Transformer-GAN: Symbolic music generation using a learned loss

Aashiq Muhamed[*†], **Liang Li**[*†], **Xingjian Shi**[†], **Suri Yaddanapudi**[†], **Wayne Chi**[†], **Dylan Jackson**[†], **Rahul Suresh**[†], **Zachary Lipton**[‡], **Alexander J. Smola**[†]
[†]Amazon Web Services, [‡] Carnegie Mellon University

Abstract

The conventional approach to symbolic music generation uses the Transformer, an autoregressive model that is commonly trained by minimizing the negative log-likelihood (NLL) of the observed sequence. The quality of samples from these models tends to degrade significantly for long sequences, a phenomenon attributed to exposure bias. However, we are able to detect these failures with classifiers trained to distinguish between real and sampled sequences, an observation that motivates our exploration of adversarial losses to complement the NLL objective. We use a pre-trained SpanBERT model for the discriminator of the GAN, which in our experiments helped with training stability. We demonstrate via human evaluations and a new discriminative metric that music generated by our approach outperforms a baseline trained with likelihood maximization, the state-of-the-art Music Transformer, and other GANs used for sequence generation. 57% of people prefer music generated via our approach while 43% prefer Music Transformer.

1 Introduction

Recent advancements in natural language processing, especially the attention mechanism and the Transformer architecture [1], have helped push the state of the art in symbolic music generation [2–4]. These approaches represent a piece of music as a sequence of time-ordered events. A Transformer-based language model is then trained on top of these event sequences by maximizing likelihood and music is generated by sampling from this language model. Despite recent improvements, these approaches exhibit crucial failure modes which we argue arise from the training objective. As stated by the authors, Music Transformer [2] occasionally forgets to switch off notes and loses coherence beyond a few target lengths. Sometimes it produces highly repetitive songs, sections that are almost empty, and discordant jumps between contrasting phrases and motifs. Consequently, music generated by such models can be distinguished from real music by a simple classifier. This suggests that a distribution distance, such as the adversarial objective of a GAN [5] should improve the fidelity of the generative model. Inspired by recent works on evaluation metrics in text generation [6–8] which suggest that BERT-based scores [9] are well correlated with human rankings, we propose to use a pretrained BERT as the discriminator. Unfortunately, incorporating GAN losses for discrete sequences can be difficult as differentiating through the sampling process is challenging. As such, many models [10, 11] are limited to 20-40 token-length sentences, in contrast to the more than 1000 tokens required for minutes-long musical compositions. In this paper we propose an adversarial framework that uses the Transformer-XL as generator and SpanBERT-style pretrained BERT as discriminator. We highlight a few additional tricks that helped us train on *long* sequences.

2 Transformer-GAN

Given a music sequence $\mathbf{x}^r$ from the the unknown data distribution $p_r(\mathbf{x})$ and an autoregressive model $p_\theta(\mathbf{x})$, maximum likelihood seeks to find a θ that minimizes $L_{\text{mle}} = -\mathbb{E}_{\mathbf{x}^r \sim p_r}\left[\log p_\theta\left(\mathbf{x}^r\right)\right]$. Despite

[*]Equal contribution; muhaaash@amazon.com, mzliang@amazon.com

4th Workshop on Machine Learning for Creativity and Design at NeurIPS 2020, Vancouver, Canada.

its attractive *theoretical* properties, maximum likelihood training suffers from many limitations, e.g. whenever the model is misspecified. This is illustrated by [12] in image-to-image translation, where no explicit loss function is available. Furthermore, teacher forcing introduces exposure bias [13, 14]—a distributional shift between training sequences used for learning and model data required for generation. This amplifies any errors in the estimate, sometimes creating unrealistic, repetitive outputs. We address this problem by incorporating an adversarial loss into our objective. During training, we alternate updates between the generator and discriminator objectives:

$$L_{\mathrm{G}} = L_{\mathrm{mle}}[G_\theta] + \lambda L_{\mathrm{gen}}[G_\theta] \tag{1}$$

$$L_{\mathrm{D}} = L_{\mathrm{disc}}[D_\phi] + \gamma L_{\mathrm{reg}}[D_\phi] \tag{2}$$

Here $\lambda, \gamma > 0$ are hyperparameters. We investigate several choices for $L_{\mathrm{gen}}, L_{\mathrm{disc}}$ and L_{reg}: WGAN loss with gradient penalty [15], RSGAN loss [16], and PPO-GAN loss [17]. We use the Transformer-XL [18] as generator G_θ, which introduces the notion of recurrence into the deep self-attention network. In prior work [11] , CNNs [19] have proven to be useful discriminators for text generation. In this work however, we use BERT as the discriminator D_ϕ to extract sequence embeddings followed by a pooling and linear layer. We speculate that this would help the discriminator provide informative gradients to the generator and stabilize the training process. We conjecture that freezing earlier layers (closer to input) of the pretrained discriminator can be viewed as transfer learning, where we transfer music representations useful for generation from a different dataset. Unlike FreezeD [20], where the discriminator is transferred between trained GANs on different datasets, we pretrain our discriminator on the same dataset using a SpanBERT style self-supervised loss [21] . Freezing the discriminator also reduces the number of trainable parameters and training memory requirements. We leverage the Gumbel-Softmax [22] trick to obtain a differentiable approximation of the sampling process and a variant of the Truncated Backpropagation Through Time (TBPTT) [23] algorithm for gradient propagation on long sequences. The latter keeps memory requirements at bay. We provide details on these training tricks in the Appendix.

3 Experiments

We used the MAESTRO MIDI v1 dataset [24] and the same data augmentation and event representation as in Music Transformer. The Transformer-GAN is trained with the RSGAN, RSGAN-GPen, WGAN-Gpen and PPO-Gpen losses where Gpen denotes gradient penalty. We conducted a survey where participants were presented with samples from the different models conditioned on the same priming melody. We received 448 ratings for quality (scored 0 to 5) and 672 pairwise comparisons (compared for coherence and consistency) from 15 subjects. The results in Table 1 and the survey led to the following findings: (i) We achieve comparable results overall between Music Transformer (Random) and a baseline Transformer-XL (TopK). (ii) Transformer-GAN with WGAN-GPen performs especially the best at polyphonic generation and is seen to be closest to the training dataset on multiple quantitative metrics [25, 26]. (iii) A Kruskal Wallis test on the ratings yields $\mathscr{X}^2(2) = 3.272, p = 0.031$, and Transformer-GAN (Random) outperforms the Transformer-XL (Random) with 62 wins and 50 losses. (iv) Transformer-GAN (Random) outperforms all other models sampled with either Random or TopK sampling. (v) Training with gradient penalty improves quantitative metrics overall. In the appendix, we include qualitative feedback from professional musicians on some of the examples generated. All of them observed that the Transformer-GAN helped retain long term coherence and the samples made fewer mistakes in general.

	Discriminator	NLL ↓	Sampling	CA ↓	PLL ∼	PCU ∼	NPU ∼	EBR ∼	ISR ∼	PRS ∼	TUP ∼	PR ∼	APS ∼	IOI ∼
Training Set	–	–	–	–	2.0203	7.810	65.848	0.985	0.586	0.399	65.28	67.34	11.531	0.133
Music Transformer	–	1.79	Random	0.8443	2.5666	7.214	**54.552**	0.997	0.599	0.465	54.95	62.035	**11.619**	0.113
Transformer-XL	–	**1.74**	Top32	0.8377	2.1531	7.045	52.965	0.981	0.572	0.284	52.95	60.39	11.117	0.107
WGAN-GPen	CNN	1.75	Random	0.8401	2.3087	6.95	51.764	0.98	0.608	**0.327**	52.28	59.37	10.832	**0.119**
WGAN-GPen	Pretrained BERT	1.75	Random	**0.8179**	**2.1020**	7.19	52.915	0.975	**0.585**	0.277	**55.56**	**63.23**	11.935	0.145
PPO-GPen	Pretrained BERT	1.75	Random	0.8213	2.3549	6.932	52.2	**0.983**	0.598	0.298	52.31	59.245	10.808	0.163
RSGAN-GPen	Pretrained BERT	1.75	Random	0.8307	2.2766	**7.285**	53.325	0.971	0.578	0.304	54.11	62.83	11.461	0.136
RSGAN	Pretrained BERT	1.75	Random	0.8615	2.1084	6.56	47.245	0.993	0.607	0.192	48.165	55.62	11.256	0.082

Table 1: Quantitative music metrics: NLL (Negative likelihood); PCU (Unique pitch classes); NPU (Unique pitches); EBR (Empty beats/Total beats); ISR (Nonzero entries in C major scale/ to the total nonzero entries); PRS (Time steps where the no. of pitches ≥ 4 / Total time steps); TUP (Different pitches within a sample); PR (Avg. difference of the highest and lowest pitch in semitones); APS (Avg. semitone interval between two consecutive pitches); IOI (Time between two consecutive notes); CA (SpanBERT classifier accuracy distinguishing real and generated data); PLL (Pseudo-log-likelihood score). Bolded values are better. Metrics marked with $\sim$ are better when closer to the dataset.

References

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[2] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, I. Simon, C. Hawthorne, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck, "Music transformer," *arXiv preprint arXiv:1809.04281*, 2018.

[3] C. Payne, "MuseNet," Apr 2019. [Online]. Available: openai.com/blog/musenet

[4] C. Donahue, H. H. Mao, Y. E. Li, G. W. Cottrell, and J. McAuley, "Lakhnes: Improving multi-instrumental music generation with cross-domain pre-training," *arXiv preprint arXiv:1907.04868*, 2019.

[5] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[6] J. Salazar, D. Liang, T. Q. Nguyen, and K. Kirchhoff, "Masked language model scoring," *arXiv preprint arXiv:1910.14659*, 2019.

[7] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, "BERTScore: Evaluating text generation with BERT," *arXiv preprint arXiv:1904.09675*, 2019.

[8] E. Montahaei, D. Alihosseini, and M. S. Baghshah, "Jointly measuring diversity and quality in text generation models," *arXiv preprint arXiv:1904.03971*, 2019.

[9] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[10] C. de Masson d'Autume, S. Mohamed, M. Rosca, and J. Rae, "Training language GANs from scratch," in *Advances in Neural Information Processing Systems*, 2019, pp. 4300–4311.

[11] W. Nie, N. Narodytska, and A. Patel, "RelGAN: Relational generative adversarial networks for text generation," in *ICLR*, 2019.

[12] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1125–1134.

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

[14] A. Holtzman, J. Buys, M. Forbes, and Y. Choi, "The curious case of neural text degeneration," *arXiv preprint arXiv:1904.09751*, 2019.

[15] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," in *Advances in neural information processing systems*, 2017, pp. 5767–5777.

[16] A. Jolicoeur-Martineau, "The relativistic discriminator: a key element missing from standard gan," *arXiv preprint arXiv:1807.00734*, 2018.

[17] Y. Wu, P. Zhou, A. G. Wilson, E. P. Xing, and Z. Hu, "Improving gan training with probability ratio clipping and sample reweighting," *arXiv preprint arXiv:2006.06900*, 2020.

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

[19] Y. Kim, "Convolutional neural networks for sentence classification," *arXiv preprint arXiv:1408.5882*, 2014.

[20] S. Mo, M. Cho, and J. Shin, "Freeze discriminator: A simple baseline for fine-tuning gans," *arXiv preprint arXiv:2002.10964*, 2020.

[21] M. Joshi, D. Chen, Y. Liu, D. S. Weld, L. Zettlemoyer, and O. Levy, "Spanbert: Improving pre-training by representing and predicting spans," *Transactions of the Association for Computational Linguistics*, vol. 8, pp. 64–77, 2020.

[22] 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.

[23] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems (NeurIPS)*, 2014.

[24] C. Hawthorne, A. Stasyuk, A. Roberts, I. Simon, C.-Z. A. Huang, S. Dieleman, E. Elsen, J. Engel, and D. Eck, "Enabling factorized piano music modeling and generation with the MAESTRO dataset," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=r1lYRjC9F7

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

[26] L.-C. Yang and A. Lerch, "On the evaluation of generative models in music," *Neural Computing and Applications*, vol. 32, no. 9, pp. 4773–4784, 2020.

[27] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," *arXiv preprint arXiv:1611.01144*, 2016.

[28] C. Tallec and Y. Ollivier, "Unbiasing truncated backpropagation through time," *arXiv preprint arXiv:1705.08209*, 2017.

[29] L. Yu, W. Zhang, J. Wang, and Y. Yu, "SeqGAN: Sequence generative adversarial nets with policy gradient," in *Thirty-first AAAI conference on artificial intelligence*, 2017.

[30] N. Zhang, "Learning adversarial transformer for symbolic music generation," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[31] X. Chen, Y. Li, P. Jin, J. Zhang, X. Dai, J. Chen, and G. Song, "Adversarial sub-sequence for text generation," *arXiv preprint arXiv:1905.12835*, 2019.

Appendix

A Training tricks for the Transformer-GAN

In this section, we present details of the Transformer-GAN model architecture and adversarial training tricks.

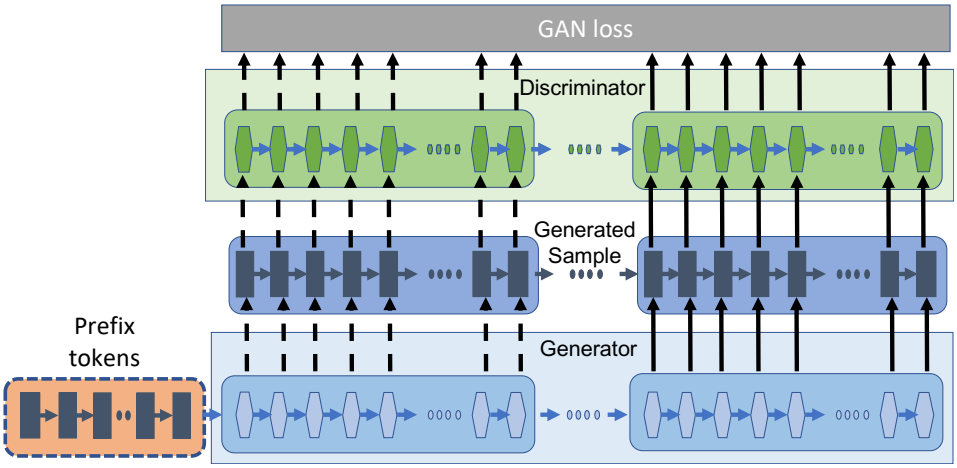

Figure A.1: Music generation with the Transformer-GAN network. Arrows are in the direction of the forward pass; solid lines denote gradients are backpropagated in training; dashed lines denote no gradient propagation.

The Transformer-GAN consists of the Transformer-XL as generator and a SpanBERT style pretrained BERT as discriminator as shown Fig. A.1. The adversarial training of this Transformer-GAN involves generating discrete samples $\mathbf{x}^f \sim p_\theta(\mathbf{x})$ autoregressively from the generator to feed into the discriminator. However, several issues arise during the generation and training on these discrete samples, e.g., the non-differentiable sampling step, the repetition and high variance of generated samples, the high memory and compute complexity during backpropagation, and the instability during GAN training. Here, we highlight a few critical tricks to address these issues.

A.1 Gumbel-Softmax

The discrete samples are generated sequentially. To generate the next token x^f_{t+1}, we sample from the multinomial distribution $\text{softmax}(o_t)$ on the vocabulary set V which can be formulated as

$$x^f_{t+1} \sim \text{softmax}(o_t), \tag{3}$$

where $o_t \in \mathbb{R}^V$ denotes the output logits from the generator obtained by attending over the past tokens $\{x^f_1, .., x^f_t\}$. However, this sampling process is not differentiable, as the derivative of a step function is 0 or undefined everywhere.

To deal with this, we reparameterize the sampling operation using the Gumbel-Max trick as

$$x^f_{t+1} = \arg\max_{1 \le i \le V} (o^{(i)}_t + g^{(i)}_t), \tag{4}$$

where $o^{(i)}_t$ denotes the i-th entry of o_t, $g^{(i)}_t$ is the i-th entry of g_t, which follows the element-wise i.i.d. standard Gumbel distribution. As this arg max is still not differentiable, we approximate arg max in the backward pass using the Gumbel-softmax trick, where the Gumbel-softmax is both continuous and differentiable as shown in [27]. Therefore, in the backward pass, (4) becomes

$$\text{softmax}(\beta(o_t + g_t)), \tag{5}$$

where $\beta > 0$ is a tunable parameter called inverse temperature. At last, $\{x^f_1, .., x^f_n\}$ forms the sequence as $\mathbf{x}^f$, which will be fed into the discriminator.

A.2 Exponential inverse-temperature

When using a fixed inverse temperature β in (5) to train the GAN, we noticed that the generator has a tendency to suffer from mode collapse, generating many repeated tokens. We found that this can be mitigated by using a large β_{max} and applying the exponential policy $\beta_n = \beta_{max}^{n/N}$ to increase β over iterations, where N is the maximum number of training iterations and n denotes the current iteration. [11] suggests that the exponential inverse-temperature decay policy can help balance exploration and exploitation during generator sampling. A larger β encourages more exploration for better sample diversity, while a smaller β encourages more exploitation for better sample quality.

A.3 Conditional generation

Another issue we notice is that (5) can lead to a large variance in gradient estimation due to the randomness in sampling. In order to reduce this variance, we reduce the distance between the real $\mathbf{x}^r \sim p_r(\mathbf{x})$ and fake $\mathbf{x}^f \sim p_\theta(\mathbf{x})$ samples by applying conditional sampling where the real and fake samples share a common priming sequence. To generate the fake samples, we condition the generator on the shared priming sequence $[x_1^r, ..., x_c^r]$ and sample the remaining $[x_{c+1}^f, ..., x_n^f]$ autoregressively.

A.4 Truncated Backpropagation Through Time (TBPTT)

The generator sampling step in (5) is sequential and therefore the backward pass in the generator resembles the backpropagation through time (BPTT) algorithm [28]. However, the generated sequences that are sequentially sampled can be very long, potentially ≥ 2000 tokens. Standard BPTT on those long sequences is both compute-intensive and memory-intensive.

To resolve this, we truncate our generated sequences into segments and feed the segments into the discriminator. Then, we do backpropagation on each segment and accumulate the gradients. The truncation improves memory efficiency as it avoids holding all forward computation graphs during sampling. The length of the subsequence is also well suited to our BERT since it is trained to accept a smaller fixed length sequence. TBPTT can be formally expressed in terms of parameters k_1 and k_2. k_1 is the number of forward-pass timesteps between updates and k_2 is the number of timesteps to which to apply BPTT. In this paper, we use $k_1 = k_2$. Breaking the generated sequence into subsequences for gradient propagation resembles the subsequence reward training [29–31] in Reinforcement Learning based GANs.

A.5 Gradient penalty

During training, we notice that the discriminator can be easily trained to optimality before the generator parameter update. In addition, exploding or vanishing gradients was a recurrent problem. We discovered that in order to stabilize the GAN training, it was necessary to add the gradient penalty regularizer [15].

Our hypothesis and findings on the importance of discriminator regularization align with prior work [15] on image GANs. We find that discriminator regularization in the form of layer normalization, dropout and L2 weight decay offered a less significant performance boost than the gradient penalty regularizer.

B Qualitative study on generated samples

We gave a small set of clips from the baseline Transformer-XL and Transformer-GAN to a few musicians and composers and simply asked for any initial reactions/comments. Here is a small, representative subset of the comments we received.

- "The Transformer XL compositions sound somewhat virtuosic, there seems to often be one hand that's moving quite quickly. They are quite pleasant to listen to but end up at quite different places compared to the beginning. PPO-GAN's music is *significantly more polyphonic*, and also develops more slowly and consistently. It does interesting transitions as well, and *maintains the new style for quite some time.*"

- "Overall, the quality of the performance (of Transformer-GAN samples) is *excellent, showcasing varied dynamics and proper phrasing*. The tempo is not rigid but expressive. The (sustaining) pedaling seems to be a little muddled at times, but not to a point where it hampers the delivery."
- "For the most part, harmonic choices (in the Transformer-GAN samples) are sensible locally, and chord progressions are constructed on the appropriate scale. Aside from a few exceptions, *the composition style remains consistent throughout, without abrupt or unreasonable shifts.* In half of the samples, a recurring motif can be recognized, at least for about 10 measures initially, and in some cases, developing in an interesting fashion. However, none of the samples demonstrate a global structure (exposition, development, recapitulation), which is prevalent in classical compositions. Occasionally a sequence of notes or trills are repeated for an excessively long time, *but as a whole the melodic line feels natural and pleasing to the ear.*"

C Samples

We include both conditional and unconditional samples from the baseline Transformer-XL, PPO-Gpen and WGAN-Gpen in our supplementary material. All samples are 4096 tokens in length. The conditional samples were generated by extending a 10 second priming melody. Samples can be heard at https://tinyurl.com/y6awtlv7.