019b). For each set of comparisons, users were asked if each sample improved on the input and to rank the samples based on music quality and based on their similarity to music composed by Bach. In Figure 1 we see that our approach outperforms both orderless NADE and Gibbs Sampling overall.

[2]We trained a U-Net (Ronneberger et al., 2015), but the choice of architecture is not critical to our approach.

[3]https://github.com/czhuang/JSB-Chorales-dataset

References

Bengio, S., Vinyals, O., Jaitly, N., and Shazeer, N. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pp. 1171–1179, 2015.

Boulanger-Lewandowski, N., Bengio, Y., and Vincent, P. High-dimensional sequence transduction. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 3178–3182. IEEE, 2013.

Chan, W., Kitaev, N., Guu, K., Stern, M., and Uszkoreit, J. Kermit: Generative insertion-based modeling for sequences. *arXiv preprint arXiv:1906.01604*, 2019.

Child, R., Gray, S., Radford, A., and Sutskever, I. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

Chu, H., Urtasun, R., and Fidler, S. Song from pi: A musically plausible network for pop music generation. *arXiv preprint arXiv:1611.03477*, 2016.

Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q. V., and Salakhutdinov, R. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

Dong, H.-W., Hsiao, W.-Y., Yang, L.-C., and Yang, Y.-H. Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Gu, J., Wang, C., and Zhao, J. Levenshtein transformer. In *Advances in Neural Information Processing Systems*, pp. 11181–11191, 2019.

Huang, C.-Z. A., Vaswani, A., Uszkoreit, J., Simon, I., Hawthorne, C., Shazeer, N., Dai, A. M., Hoffman, M. D., Dinculescu, M., and Eck, D. Music transformer: Generating music with long-term structure. 2018.

Huang, C.-Z. A., Cooijmans, T., Roberts, A., Courville, A., and Eck, D. Counterpoint by convolution. *arXiv preprint arXiv:1903.07227*, 2019a.

Huang, C.-Z. A., Hawthorne, C., Roberts, A., Dinculescu, M., Wexler, J., Hong, L., and Howcroft, J. The Bach Doodle: Approachable music composition with machine learning at scale. In *International Society for Music Information Retrieval (ISMIR)*, 2019b. URL https://goo.gl/magenta/bach-doodle-paper.

Huszár, F. How (not) to train your generative model: Scheduled sampling, likelihood, adversary? *arXiv preprint arXiv:1511.05101*, 2015.

Lamb, A. M., Goyal, A. G. A. P., Zhang, Y., Zhang, S., Courville, A. C., and Bengio, Y. Professor forcing: A new algorithm for training recurrent networks. In *Advances In Neural Information Processing Systems*, pp. 4601–4609, 2016.

Larochelle, H. and Murray, I. The neural autoregressive distribution estimator. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pp. 29–37, 2011.

Mogren, O. C-rnn-gan: Continuous recurrent neural networks with adversarial training. *arXiv preprint arXiv:1611.09904*, 2016.

Oord, A. v. d., Kalchbrenner, N., and Kavukcuoglu, K. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*, 2016.

Payne, C. Musenet, 2019. *URL https://openai.com/blog/musenet*, 2019.

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

Stern, M., Chan, W., Kiros, J., and Uszkoreit, J. Insertion transformer: Flexible sequence generation via insertion operations. *arXiv preprint arXiv:1902.03249*, 2019.

Sturm, B. L., Santos, J. F., Ben-Tal, O., and Korshunova, I. Music transcription modelling and composition using deep learning. *arXiv preprint arXiv:1604.08723*, 2016.

Uria, B., Murray, I., and Larochelle, H. A deep and tractable density estimator. In *International Conference on Machine Learning*, pp. 467–475, 2014.

Uria, B., Côté, M.-A., Gregor, K., Murray, I., and Larochelle, H. Neural autoregressive distribution estimation. *The Journal of Machine Learning Research*, 17 (1):7184–7220, 2016.

Venkatraman, A., Hebert, M., and Bagnell, J. A. Improving multi-step prediction of learned time series models. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

Wu, J., Hu, C., Wang, Y., Hu, X., and Zhu, J. A hierarchical recurrent neural network for symbolic melody generation. *IEEE Transactions on Cybernetics*, 2019.

Yang, L.-C., Chou, S.-Y., and Yang, Y.-H. Midinet: A convolutional generative adversarial network for symbolic-domain music generation. *arXiv preprint arXiv:1703.10847*, 2017.