

Figure 1. The proposed 2-level audio autoencoder. A log-magnitude spectrogram is used as the encoder input, while the decoder outputs magnitude and phase spectrograms which are then inverted with iSTFT to the waveform domain. A discriminator evaluates the magnitude spectrogram of two adjacent excerpts passed through the autoencoder. This training process removes both phase errors (which would manifest after the iSTFT and STFT) and boundary artifacts.

tion retrieval is the Short-Time Fourier Transform (STFT) spectrogram: while the phase component of the spectrogram is usually discarded, in case of audio synthesis applications both magnitude and phase components are necessary to perform the inverse STFT (iSTFT) and obtain a waveform.

We design an audio autoencoder with the aim of minimizing inference and training time while maximizing the compression ratio allowing to reconstruct samples with reasonable accuracy. Our proposed autoencoder takes a log-magnitude STFT spectrogram as input, and outputs magnitude and phase spectrograms which can be inverted to a waveform. Parallel to our work, iSTFTNet [26] also proposes to improve the inference speed of the model by generating magnitude and phase of a STFT spectrogram: however, they only report experiments using spectrograms with very high temporal resolution and low frequency resolution, while our proposed autoencoder reconstructs spectrograms with low temporal resolution and high frequency resolution. This should result in an even higher inference speed for similarly-sized models. In practice, we separately train two stacked autoencoders; this allows a higher compression ratio with satisfactory reconstruction quality, especially for more complex music domains. Similarly to RAVE [7], we utilize a two-step training process:

3.1.1 First training phase

We first train the model to autoencode log-magnitude spectrograms, not producing phases for now. We use a L1 loss function for the reconstruction task:

$$\mathcal{L}_{(Enc,Dec),rec} = \mathbb{E}_{s \sim p(s)} \|Dec(Enc(s)) - s\|_1$$

where Enc and Dec are the encoder and decoder, and s is a log-magnitude spectrogram of a waveform w .

3.1.2 Second training phase

In the second phase, we freeze the encoder weights and have the decoder produce a phase spectrogram as well, such that we can reconstruct a waveform through an iSTFT. We add an adversarial objective to aid the modeling of both the magnitudes and phases, ensuring the wave-

form is of perceptually satisfactory quality. Since directly modeling phase spectrograms with deep learning models is known to be difficult [13, 18], we propose to model the phases indirectly, by encouraging waveforms whose magnitude spectrogram must appear realistic. Specifically, we compute a log-magnitude spectrogram \tilde{s} from the reconstructed waveform \tilde{w} :

$$\begin{aligned}\tilde{w} &= iSTFT(Dec(Enc(s))) \\ \tilde{s} &= \log(|STFT(\tilde{w})|^2 + \epsilon)\end{aligned}$$

The reconstructions \tilde{s} are fed to a discriminator D , using the hinge loss [27] to distinguish them from originals s :

$$\begin{aligned}\mathcal{L}_D &= -\mathbb{E}_{s \sim p(s)} [\min(0, -1 + D(s))] \\ &\quad - \mathbb{E}_{s \sim p(s)} [\min(0, -1 - D(\tilde{s}))]\end{aligned}$$

The decoder is trained to fool the discriminator:

$$\mathcal{L}_{Dec,adv} = -\mathbb{E}_{s \sim p(s)} D(\tilde{s})$$

Note that we can calculate spectrograms from the reconstructed waveforms with different hop size and window length than used for the spectrograms fed to the autoencoder. We leverage this by including the multi-scale spectral distance [7, 28] in the objective of the decoder:

$$\begin{aligned}\mathcal{L}_{Dec,ms} &= \mathbb{E}_{w \sim p(w)} \sum_{hop} \log(|STFT_{hop}(w)| \\ &\quad - |STFT_{hop}(\tilde{w})| |)_1\end{aligned}$$

where hop indicates a choice of hop_size and fft_size .

In total, we train the discriminator with \mathcal{L}_D , and the decoder with a linear combination of three losses:

$$\mathcal{L}_{Dec} = \mathcal{L}_{Dec,adv} + \lambda_{rec} \mathcal{L}_{Dec,rec} + \lambda_{ms} \mathcal{L}_{Dec,ms}$$

3.2 Latent Coordinate System

We use a GAN to model sequences of latent vectors produced by a trained audio encoder. In order to generate independent audio samples that can be seamlessly concatenated with each other along the temporal axis, we condition the generator with the latent coordinate system proposed by [22], originally introduced to generate landscape

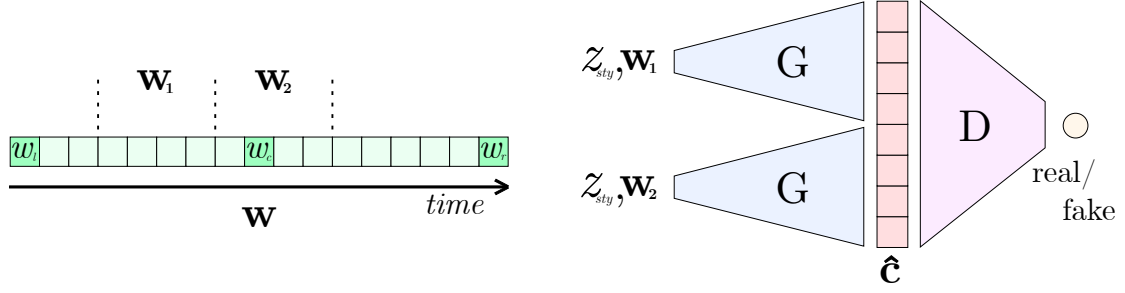


Figure 2. The proposed latent GAN training process. Two adjacent latent coordinate sequences are randomly cropped from the linear interpolation between three anchor vectors. They are then used as input to the generator, together with a shared style vector and a conditioning signal in the case of a conditional model. The discriminator takes as input the two concatenated generated sequences and a real sequence of latent vectors.

images of infinite width. Specifically, during training we sample three noise vectors w_l, w_c, w_r with dimension d that are used as anchor points (left, center, right anchors) to guide the generation process. With seq_len being the length of the sequence of latent vectors that is produced by the generator, we linearly interpolate the three anchor vectors to create a sequence of coordinate vectors of length equal to $4 \cdot seq_len + 1$:

$$\mathbf{w} = [w_l, \dots, (1-k)w_l + kw_c, \dots, w_c, \dots, w_r] \in \mathbb{R}^{4seq_len+1 \times d}$$

To generate sequences that are temporally coherent with each other, we follow [22]: we randomly crop a sequence \mathbf{w}_{12} of $2 \cdot seq_len$ coordinate vectors from \mathbf{w} , divide it into two sequences $\mathbf{w}_1, \mathbf{w}_2$ with length seq_len , generate two patches using each sequence as conditioning, concatenate the two patches along the time axis, and feed the resulting generated sample of length $2 \cdot seq_len$ to the discriminator. This process is illustrated in Figure 2. It allows the generator to align the sequence of latent coordinates with the generated sequence of latent vectors. Specifically, the discriminator forces the generator to learn that adjacent sequences of latent coordinates must result in adjacent sequences of latent vectors, which can be temporally concatenated resulting in a coherent final sample without artifacts at the boundaries of the generated patches.

Similarly to InfinityGAN [21], when generating adjacent sequences of latent vectors, we also condition both generations on a single random vector z_{sty} : during the learning process, this vector serves as conditioning for the global style of the generated samples. Specifically, while the latent coordinate vectors allow the generator to produce sequences of latent vectors that can be seamlessly concatenated along the temporal axis, the global style vector allows the final concatenated sequence of possibly infinite length to be stylistically coherent throughout. Without the global style vector, any temporal context available to the generator would completely change every $4 \cdot seq_len$ samples of a sequence, resulting in a final generated sample which continuously changes style through time.

Formalized, we have

$$\hat{\mathbf{c}} = \text{concat}[G(\mathbf{w}_1, z_{sty}), G(\mathbf{w}_2, z_{sty})],$$

where $\hat{\mathbf{c}}$ is a stylistically and temporally coherent sequence

of latent vectors of length $2 \cdot seq_len$, and G is the generator model.

At inference time, a latent coordinate sequence of the desired length is created. The coordinate sequence is prepared in the same way as during the training phase, by placing a latent anchor vector at positions that are multiples of $2 \cdot seq_len$, and by linearly interpolating these anchor vectors to calculate in-between vectors. A single random global style vector is also sampled. Each generation considers a seq_len crop and the global style vector as conditioning, and finally all generated latent vectors are concatenated together in the appropriate order. This process can be performed in a parallel manner, thus resulting in a fast generation on modern hardware.

4. IMPLEMENTATION DETAILS

4.1 Audio Autoencoder Architecture

We first train an audio autoencoder with a relatively low compression ratio, then train a second-level autoencoder that encodes the first-level latent vectors, as shown in Figure 1. During the training of the second-level autoencoder, we utilize the same training strategy and objective as explained in section 3.1, by propagating gradients through the frozen weights of the previously trained first-level decoder and adversarially discriminating between samples reconstructed by both decoders and samples reconstructed by only the first-level decoder. Both model architectures are fully convolutional, and we do not use any padding in both encoders, such that possible boundary artifacts in the encoded representations are avoided. We utilize 1d-convolutions considering the frequency bins as different channels for both encoders and decoders: this is usually not efficient regarding total number of model parameters when compared to using 2d-convolutions across the two spectrogram dimensions, but can result in a much faster inference time. We use 2d-convolutions for the discriminator, as inference time for this model is not a priority. We use \tanh as the activation for the bottleneck layer of both encoders. Regarding the multi-scale spectral distance loss, we use $hop_size \in [64, 128, 256, 512]$ and we always choose $fft_size = 4 \cdot hop_size$, while the discriminator takes as input log-magnitude spectrograms calculated with

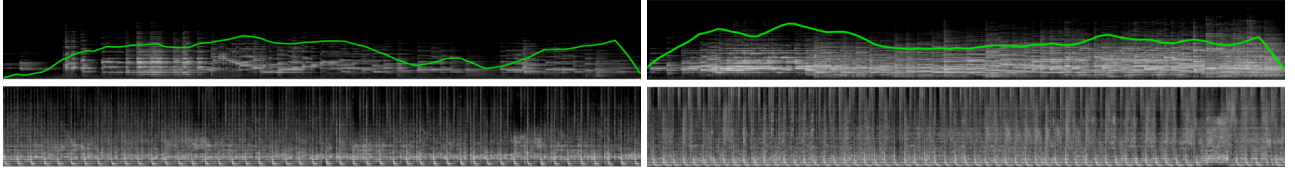


Figure 3. Log-melspectrograms of generated piano and techno samples from the conditional models. For the piano samples (top row), we indicate the corresponding note density conditioning with a green line. Note density signals were generated using a random walk algorithm. The tempo used as conditioning for the techno samples (bottom row) is 120 bpm and 160 bpm, respectively. Each sample is 23 seconds long. Visit marcoppasini.github.io/musika to listen to the examples.

$hop_size = 256$ and $fft_size = 6 \cdot hop_size$. As proposed by [29, 30], two consecutive reconstructed spectrograms are concatenated along the temporal dimension and fed to the discriminator, such that concatenated reconstructions do not suffer from boundary artifacts. During training, spectrograms calculated from 0.76 s of audio are used as input to both autoencoders. We use spectral normalization [31] on the weights of the discriminator. Regarding the training loss weights, we use $\lambda_{rec} = 1$ and $\lambda_{ms} = 4$. We choose Adam [32] as the optimizer with learning rate of 0.0001 and $\beta_1 = 0.5$, and train the first-level autoencoder for 1 million iterations with batch size of 32 for both training phases, and the second-level autoencoder for 400k iterations with batch size of 32 for both training phases.

4.2 Latent GAN Architecture

We choose to adapt the FastGAN [33] architecture to our specific task. The FastGAN architecture promises fast convergence with limited amounts of data. To achieve this, it proposes a Skip-Layer channel-wise Excitation (SLE) module in the generator, for more direct propagation of gradients, and proposes to strongly regularize the discriminator with an added self-supervised reconstruction objective. We adapt the proposed architectures to use 1d-convolutions instead of 2d-convolutions and we simplify the added reconstruction objective of the discriminator, by using a single lightweight decoder which reconstructs the whole input of the discriminator. Differently from FastGAN, we do not use Batch Normalization [34] in both the generator and discriminator, while we apply the variation of Adaptive Instance Normalization [35] (AdaIN) called Spatially Aligned AdaIN (SA-AdaIN), originally proposed in [22], after each convolutional layer in the generator. To generate stereo samples, the generator produces two latent vectors at each timestep, one for each audio channel, stacked on the channel axis. We use Cross Channel Mixing (CCM), first introduced in [36], to randomly mix channels of the stereo stacked latent vectors before being fed to the discriminator. In our experience, this technique helps reducing collapses during training. Both anchor and style vectors are sampled from a normal distribution with zero mean and unit variance, and have dimension d of 64. We use R1 gradient penalty [37] as regularization, and Adam with learning rate of 0.0001 and $\beta_1 = 0.5$ as the optimizer. We train for 1.5 mio iterations with a batch size of 32 for all experiments. Training takes 23 h on a RTX 2080 Ti GPU.

Model (Faster than real-time)	GPU	CPU
Musika Uncond. Piano	972x	40x
Musika Cond. Piano	921x	40x
UNAGAN [20] Piano	28x	11x
Musika Uncond. Techno	994x	39x
Musika Cond. Techno	917x	39x

Table 1. Comparison of generation speed between the different models. For the Musika models, we include both the generation of the latent vectors and the decoding step to the waveform domain. We use a RTX 2080 Ti and a Ryzen 3950x as the GPU and CPU, respectively. We report the average of 100 trials.

Model	FAD
Musika Uncond. Piano	1.641
Musika Cond. Piano Rand.	2.150
Musika Cond. Piano Const. 0.15	2.584
Musika Cond. Piano Const. 0.30	3.400
Musika Cond. Piano Const. 0.45	4.389
Musika Cond. Piano Const. 0.60	4.839
Musika Cond. Piano Const. 0.75	5.434
UNAGAN [20] Piano	11.183

Table 2. FAD evaluation for generated piano music. We evaluate conditional Musika models using different constant values of note density as conditioning. We notice that FAD increases with higher note density.

5. EXPERIMENTS

Considering the relatively low compression ratio of the first autoencoder and thus its need to only encode low-level audio features, we find it possible to train a single universal model which we can later use for different music domains. As training data, we choose to use songs released and made freely available by South by SouthWest¹ (SXSW) in occasion of their yearly conference. The current collection consists of 17k songs of various genres, and for this reason it represents a fitting choice for training our universal model. We use the LibriTTS corpus [38] as additional training data, to steer the universal model into accurately synthesizing human voices, which are notoriously hard to model. Even though LibriTTS only contains speech, in-

¹ <https://www.sxsw.com/festivals/music/>

cluding it improves reconstructions of singing voice. We resample audio to 22.05 kHz for all experiments. We use single channel audio to train the audio autoencoders, as the latent GAN is able to generate stereo samples by using latent representations of the two mono samples stacked in the channel dimension as training data. We use $r_{time}^1 = 256$ as the time compression ratio, which results in a sampling rate of the first-level latent representations of 190.22 Hz. Each of the encoded latent vectors has a dimension of 128.

5.1 Piano Music

We use the MAESTRO dataset [39], consisting of 200 hours of piano performances, to train a second-level autoencoder and a latent GAN. The final time compression ratio achieved by both autoencoders is $r_{time} = 4096$, which results in a sampling rate of the second-level latent representations of 11.89 Hz. The dimension of each latent vector is 32. We train both an unconditional and a conditional latent GAN. For both models, the generator outputs latent vectors with $seq_len = 64$, which results in about 12 s of audio after decoding. For the conditional model, we apply the CNN-based onset detector [40] of the madmom Python library [41] to all audio files in the dataset. We then use Gaussian Kernel Density Estimation (KDE) with bandwidth of 0.004 on the detected onsets to estimate a continuous note density signal for each sample. This signal is log-scaled between 0 and 1 and serves as a conditioning signal for the conditional (and thus controllable) GAN.

5.2 Techno Music

To evaluate the performance of the system on a more musically varied domain, we scrape 10,190 songs categorized with the “techno” genre from *jamendo.com* and use them as training data. Considering the wide diversity of sounds that are present in the dataset, we train the second-level autoencoder with the same SXS data used to train the first-level universal autoencoder. Comparing to what is achievable when training an autoencoder on a single and limited domain, such as piano music, a lower compression ratio is needed to reach a satisfactory reconstruction accuracy. However, this solution allows users to directly train a latent GAN on a new audio domain using the universal latent representations, without the need to train an autoencoder on the domain of interest. The final achieved time compression ratio is $r_{time} = 2048$, which results in a sampling rate of the second-level latent representations of 23.78 Hz. The dimension of each latent vector is 64. We train an unconditional and a conditional latent GAN model, both generating stereo latent vectors with $seq_len = 128$, resulting in about 12 s of decoded audio. We use the Tempo-CNN framework² [42] to estimate the global tempo of each song in the dataset. Tempo information is then used as conditioning for the conditional model.

² <https://github.com/hendriks73/tempo-cnn>

6. RESULTS

A comprehensive collection of generated audio samples is available on marcoppasini.github.io/musika. Since current quantitative evaluation metrics are not able to assess the overall compositional and musical quality of generated music, we strongly encourage the reader to listen to the provided samples while reading the paper.

We report the generation speed of the system trained on the MAESTRO and on the techno datasets in Table 1, on both GPU and CPU. We also use the Frechét Audio Distance [43] (FAD) metric to quantitatively evaluate the quality of the generated piano samples in Table 2. A UN-AGAN [20] model that was trained on the same dataset is used as comparison. While our system is capable of generating stereo audio, UN-AGAN can only produce single-channel audio. The unconditional model obtains the lowest FAD, while the conditional system results in higher FADs when using more intense note density values as conditioning. This is expected, since samples with low note density are more common than samples with high note density in the MAESTRO dataset. However, considering that audio is split in short 1 s samples to calculate embeddings, FAD is not designed to evaluate overall musical and compositional quality of samples, and to the best of our knowledge there are no available quantitative metrics to evaluate these characteristics. Piano and techno samples generated by the system seem to often demonstrate long-range coherence and successfully keep a fixed general music style through time. Both conditional models successfully generate samples that are coherent with the conditioning signal, as can be seen in Figure 3.

7. CONCLUSION

We proposed Musika, a non-autoregressive music generation system that generates raw-audio samples of arbitrary length much faster than real-time on a consumer CPU. An efficient hierarchical autoencoder allows to encode audio to a sequence of low-dimensional latent vectors, from which a waveform can be reconstructed. A GAN is then used to generate new sequences of latent vectors, using a latent coordinate system that allows for generation of samples of infinite length. A style conditioning vector is introduced to force the samples to be stylistically coherent through time. We successfully use the system to generate piano and techno music, and show that the generation process can be conditioned on note density and tempo information for piano and techno music, respectively. We finally show that the system achieves lower FAD than comparable systems on piano music generation while being faster. We release the source code and pretrained models, enabling users to generate samples of different music domains and test new conditioning signals with ease and using consumer hardware. We see our system as solving an important technical challenge – real-time music generation of sufficient quality, conditioned on user input – and hope it can serve as a basis for interactive real-world applications and for research into human-AI co-creation.

8. REFERENCES

- [1] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. W. Senior, and K. Kavukcuoglu, “WaveNet: A generative model for raw audio,” in *The 9th ISCA Speech Synthesis Workshop*, Sep. 2016, p. 125.
- [2] S. Mehri, K. Kumar, I. Gulrajani, R. Kumar, S. Jain, J. Sotelo, A. C. Courville, and Y. Bengio, “SampleRNN: An unconditional end-to-end neural audio generation model,” in *5th International Conference on Learning Representations (ICLR)*, Apr. 2017.
- [3] P. Dhariwal, H. Jun, C. Payne, J. W. Kim, A. Radford, and I. Sutskever, “Jukebox: A generative model for music,” *arXiv preprint arXiv:2005.00341*, 2020.
- [4] A. van den Oord, O. Vinyals, and K. Kavukcuoglu, “Neural discrete representation learning,” in *Advances in Neural Information Processing Systems 30*, Dec. 2017, pp. 6306–6315.
- [5] A. Razavi, A. van den Oord, and O. Vinyals, “Generating diverse high-fidelity images with VQ-VAE-2,” in *Advances in Neural Information Processing Systems 32*, Dec. 2019, pp. 14 837–14 847.
- [6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Advances in Neural Information Processing Systems 30*, Dec. 2017, pp. 5998–6008.
- [7] A. Caillon and P. Esling, “RAVE: A variational autoencoder for fast and high-quality neural audio synthesis,” *arXiv preprint arXiv:2111.05011*, 2021.
- [8] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” in *2nd International Conference on Learning Representations (ICLR)*, Apr. 2014.
- [9] K. Kumar, R. Kumar, T. de Boissiere, L. Gestein, W. Z. Teoh, J. Sotelo, A. de Brébisson, Y. Bengio, and A. C. Courville, “MelGAN: Generative adversarial networks for conditional waveform synthesis,” in *Advances in Neural Information Processing Systems 32*, Dec. 2019, pp. 14 881–14 892.
- [10] J. Kong, J. Kim, and J. Bae, “HiFi-GAN: Generative adversarial networks for efficient and high fidelity speech synthesis,” in *Advances in Neural Information Processing Systems 33*, Dec. 2020.
- [11] I. Elias, H. Zen, J. Shen, Y. Zhang, Y. Jia, R. J. Skerry-Ryan, and Y. Wu, “Parallel tacotron 2: A non-autoregressive neural TTS model with differentiable duration modeling,” in *22nd Annual Conference of the International Speech Communication Association (INTERSPEECH)*, Aug. 2021, pp. 141–145.
- [12] C. Donahue, J. J. McAuley, and M. S. Puckette, “Adversarial audio synthesis,” in *7th International Conference on Learning Representations (ICLR)*, May 2019.
- [13] J. H. Engel, K. K. Agrawal, S. Chen, I. Gulrajani, C. Donahue, and A. Roberts, “GANSynth: Adversarial neural audio synthesis,” in *7th International Conference on Learning Representations (ICLR)*, May 2019.
- [14] J. Nistal, S. Lattner, and G. Richard, “DRUMGAN: synthesis of drum sounds with timbral feature conditioning using generative adversarial networks,” in *Proceedings of the 21th International Society for Music Information Retrieval Conference (ISMIR)*, Oct. 2020, pp. 590–597.
- [15] K. v. d. Broek, “Mp3net: coherent, minute-long music generation from raw audio with a simple convolutional GAN,” *arXiv preprint arXiv:2101.04785*, 2021.
- [16] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio, “Generative adversarial nets,” in *Advances in Neural Information Processing Systems 27*, Dec. 2014, pp. 2672–2680.
- [17] J. H. Engel, C. Resnick, A. Roberts, S. Dieleman, M. Norouzi, D. Eck, and K. Simonyan, “Neural audio synthesis of musical notes with WaveNet autoencoders,” in *Proceedings of the 34th International Conference on Machine Learning (ICML)*, ser. Proceedings of Machine Learning Research, vol. 70, Aug. 2017, pp. 1068–1077.
- [18] J. Nistal, S. Lattner, and G. Richard, “Comparing representations for audio synthesis using generative adversarial networks,” in *28th European Signal Processing Conference (EUSIPCO)*. IEEE, Jan. 2020, pp. 161–165.
- [19] T. Karras, T. Aila, S. Laine, and J. Lehtinen, “Progressive growing of GANs for improved quality, stability, and variation,” in *6th International Conference on Learning Representations (ICLR)*, Apr. 2018.
- [20] J. Liu, Y. Chen, Y. Yeh, and Y. Yang, “Unconditional audio generation with generative adversarial networks and cycle regularization,” in *21st Annual Conference of the International Speech Communication Association (INTERSPEECH)*, Oct. 2020, pp. 1997–2001.
- [21] C. H. Lin, Y.-C. Cheng, H.-Y. Lee, S. Tulyakov, and M.-H. Yang, “InfinityGAN: Towards infinite-pixel image synthesis,” in *10th International Conference on Learning Representations (ICLR)*, Apr. 2022.
- [22] I. Skorokhodov, G. Sotnikov, and M. Elhoseiny, “Aligning latent and image spaces to connect the unconnectable,” in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, Oct. 2021, pp. 14 124–14 133.
- [23] P. Esser, R. Rombach, and B. Ommer, “Taming transformers for high-resolution image synthesis,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Computer Vision Foundation / IEEE, Jun. 2021, pp. 12 873–12 883.

- [24] O. Prykhodko, S. Johansson, P. Kotsias, J. Arús-Pous, E. J. Bjerrum, O. Engkvist, and H. Chen, “A de novo molecular generation method using latent vector based generative adversarial network,” *J. Cheminformatics*, vol. 11, no. 1, p. 74, 2019.
- [25] M. J. Kusner and J. M. Hernández-Lobato, “GANs for sequences of discrete elements with the Gumbel-softmax distribution,” *arXiv preprint arXiv:1611.04051*, 2016.
- [26] T. Kaneko, K. Tanaka, H. Kameoka, and S. Seki, “iSTFTNET: fast and lightweight mel-spectrogram vocoder incorporating inverse short-time fourier transform,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, May 2022, pp. 6207–6211.
- [27] J. H. Lim and J. C. Ye, “Geometric GAN,” *arXiv preprint arXiv:1705.02894*, 2017.
- [28] J. H. Engel, L. Hantrakul, C. Gu, and A. Roberts, “DDSP: differentiable digital signal processing,” in *8th International Conference on Learning Representations (ICLR)*, Apr. 2020.
- [29] C. H. Lin, C. Chang, Y. Chen, D. Juan, W. Wei, and H. Chen, “COCO-GAN: generation by parts via conditional coordinating,” in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, Oct. 2019, pp. 4511–4520.
- [30] M. Pasini, “MelGAN-VC: Voice conversion and audio style transfer on arbitrarily long samples using spectrograms,” *arXiv preprint arXiv:1910.03713*, 2019.
- [31] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, “Spectral normalization for generative adversarial networks,” in *6th International Conference on Learning Representations (ICLR)*, Apr. 2018.
- [32] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *3rd International Conference on Learning Representations (ICLR)*, May 2015.
- [33] B. Liu, Y. Zhu, K. Song, and A. Elgammal, “Towards faster and stabilized GAN training for high-fidelity few-shot image synthesis,” in *9th International Conference on Learning Representations (ICLR)*, May 2021.
- [34] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, ser. JMLR Workshop and Conference Proceedings, vol. 37, Jul. 2015, pp. 448–456.
- [35] X. Huang and S. J. Belongie, “Arbitrary style transfer in real-time with adaptive instance normalization,” in *IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017, pp. 1510–1519.
- [36] A. Sauer, K. Chitta, J. Müller, and A. Geiger, “Projected GANs converge faster,” in *Advances in Neural Information Processing Systems 34*, Dec. 2021, pp. 17 480–17 492.
- [37] L. M. Mescheder, A. Geiger, and S. Nowozin, “Which training methods for GANs do actually converge?” in *Proceedings of the 35th International Conference on Machine Learning (ICML)*, ser. Proceedings of Machine Learning Research, vol. 80, Jul. 2018, pp. 3478–3487.
- [38] H. Zen, V. Dang, R. Clark, Y. Zhang, R. J. Weiss, Y. Jia, Z. Chen, and Y. Wu, “LibriTTS: A corpus derived from LibriSpeech for text-to-speech,” in *20th Annual Conference of the International Speech Communication Association (INTERSPEECH)*, Sep. 2019, pp. 1526–1530.
- [39] C. Hawthorne, A. Stasyuk, A. Roberts, I. Simon, C. A. Huang, S. Dieleman, E. Elsen, J. H. Engel, and D. Eck, “Enabling factorized piano music modeling and generation with the MAESTRO dataset,” in *7th International Conference on Learning Representations (ICLR)*, May 2019.
- [40] J. Schlüter and S. Böck, “Improved musical onset detection with convolutional neural networks,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2014, pp. 6979–6983.
- [41] S. Böck, F. Korzeniowski, J. Schlüter, F. Krebs, and G. Widmer, “madmom: A new python audio and music signal processing library,” in *Proceedings of the 2016 ACM Conference on Multimedia Conference (MM)*, Oct. 2016, pp. 1174–1178.
- [42] H. Schreiber and M. Müller, “A single-step approach to musical tempo estimation using a convolutional neural network,” in *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, Sep. 2018, pp. 98–105.
- [43] K. Kilgour, M. Zuluaga, D. Roblek, and M. Sharifi, “Fréchet audio distance: A reference-free metric for evaluating music enhancement algorithms,” in *20th Annual Conference of the International Speech Communication Association (INTERSPEECH)*, Sep. 2019, pp. 2350–2354.

SYMPHONY GENERATION WITH PERMUTATION INVARIANT LANGUAGE MODEL

Jiafeng Liu^{1*}

Yuanliang Dong^{1*}

Zehua Cheng²

Xinran Zhang¹

Xiaobing Li¹

Feng Yu¹

Maosong Sun^{1,3†}

¹ Department of Music AI and Music Information Technology, Central Conservatory of Music

² Department of Computer Science, University of Oxford

³ Department of Computer Science and Technology, Tsinghua University

{jiafeng.liu, gunterdong}@mail.ccom.edu.cn

ABSTRACT

In this work, we propose a permutation invariant language model, SymphonyNet, as a solution for symbolic symphony music generation. We propose a novel Multi-track Multi-instrument Repeatable (MMR) representation for symphonic music and model the music sequence using a Transformer-based auto-regressive language model with specific 3-D positional embedding. To overcome length overflow when modeling extra-long symphony tokens, we also propose a modified Byte Pair Encoding algorithm (Music BPE) for music tokens and introduce a novel linear transformer decoder architecture as a backbone. Meanwhile, we train the decoder to learn automatic orchestration as a joint task by masking instrument information from the input. We also introduce a large-scale symbolic symphony dataset for the advance of symphony generation research. Empirical results show that the proposed approach can generate coherent, novel, complex and harmonious symphony as a pioneer solution for multi-track multi-instrument symbolic music generation.

1. INTRODUCTION

Symphony is one of the most complex and brilliant musical composition forms in human history, where many instruments are intertwined to express rich human emotions. The past decade has seen the rapid development and tremendous success of the symbolic music generation in both research and industrial field [1–3]. Most current works follow conventional text modeling and generation method by applying language model to sequences of symbolic musical events [4–6]. However, symphony modeling

and generation still constitutes in itself a considerable challenge since symphony music sequences differ from text sequences in various aspects.

Natural language could be modeled as a purely linear sequence constructed strictly by a sequential order of words. Symphony scores, on the other hand, are usually viewed as two-dimensional symbolic sequences in which many notes can be played concurrently. Notes in a symphony score are **semi-permutation invariant**. More specifically, as shown in Fig. 1, the blue box indicates the musical instrument tracks, and the corresponding staves on the right side are permutation invariant. Similarly, the notes inside the red box are also permutation invariant. In contrast, notes in the upper yellow box are permutation variant since each note is played sequentially. Changes in the order of notes will impair the music itself. The yellow box at the bottom is a more complicated situation: a permutation variant note sequence in general containing permutation invariant notes. Simply flattening the score into a 1-D text-like sequence may damage the local structure of music [7]. To address this problem, we propose the Multi-track Multi-instrument Repeatable (MMR) representation with particular 3-D positional embedding in Section 3 which fully considers the properties of semi-permutation invariance in symbolic music scores.

Moreover, when comparing music scores with text, conventionally notes could be considered as characters, while intervals or chords are comparable to words. Modeling musical events at note level is a common practice [5,6,8,9]. However, this may be confronted with similar problems in char-level text generation, such as extremely long sequences and less meaningful individual tokens. Word-level tokenization suffers from large vocabulary size and out of vocabulary (OOV) problems. Byte Pair Encoding (BPE) [10, 11] subword tokenization is a tradeoff between word-level and character-level tokenization. Inspired by BPE, we propose the Music BPE algorithm in Section 4, which could automatically aggregate notes to intervals and chords as subwords without a pre-defined vocabulary and construct music sequences with richer semantics.

Generating symphony music with proper instruments for different tracks is another challenging task. Recent

* The authors contributed equally to this work.

† Corresponding author. Email: sms@tsinghua.edu.cn





Figure 1: A simple example of Multi Instruments & Multi Tracks & Repeat Instruments symphony score.

work like Arranger [12] focuses on instrumentation by learning to separate parts from the mixture in symbolic multi-track music. However, it does not incorporate music generation task. In this paper, we present a unique linear transformer decoder architecture for instrument classification with joint-task training, which allows the model to learn auto-orchestration rather than relying on instrument information as an pre-defined input source. [5, 13–15].

The contributions of this paper are presented as below:

- We propose a novel Multi-track Multi-instrument Repeatable (MMR) representation for symphony music, including particular 3-D positional embedding designed to address the semi-permutation invariant challenge in symphony generation. Our method is also compatible with all existing symbolic music ensembles, including but not limited to piano solo, quartet and pop band music.
- We propose a novel algorithm, Music BPE, to model the symbolic music at subword-level. Furthermore, we found that our Music BPE algorithm could aggregate notes to intervals and chords, which are consistent with common chords summarized by human musicians.
- We introduce SymphonyNet, a novel music generation model with joint-task training for instrument classification based on our proposed MMR representation and Music BPE. The model can learn the proper orchestration according to the distribution of the notes.
- We collect a symphony MIDI dataset, consisting of 46,359 high-quality MIDI files with multiple instruments and tracks to advance researches on symphony generation with deep learning.

2. RELATED WORK

We organize some existing works in Table 1 in terms of five aspects of symbolic music modeling: time unit, representation method, backbone model, music type and the

ability to model music with repeat instruments. Generation works are presented above and understanding works are presented below. Pianoroll, MIDI event timeshift, and Beat-based onset and duration are the mainstream time units in music generation and understanding tasks. However, Pianoroll divides music into fixed-length grids, and MIDI format provides overprecise timeshift events, both suffering from sparsity problems, which raises another handicap for applying deep learning models in this multi-track generation. Pop Music Transformer [8] is the first attempt to introduced the beat-based REMI representation in music generation. It supports variable-length duration of notes, which is more musically inspired. Compound Word [6], derived from REMI representation, classifies the sequence of REMI into note-related or metric-related events, which are then aggregated, greatly decreasing the sequence length.. This has engendered a new trend of beat-based symbolic music generation.

Language models are now prevalent in natural language processing tasks [18]. However, applying language models to the creation of multi-track music remains challenging. MuMIDI [5] and OctupleMIDI [9] models multiple attributes of one note in one sequence step and also incorporates instrument tokens for multi-track representation. However, if one musical piece contains more than one track for the same instrument, their representation could not distinguish them in different tracks. MMM [15] introduced a MIDI-event-like representation, creating a time-ordered sequence of musical events for each track and concatenating several tracks into a single sequence. However, MMM adopts time-delta tokens and fixed positional encoding which weakens the note-level correlation and structure between tracks. MuseBert [7] proposes a permutation invariant bert-like language model with generalized relative position encoding (RPE) which, however, is not compatible with multi-track music generation.

Though various symbolic music representation strategies have been proposed, few are compatible with multi-track music with repeatable instruments or tracks, such as the symphony. Furthermore, permutation invariance of music, as is discussed in Section 1, has scarcely been considered. To our knowledge, this work proposes the first representation and tokenization method to encode music with multiple repeatable instruments and multiple repeatable tracks and designs a universal and effective strategy for generating symphony music with permutation invariant language model.

3. MULTI-TRACK MULTI-INSTRUMENT REPEATABLE REPRESENTATION

To further analyze the symphony generation task, it is crucial to understand the difference between the symphony format and other genres of music.

- **Single Instrument in Single Track.** No more than one note is played at any timestep by one instrument. Also called monophonic music. e.g., flute.
- **Multi Instruments & Each in Single Track.** Only one