Symbolic Music Generation with Transformer-GANs

Aashiq Muhamed[1*], Liang Li[1*], Xingjian Shi[1], Suri Yaddanapudi[1], Wayne Chi[1], Dylan Jackson[1], Rahul Suresh[1], Zachary C. Lipton[2], Alexander J. Smola[1]

[1] Amazon Web Services, [2] Carnegie Mellon University

{muhaaash, mzliang, xjshi, yaddas, waynchi, jacdylan, surerahu}@amazon.com

zlipton@cmu.edu, alex@smola.org

Abstract

Autoregressive models using Transformers have emerged as the dominant approach for music generation with the goal of synthesizing minute-long compositions that exhibit large-scale musical structure. These models are commonly trained by minimizing the negative log-likelihood (NLL) of the observed sequence in an autoregressive manner. Unfortunately, the quality of samples from these models tends to degrade significantly for long sequences, a phenomenon attributed to exposure bias. Fortunately, 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 Span-BERT model for the discriminator of the GAN, which in our experiments helped with training stability. We use the Gumbel-Softmax trick to obtain a differentiable approximation of the sampling process. This makes discrete sequences amenable to optimization in GANs. In addition, we broke the sequences into smaller chunks to ensure that we stay within a given memory budget. We demonstrate via human evaluations and a new discriminative metric that the 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.

Introduction

At present, neural sequence models are generally trained to maximize the likelihood of the observed sequences. This ensures statistical consistency but it can lead to undesirable artifacts when generating long sequences. While these artifacts are difficult to suppress with maximum likelihood training alone, they are easily detected by most sequence classifiers. We take advantage of this fact, incorporating an adversarial loss derived from GANs. To illustrate its benefits, we demonstrate improvements in the context of symbolic music generation.

Generative modeling as a field has progressed significantly in recent years, particularly with respect to creative applications such as art and music (Briot, Hadjeres, and Pachet 2017; Carnovalini and Rodà 2020; Anantrasirichai and Bull 2020). A popular application is the generation of symbolic music, a task that presents unique challenges not found in text generation due to polyphony and rhythm. At the same time, generating symbolic music can be simpler than audio generation due to the higher level of abstraction. Many language models from the NLP literature have been applied and extended to music generation. Since we build on this line of work, we use the terms *sequence models* and *language models* interchangeably throughout, depending on the context in which a model is mentioned.

Neural models for music sequences convert a digital representation of a musical score into a time-ordered sequence of discrete tokens. Language models are then trained on the event sequences with the objective of maximizing the likelihood of the data. Music can then be generated by sampling or beam-decoding from this model. Recent advancements in Natural Language Processing (NLP), especially the attention mechanism and the Transformer architecture (Vaswani et al. 2017), have helped advance state of the art in symbolic music generation (Huang et al. 2018; Payne 2019; Donahue et al. 2019). Music Transformer (Huang et al. 2018) and MuseNet (Payne 2019) use relative attention and sparse kernels (Child et al. 2019) respectively to remember long-term structure in the composition. More recent works in music generation (Donahue et al. 2019; Huang and Yang 2020; Wu, Wang, and Lei 2020) adopt the TransformerXL architecture (Dai et al. 2019) which uses recurrent memory to attend beyond a fixed context.

Despite recent improvements, these approaches exhibit crucial failure modes which we argue arise from the training objective; Music Transformer (Huang et al. 2018) occasionally forgets to switch off notes and loses coherence beyond a few target lengths as stated by the authors. 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 discriminative objective of a GAN (Goodfellow et al. 2014) should improve the fidelity of the generative model.

Unfortunately, incorporating GAN losses for discrete sequences can be difficult. Computing the derivative of the samples through the discrete sampling process is challenging. As such, many models (de Masson d'Autume et al.

[*]Equal contribution, corresponding authors.

2019; Nie, Narodytska, and Patel 2019) are limited to 20-40 token-length sentences, in contrast to the more than 1000 tokens required for minutes-long musical compositions. We leverage the Gumbel-Softmax (Kusner and Hernández-Lobato 2016) trick to obtain a differentiable approximation of the sampling process. Moreover, we use the Truncated Backpropagation Through Time (TBPTT) (Sutskever, Vinyals, and Le 2014) for gradient propagation on long sequences. The latter keeps memory requirements at bay.

Recent works on evaluation metrics in text generation (Salazar et al. 2019; Zhang et al. 2019; Montahaei, Alihosseini, and Baghshah 2019) suggest that BERT-based scores (Devlin et al. 2018) are well correlated with human rankings and jointly measure quality and diversity. As BERT is trained using a self-supervised loss on bidirectional contexts of all attention layers, it can be an effective way of extracting representations. We use them to obtain an effective metric for the generative aspect of the model.

Experiments show that the resulting Transformer-GAN improves over its vanilla counterpart. We evaluate the performance using the music quality metric of (Briot, Hadjeres, and Pachet 2017) and a number of more conventional metrics. In summary, our main contributions include:

- A novel Transformer-GAN approach for generating long music sequences of over 1000 tokens, using a pretrained SpanBERT as the discriminator.
- A detailed investigation of the influence of pretraining, loss functions, regularization, and number of frozen layers in the discriminator on music quality;
- A number of critical tricks for adversarial training;
- A classifier-based metric to evaluate generative models.

Related work

Generative models for sequences have a long history, from n-gram Markov-models to HMMs, to the more recent surge of neural sequence modeling research with rise in popularity of LSTMs in the early 2010s (Hochreiter and Schmidhuber 1997; Sutskever, Vinyals, and Le 2014; Briot, Hadjeres, and Pachet 2017). To apply sequential models to polyphonic music, the musical score (or performance data) is typically serialized into a single sequence by interleaving different instruments or voices (Oore et al. 2018).

Owing to their ability to model correlations at multiple timescales over long sequences, self-attention -based architectures are increasingly popular for generative music. Models such as the Transformer (Vaswani et al. 2017) can access any part of its previously generated output, at every step of generation. Two popular models, the Music Transformer (Huang et al. 2018) and MuseNet (Payne 2019) use Transformer decoders to generate music. Music Transformer uses the relative attention mechanism (Shaw, Uszkoreit, and Vaswani 2018) to generate long-term music structure at the scale of 2000 tokens. MuseNet adds several learned embeddings to guide the model to learn structural context. These embeddings were handcrafted to capture information related to chords and passage of time. Choi et al. (2019) uses Transformer encoders and decoders to harmonize or generate accompaniments to a given melody.

These autoregressive models all follow the standard teacher forcing strategy where one trains always to predict the next token, given a real sequence of the previous tokens as context. While models of the form $p(\mathbf{x}) = \prod_i p(x_{i+1}|x_{[1:i]})$ are statistically consistent, they suffer from an amplification of prediction errors: when generating a sequence, we end up *conditioning* on previously generated sequences (here, synthetic music) to produce the next note. The problem emerges because we condition on data that is distributionally unlike that seen at training time. This is a well known problem in sequence modeling (Bengio et al. 2015; Ranzato et al. 2015).

GANs (Goodfellow et al. 2014) have been applied to several domains as an alternative to maximum likelihood training, but directly applying GANs to sequence generation is known to be difficult (Lu et al. 2018). This is due to a number of reasons—training tends to be unstable, and these models tend to exhibit a phenomenon called mode collapse, where part of the input distribution's support is not covered by the generative model.

Language models are inherently discrete, since they involve sampling from a multinomial distribution of tokens. One option is to use an empirical average of $\mathbf{E}_{x\sim p(x)}[\partial_\theta \log p_\theta(x)]$, i.e. of the gradient of the log-likelihood of the data. This leads to the well-known REINFORCE algorithm (Williams 1992) and the application of Reinforcement Learning machinery (Wu, Li, and Yu 2020; Yu et al. 2017; Guo et al. 2018). It's best to view the resulting problem as one of a sequential decision-making process where the generator is the agent, the state comprises the generated tokens up to that time and the action is the next token to select. The generator is then trained via policy gradient with several designed reward functions (Lu et al. 2018).

An alternative is to obtain a continuously differentiable approximation of the sampling process via the Gumbel-Softmax (Kusner and Hernández-Lobato 2016). This yields a distribution over the token probability simplex. Nie, Narodytska, and Patel (2019) combine this with a relational memory based generator (akin to memory networks) and multiple embedded representations in the CNN discriminator. Wu et al. (2020) propose a new variational GAN training framework for text generation using a connection of GANs and reinforcement learning under a variational perspective.

Lastly, Zhang (2020) proposes an adversarial framework that uses Transformers for both the generator and discriminator. The author trained the GAN with a local and global reward function, noting that "a specific note should be in harmony with in its local pattern and the whole sequence should be in harmony with its global pattern".

Our proposed model combines GANs with Transformers to generate long, high-quality music sequences. It differs from previous works in the following ways:

- We use the Transformer-XL as our generator and pretrained BERT as the discriminator;
- We pretrain the BERT discriminator in the SpanBERT style (Joshi et al. 2020);
- We design an algorithm using the Gumbel-Softmax trick and a variant of the Truncated Backpropagation Through Time (TBPTT) algorithm (Sutskever, Vinyals, and Le

2014) to train on long sequences.

We are unaware of prior work studying self-supervised pretraining of the discriminator and its influence on generation.

Methodology

The key challenge is to make *long* sequence generation in GANs practical. We begin with an overview of data representation and training objectives, followed by a discussion of the network architecture, for generation and discrimination. We conclude with a list of useful tricks and techniques.

Data Representation

We take a language-modeling approach to train generative models for tokens. Picking a representation matching (Huang et al. 2018) allows us to compare directly to the Music Transformer in terms of its log-likelihood. More specifically, we use the encoding proposed by Oore et al. (2018), which consists of a vocabulary of 88 NOTE_ON events, 88 NOTE_OFFs, 100 TIME_SHIFTs, and 32 VELOCITY bins. This allows for expressive timing at 10ms and expressive dynamics (for details, see the Appendix).

Maximum Likelihood

Given a music sequence $\mathbf{x} = [x_1, x_2, ..., x_n]$, we model the unknown data distribution $p_r(\mathbf{x})$ autoregressively as

$$p_\theta(\mathbf{x}) = \prod_{t=1}^{n} p_\theta(x_t|x_1, ..., x_{t-1}).$$

Language models are typically trained using Maximum Likelihood. We seek a weight vector θ to minimize

$$L_{\text{mle}} = -\mathbb{E}_{\mathbf{x}^r \sim p_r} \left[\log p_\theta \left(\mathbf{x}^r\right)\right]. \tag{1}$$

Despite its attractive *theoretical* properties, maximum likelihood training suffers from many limitations, e.g. whenever the model is misspecified. This is illustrated by (Isola et al. 2017) in image-to-image translation, where no explicit loss function is available. Furthermore, teacher forcing introduces exposure bias (Bengio et al. 2015; Holtzman et al. 2019)—a distributional shift between training sequences used for learning and model data required for generation. This amplifies any errors in the estimate, sometimes creating strange, repetitive outputs.

Adversarial Losses

We address this problem by incorporating an adversarial loss into our objective. That is, we cast our model as *generator* G_θ and ensure that the sequences obtained from it match those on the training set as assessed by an appropriate *discriminator* D_ϕ. For our discriminator, we select a BERT model pretrained on music. We also regularize D_ϕ to prevent overfitting. During training, we alternate updates between the generator and discriminiator objectives:

$$L_{\text{G}} = L_{\text{mle}}[G_\theta] + \lambda L_{\text{gen}}[G_\theta] \tag{2}$$

$$L_{\text{D}} = L_{\text{disc}}[D_\phi] + \gamma L_{\text{reg}}[D_\phi] \tag{3}$$

Here $\lambda, \gamma > 0$ are hyperparameters. We investigate several choices for $L_{\text{gen}}, L_{\text{disc}}$ and L_{reg}: the gradient penalty of WGANs (Gulrajani et al. 2017), RSGAN losses (Jolicoeur-Martineau 2018), and PPO-GAN's loss (Wu et al. 2020).

WGAN with gradient penalty loss

$$L_{\text{gen}} = -\mathbb{E}_{\mathbf{x}^f \sim p_\theta}[D_\phi(\mathbf{x}^f)] \tag{4}$$

$$L_{\text{disc}} = -\mathbb{E}_{\mathbf{x}^r \sim p_r}[D_\phi(\mathbf{x}^r)] + \mathbb{E}_{\mathbf{x}^f \sim p_\theta}[D_\phi(\mathbf{x}^f)] \tag{5}$$

$$L_{\text{reg}} = \mathbb{E}_{\hat{\mathbf{x}} \sim p_{\hat{\mathbf{x}}}}[(\|\nabla_{\hat{\mathbf{x}}} D_\phi(\hat{\mathbf{x}})\|_2 - 1)^2] \tag{6}$$

D_ϕ is assumed to be a 1-Lipschitz continuous function. The smoothness is enforced with the gradient penalty loss. $\hat{\mathbf{x}}$ is drawn uniformly along straight lines (in embedding space) between pairs of points sampled from p_r and p_θ.

RSGAN loss The effective discriminator, $C(\mathbf{x}^r, \mathbf{x}^f)$ defined in terms of D_ϕ estimates the probability that the given real data is more realistic than randomly sampled fake data (σ is the sigmoid function).

$$C(\mathbf{x}^r, \mathbf{x}^f) = \sigma(D_\phi(\mathbf{x}^f) - D_\phi(\mathbf{x}^r)) \tag{7}$$

$$L_{\text{gen}} = -\mathbb{E}_{(\mathbf{x}^r, \mathbf{x}^f) \sim (p_r, p_\theta)}[\log(C(\mathbf{x}^r, \mathbf{x}^f))] \tag{8}$$

$$L_{\text{disc}} = -\mathbb{E}_{(\mathbf{x}^r, \mathbf{x}^f) \sim (p_r, p_\theta)}[\log(1 - C(\mathbf{x}^r, \mathbf{x}^f))] \tag{9}$$

PPO-GAN loss G_θ is treated as a policy and D_ϕ as a reward function. It transforms the conventional GAN objective as $L(\theta, q) = \mathbb{E}_q[D_\phi(\mathbf{x})] - \text{KL}(q(\mathbf{x})||p_\theta(\mathbf{x}))$ by introducing an auxiliary non-parametric function $q(\mathbf{x})$. Using the EM algorithm to alternatively optimize for q and θ leads to

$$L_{\text{gen}} = -\mathbb{E}_{\mathbf{x}^f \sim p_\theta}[D_\phi(\mathbf{x}^f)] + \text{KL}(p_\theta(\mathbf{x}) \,||\, p_{\theta^{(t)}}(\mathbf{x})) \tag{10}$$

$$L_{\text{disc}} = \mathbb{E}_{\mathbf{x}^r \sim p_r}[D_\phi(\mathbf{x}^r)] - \log(\int_x p_{\theta^{(t)}}(\mathbf{x}) \exp\{D_\phi(\mathbf{x})\} d\mathbf{x}) \tag{11}$$

$$L_{\text{reg}} = \mathbb{E}_{\hat{\mathbf{x}} \sim p_{\hat{\mathbf{x}}}}[(\|\nabla_{\hat{\mathbf{x}}} D_\phi(\hat{\mathbf{x}})\|_2 - 1)^2], \tag{12}$$

where $p_{\theta^{(t)}}$ is the generator distribution at the previous iteration. The KL penalty term is enforced using a clipped surrogate objective as in Proximal Policy Optimization (PPO) (Schulman et al. 2017). This loss also enforces 1-Lipschitz continuity in D_ϕ using a gradient penalty regularizer.

Transformer-GAN Architecture

Generator We use Transformer-XL (Dai et al. 2019) which introduces the notion of recurrence into the deep self-attention network. Instead of computing the hidden states from scratch for each new segment, it reuses the ones obtained in previous segments. These states serve as memory for the current segment, which builds up a recurrent connection between the segments. This makes it possible to model very long range dependencies, since information can be propagated through the recurrent connections. Note that our approach applies equally to other SOTA Transformers.

Discriminator Transformer-GANs differ from other efforts to incorporate GAN losses into sequence modeling (Lu et al. 2018) in the choice of discriminator. Training GANs using the Transformer as generator is a difficult problem (Chen et al. 2020; Zhang 2020) since training dynamics, memory overhead and the generator and discriminator losses need to be carefully balanced. In prior work, CNNs have proven to be useful discriminators for text generation (Kim 2014). The CNN-based discriminator in (Nie, Narodytska,

and Patel 2019) uses multiple embedded representations to provide more informative signals to the generator.

In this work, we propose using BERT as the discriminator to extract sequence embeddings followed by a pooling and linear layer. The bidirectional transformer is comparable in capacity to the transformer-based generator and uses the self-attention mechanism that captures meaningful aspects of the input music sequence. We speculate that this would help the discriminator provide informative gradients to the generator and stabilize the training process. Inspired by the observation by Mo, Cho, and Shin (2020) for images, we conjecture that freezing earlier layers of the pretrained discriminator in language GANs is a GAN transfer learning technique that is not prone to overfitting. Freezing layers in a pretrained discriminator, according to Mo, Cho, and Shin (2020) can be viewed as transfer learning, where we transfer knowledge useful for generation from a different dataset in the form of music representations. The lower layers of the discriminator—closest to the generator learn generic features of text—while the upper layers learn to classify whether the text is real or fake based on the extracted features.

Unlike Mo, Cho, and Shin (2020), where the discriminator is transferred between trained GANs on different datasets, we reshape this idea into a form that resembles representation learning in NLP. We pretrain our discriminator on the same dataset we train on using a self-supervised loss and test the hypothesis that the resulting learnt bidirectional representations are useful for the discriminator to classify real and fake data. In our setup, we simply freeze the lower layers of the discriminator and only fine-tune the upper layers. As we will show in Table 2, this achieved good performance on various proposed metrics.

SpanBERT style self-supervised pretraining, where we predict spans of masked tokens, enables the model to learn span representations. We hypothesize that span representations are better inductive biases for modeling coherence in music, as music is composed in spans of notes or chords. Given a masked span x_{mask} comprising tokens $(x_s, ..., x_e) \in \mathbf{x}$, the Masked Language Model (MLM) objective from SpanBERT for each token $x_i \in x_{\text{mask}}$ is

$$L_{MLM}(x_i) = -\log(P(x_i|x_{\backslash\text{mask}})), \tag{13}$$

where $x_{\backslash\text{mask}} = \{y \mid y \in \mathbf{x} \,\text{and}\, y \notin x_{\text{mask}}\}$. Freezing the discriminator also reduces the number of trainable parameters and training memory requirements, that are usually bottlenecks when training on long sequences.

Tricks of Adversarial Training

The adversarial training in (3) involves generating discrete samples $\mathbf{x}^f \sim p_\theta(\mathbf{x})$ autoregressively from the generator to feed into the discriminator. However, several issues exist in generating and training on these discrete samples, e.g., the non-differentiable sampling step, the repetition and high variance in generated samples, the high memory and compute complexity during backpropagation, and the instability during GAN training. In this section, we highlight a few critical tricks to address these issues.

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 $x^f_{t+1} \sim \text{softmax}(o_t)$. Here $o_t \in \mathbb{R}^V$ 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

$$x^f_{t+1} = \arg\max_{1 \le i \le V} (o^{(i)}_t + g^{(i)}_t), \tag{14}$$

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 Jang, Gu, and Poole (2016). Therefore, in the backward pass, (14) becomes

$$\text{softmax}(\beta(o_t + g_t)), \tag{15}$$

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.

Exponential inverse-temperature When using a fixed inverse temperature β in (15) 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 $\beta_n = \beta^{n/N}_{max}$ to increase β over iterations, where N is the maximum number of training iterations and n denotes the current iteration. Nie, Narodytska, and Patel (2019) 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 samples quality.

Conditional generation Another issue we notice is that learning $\mathbf{x}^f$ in (15) 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 $[x^r_1, ..., x^r_c]$ and sample the remaining $[x^f_{c+1}, ..., x^f_n]$ autoregressively.

Truncated Backpropagation Through Time (TBPTT) The generator sampling step in (15) is sequential and therefore the backward pass in the generator resembles the backpropagation through time (BPTT) algorithm (Tallec and Ollivier 2017). 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 (Yu et al. 2017; Zhang 2020; Chen et al. 2019) in RL based GANs.

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 (Gulrajani et al. 2017). Each token of sequence $\hat{\mathbf{x}}$ in (5) can be obtained by interpolating the discrete tokens in embedding space as $\hat{x}_t = \alpha \,\mathrm{embed}[x_t^r] + (1-\alpha)\,\mathrm{embed}[x_t^f]$, where embed denotes the embedding layer of the discriminator, and α is drawn from a uniform distribution on the interval $(0, 1)$.

Our hypothesis and findings on the importance of discriminator regularization align with prior work (Gulrajani et al. 2017) 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.

Metrics

The assessment and evaluation of generative models in music using objective metrics is challenging. In this section, we provide two metrics (used in text generation) for music quality evaluation, i.e. Classifier Accuracy (CA) and pseudo-log-likelihood (PLL) as well as objective metrics and human evaluation metrics.

Classifier accuracy Inspired by the relative metrics used in (Yang and Lerch 2020), we propose using a classifier accuracy score as a measure of music realism that could detect exposure bias. We train a separate model to distinguish between real (our validation set) and generated data and use its accuracy on a held-out set as this classifier accuracy (CA) score. This metric is an implementation of the more general Classifier Two-Sample Tests (C2ST) (Lopez-Paz and Oquab 2016). The predictive uncertainty of the classifier can also be used to inspect where the real and generated data differ. The classifier accuracy can be used to score generative models as we show in Table 1. The lower the accuracy, the closer are the generated samples to the real data distribution. Our classifier is built with our pretrained frozen BERT model trained using SpanBERT, and a Support Vector Machine (SVM) on top. While we could pick an arbitrary classifier, we implement C2ST with the SVM, a margin classifier with finite norm known for its fast convergence. The SVM is retrained for every model we evaluate.

Pseudo-log-likelihood As BERT is trained with the Masked Language Model (MLM) objective, it learns bidirectional representations. The pseudo-log-likelihood score (PLLs) is derived from MLMs and is given by summing the conditional log probabilities $\log P(x_t|\mathbf{x}_{\setminus t})$ of each sentence token. The PLL score is better when it is closer to the PLL of the training set. The pseudo-log-likelihood score of a sentence $\mathbf{x}$ can be defined as

$$PLL(\mathbf{x}) = -\sum_{t=1}^{|\mathbf{x}|} \log(P(x_t|\mathbf{x}_{\setminus t})), \tag{16}$$

where $\mathbf{x}_{\setminus t} := (x_1, ...x_{t-1}, x_{t+1}, ..., x_{|\mathbf{x}|})$.

PLL is an intrinsic value that one can assign to sentences and corpora, allowing one to use MLMs to evaluate a sequence of tokens in applications previously restricted to conventional language model scoring. PLL scores as proposed in (Salazar et al. 2019) to measure linguistic acceptability. We therefore propose to use this as a metric to evaluate music samples. PLL scores are closely related to log-likelihood scores and their roles overlap as the models get stronger. Recent work in text evaluation (Basu et al. 2020) suggest that sample likelihood and quality are only correlated within a certain likelihood range.

Quantitative metrics To evaluate the music generated, we use several conventional objective metrics used in music evaluation research (Dong et al. 2018; Yang and Lerch 2020). These metrics are computed for both real (training set) data and generated data, and their values are compared. The closer the values are to the dataset, the better the score. We also report the validation NLL by calculating the NLL for each token and averaging over all tokens in the validation set. We generate 200 unconditional samples—each 4096 tokens in length—for each model we want to evaluate.

Human evaluation For each model, we generated 84 samples from a set of 7 priming sequences. The samples were 1 minute in duration and the primes were 10 seconds in duration. Each survey contained 7 questions corresponding to each priming sequence. Participants were given a random survey from a set of 5 surveys. For each question in the survey, participants were presented with a random set of musical samples, where each sample is from a different model, but from the same priming sequence. They were asked to (a) rate the score of the sample in a range of 0 to 5 and (b) rank the samples based on their coherence and consistency.

Implementation Details

We benchmark our models on the MAESTRO MIDI V1 dataset (Hawthorne et al. 2019), which contains over 200 hours of paired audio and MIDI recordings from ten years of the International Piano-e-Competition. The dataset is split into 80/10/10 for training/validation/evaluation. We used the same data augmentation as in Music Transformer, where we augmented the data by uniform pitch transposition from {-3, -2, ..., 2, 3} and stretched time with ratios of {0.95, 0.975, 1.0, 1.025, 1.05}.

Hyperparameter configuration The hyperparameters we used for the Transformer-XL architecture are shown in Table 3. For training, we used a 0.004 initial learning rate, the inverse square root scheduler and Adam optimizer. We used a target length of 128 for both training and evaluation, since we found this value offers a reasonable trade-off between training time and performance on metrics. Since TBPTT addresses the memory bottleneck, our framework can train on sequence lengths longer than 128. We set memory length for the Transformer-XL as 1024 during training and 2048 during evaluation. During sample generation, we set memory length equal to the number of tokens to generate. We observed, as in (Dai et al. 2019), that NLL and generated music quality were sensitive to memory length. We introduced a reset memory feature into the Transformer-XL training process as clearing the Transformer-XL memory at the beginning of each new MIDI file. We report the baseline models with the lowest validation NLL. More details on our hyperparameter sweep and settings can be found in the Appendix.

All our GANs and baselines use the same Transformer-XL configuration. We set the sequence length of generated samples during adversarial training as 128 (equal to target length). Our GAN generator is initialized using our best NLL-trained baseline model. We follow an alternating training procedure to update the generator and discriminator using the NLL and GAN losses. The NLL loss frequency is five times the GAN loss frequency. We used $\beta_{max} = 100$ in all our experiments as in Nie, Narodytska, and Patel (2019). For Music Transformer, we use the implementation in Tensor2Tensor (Vaswani et al. 2018). We run the baseline experiments with three different seeds and run the GAN experiments with one seed.

Sampling methods At each step t of generation, Random sampling samples from o_t while TopK sampling samples from the tokens corresponding to the K highest probabilities in o_t. All sampling methods use a fixed temperature of 0.95.

Experiments and Results

Transformer-XL and Music Transformer Transformer-XL achieved comparable overall performance to Music Transformer, the current state-of-the-art model. We see from Table 1 that Transformer-XL achieves lower NLL and is comparable to Music Transformer on several objective metrics. This is also reflected in the human evaluation scores in 1. Transformer-XL uses the relative attention mechanism akin to Music Transformer which explains their similar performance on metrics. These architectures act as our baselines trained with MLE.

Transformer-GAN vs MLE baselines We compare the Transformer-GAN framework trained with different loss types and discriminator choices in Table 1. We see that the Transformer-GAN with the BERT discriminator scores better on the CA metric than our baselines. This can be attributed to GAN training, that reduces the distributional discrepancy between real and generated data. Fig. 1 shows that the Transformer-GAN with WGAN with gradient penalty (WGAN-GPen) outperforms our baselines in human evaluation, proving the validity of our proposed GAN model. A Kruskal-Wallis H test of ratings showed that the statistically significant difference between Transformer-GAN (Random) and Transformer-XL (Random) is $\mathcal{X}^2(2) = 3.272, p = 0.031$ with WGAN-GPen. In the same figure, we show that the Transformer-GAN trained with the PPO-GAN with gradient penalty (PPO-GPen), also outperforms the baselines.

Discriminator architecture We experimented with two discriminator architectures—the CNN-based discriminator used in Nie, Narodytska, and Patel (2019) and our BERT discriminator. We see in Table 1 that WGAN-GPen using the CNN (i) performs worse than WGAN-GPen using BERT on several objective metrics and (ii) performs worse than Transformer-XL on CA and PLL scores. In addition, the GAN model's performance has been shown to be sensitive to both random parameter initialization and hyperparameter choices (Semeniuta, Severyn, and Gelly 2018), which can be a possible explanation to why a CNN discriminator performs worse since CNN-based started with randomly initialization. Using a pretrained BERT discriminator, to an extent, helps address this sensitivity.

GAN loss type We trained our Transformer-GAN using four different GAN losses: RSGAN, RSGAN with gradient penalty (RSGAN-GPen), WGAN-GPen, and PPO-GPen. In Table 1, we compare their performances on several quantitative metrics. We see that (i) RSGAN performs worse than RSGAN-GPen, indicating the importance of the gradient penalty regularizer. RSGAN performs worse than Transformer-XL on the CA metric, suggesting that GPen was essential to make the GAN loss work; (ii) WGAN-GPen achieves the highest scores on CA and PLL, and beats other models on our objective metrics. In informal tests, we noticed that it was hard for people to distinguish between the different Transformer-GAN models trained with GPen.

Effect of frozen layers/initialization We perform ablation studies on our BERT discriminator to understand the effect of freezing layers of our pretrained discriminator during GAN training. In Table 2, the first row corresponds to the randomly initialized BERT without any pretraining. We observe that (i) A randomly initialized BERT discriminator scores poorly on our objective metrics compared to the pretrained discriminator (ii) Freezing more layers in the pretrained discriminator tends to improve objective metric scores, in particular CA. These results suggest that discriminator priors can play an important role in GAN training.

Sampling methods We experiment with Random and TopK sampling and how it influences music evaluation scores in the Appendix. We find that (i) Transformer-XL samples sampled with TopK score better than those sampled with Random on our objective metrics. This is also reflected in the human evaluation scores in Fig. 1. (ii) Transformer-GAN samples sampled with TopK score lower than those sampled with Random on several objective metrics and human evaluation. (iii) Music generated using top-k sampling scores higher on the pseudo-likelihood metric, suggesting that this metric is sensitive to a distributional bias towards

	Discriminator	NLL ↓	Sampling	CA ↓	PLL ∼	PCU ∼	ISR ∼	PRS ∼	TUP ∼	PR ∼	APS ∼	IOI ∼
Training Set	–	–	–	–	2.0203	7.810	0.586	0.399	65.28	67.34	11.531	0.133
Music Transformer	–	1.79	Random	0.8443	2.5666	7.214	0.599	0.465	54.95	62.035	**11.619**	0.113
Transformer-XL	–	**1.74**	Top32	0.8377	2.1531	7.045	0.572	0.284	52.95	60.39	11.117	0.107
WGAN-GPen	CNN	1.75	Random	0.8401	2.3087	6.95	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	**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	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**	0.578	0.304	54.11	62.83	11.461	0.136
RSGAN	Pretrained BERT	1.75	Random	0.8615	2.1084	6.56	0.607	0.192	48.165	55.62	11.256	0.082

Table 1: Quantitative music metrics: NLL (Negative likelihood); CA (SpanBERT classifier accuracy); PLL (Pseudo-log-likelihood score); PCU (Unique pitch classes); ISR (Nonzero entries in C major scale / 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). Bolded values are better. Metrics marked with $\sim$ are better when closer to the Training Set.

	Frozen layers	NLL ↓	CA ↓	PLL ∼	PCU ∼	ISR ∼	PRS ∼	TUP ∼	PR ∼	APS ∼	IOI ∼
Training Set	–	–	–	2.0203	7.810	0.586	0.399	65.28	67.34	11.531	0.133
WGAN-Gpen	random-init	1.75	0.8359	2.2881	7.114	0.605	0.261	53.44	61.355	11.246	0.1245
WGAN-Gpen	['emb']	1.75	0.8435	2.3499	7.255	0.588	0.364	54.945	62.56	**11.540**	**0.1316**
WGAN-Gpen	['emb', '0']	1.75	0.8852	2.3490	7.135	**0.586**	0.348	53.795	61.565	11.158	0.1355
WGAN-Gpen	['emb', '0', '1', '2']	1.75	0.8586	2.4970	**7.345**	0.582	**0.397**	55.06	63.9	11.985	0.136
WGAN-Gpen	['emb', '0', '1', '2', '3']	1.75	0.8394	2.4538	6.925	0.580	0.362	50.965	59.465	10.533	0.1614
WGAN-Gpen	['emb', '0', '1', '2', '3', '4']	1.75	**0.8179**	**2.1020**	7.020	0.606	0.277	**55.56**	**63.23**	11.935	0.145

Table 2: Ablation studies of frozen layers and random weight initialization. Note BERT has 6 layers: 'embedding', 'attention 0', 'attention 1', 'attention 2', 'attention 3', 'attention 4', which is denoted as emb, 0, 1, 2, 3, 4. All samples were generated with Random Sampling. Bolded values are better. Metrics marked with $\sim$ are better when closer to the Training Set.

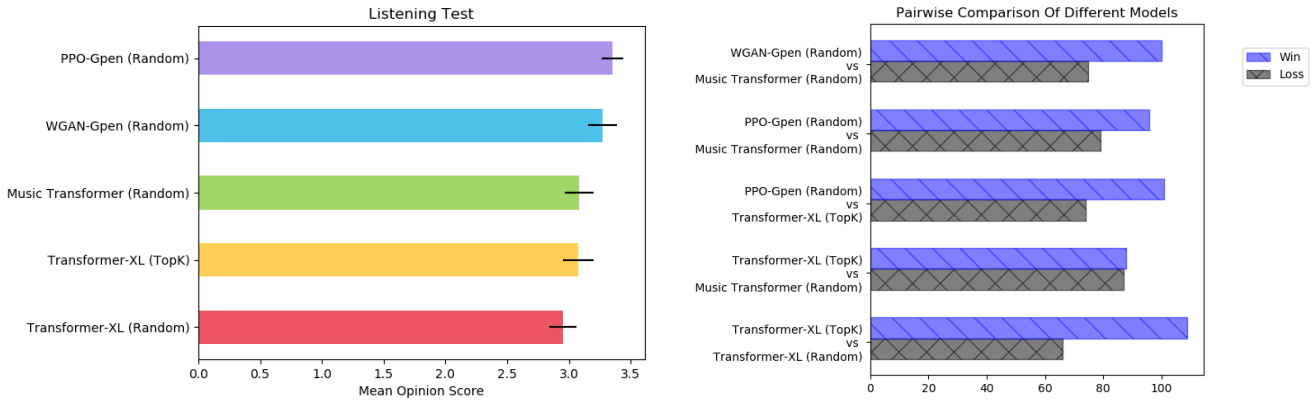

Figure 1: Human Evaluations. Left: average ratings for each model and error bars as standard error. Right: pairwise comparisons between models. For 'A vs B', when A beats B it is a 'Win' and when B beats A it is a 'Loss'.

Hyperparameters	Music Transformer	Transformer-XL
Layers	8	6
Dropout	0.2	0.1
Hidden size	384	500
Target length	2048	128
Memory length	-	2048
Number heads	8	10
Number of parameters	16253189	13677310

Table 3: Hyperparameters of baseline models

higher likelihood. A possible explanation for the apparent contradictory behavior observed in (i) and (ii) can be attributed to sampling during adversarial training. We speculate that these results might owe to the fact that the Transformer GAN is trained using the Random sampling to decode when generating sequences to feed the discriminator.

Conclusion and Discussion

We proposed a new framework for generating long-term coherent music based on adversarial training of Transformers. The results obtained from various experiments demonstrate that our Transformer-GAN achieves better performance compared to other transformers trained by maximizing likelihood alone. By sampling during training, the adversarial loss helps bridge the discrepancy between the training objective and generation. We have demonstrated that using a bidirectional transformer can indeed provide a useful signal to the generator, contrary to the findings in de Masson d'Autume et al. (2019). In future work, we plan to extend our work by pretraining on larger datasets where our idea can be beneficial.

References

Anantrasirichai, N.; and Bull, D. 2020. Artificial Intelligence in the Creative Industries: A Review. *arXiv preprint arXiv:2007.12391* .

Basu, S.; Ramachandran, G. S.; Keskar, N. S.; and Varshney, L. R. 2020. Mirostat: A Perplexity-Controlled Neural Text Decoding Algorithm. *arXiv preprint arXiv:2007.14966* .

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

Briot, J.-P.; Hadjeres, G.; and Pachet, F.-D. 2017. Deep learning techniques for music generation–a survey. *arXiv preprint arXiv:1709.01620* .

Carnovalini, F.; and Rodà, A. 2020. Computational Creativity and Music Generation Systems: An Introduction to the State of the Art. *Frontiers Artif. Intell.* 3: 14.

Chen, X.; Cai, P.; Jin, P.; Wang, H.; Dai, X.; and Chen, J. 2020. A Discriminator Improves Unconditional Text Generation without Updating the Generator. *arXiv preprint arXiv:2004.02135* .

Chen, X.; Li, Y.; Jin, P.; Zhang, J.; Dai, X.; Chen, J.; and Song, G. 2019. Adversarial Sub-sequence for Text Generation. *arXiv preprint arXiv:1905.12835* .

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

Choi, K.; Hawthorne, C.; Simon, I.; Dinculescu, M.; and Engel, J. 2019. Encoding musical style with transformer autoencoders. *arXiv preprint arXiv:1912.05537* .

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

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

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

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

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

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

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

Guo, J.; Lu, S.; Cai, H.; Zhang, W.; Yu, Y.; and Wang, J. 2018. Long text generation via adversarial training with leaked information. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Hawthorne, C.; Stasyuk, A.; Roberts, A.; Simon, I.; Huang, C.-Z. A.; Dieleman, S.; Elsen, E.; Engel, J.; and Eck, D. 2019. Enabling Factorized Piano Music Modeling and Generation with the MAESTRO Dataset. In *International Conference on Learning Representations*. URL https://openreview.net/forum?id=r1lYRjC9F7.

Hochreiter, S.; and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8): 1735–1780.

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

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

Huang, Y.-S.; and Yang, Y.-H. 2020. Pop music transformer: Generating music with rhythm and harmony. *arXiv preprint arXiv:2002.00212* .

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

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

Jolicoeur-Martineau, A. 2018. The relativistic discriminator: a key element missing from standard GAN. *arXiv preprint arXiv:1807.00734* .

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

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

Kusner, M. J.; and Hernández-Lobato, J. M. 2016. GANs for sequences of discrete elements with the gumbel-softmax distribution. *arXiv preprint arXiv:1611.04051* .

Lopez-Paz, D.; and Oquab, M. 2016. Revisiting classifier two-sample tests. *arXiv preprint arXiv:1610.06545* .

Lu, S.; Zhu, Y.; Zhang, W.; Wang, J.; and Yu, Y. 2018. Neural text generation: Past, present and beyond. *arXiv preprint arXiv:1803.07133* .

Mo, S.; Cho, M.; and Shin, J. 2020. Freeze Discriminator: A Simple Baseline for Fine-tuning GANs. *arXiv preprint arXiv:2002.10964* .

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

Nie, W.; Narodytska, N.; and Patel, A. 2019. RelGAN: Relational generative adversarial networks for text generation. In *ICLR*.

Oore, S.; Simon, I.; Dieleman, S.; Eck, D.; and Simonyan, K. 2018. This time with feeling: Learning expressive musical performance. *Neural Computing and Applications* 1–13.

Payne, C. 2019. MuseNet. URL openai.com/blog/musenet.

Ranzato, M.; Chopra, S.; Auli, M.; and Zaremba, W. 2015. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732* .

Salazar, J.; Liang, D.; Nguyen, T. Q.; and Kirchhoff, K. 2019. Masked Language Model Scoring. *arXiv preprint arXiv:1910.14659* .

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. arXiv 2017. *arXiv preprint arXiv:1707.06347* .

Semeniuta, S.; Severyn, A.; and Gelly, S. 2018. On accurate evaluation of gans for language generation. *arXiv preprint arXiv:1806.04936* .

Shaw, P.; Uszkoreit, J.; and Vaswani, A. 2018. Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155* .

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

Tallec, C.; and Ollivier, Y. 2017. Unbiasing truncated backpropagation through time. *arXiv preprint arXiv:1705.08209* .

Vaswani, A.; Bengio, S.; Brevdo, E.; Chollet, F.; Gomez, A. N.; Gouws, S.; Jones, L.; Kaiser, L.; Kalchbrenner, N.; Parmar, N.; Sepassi, R.; Shazeer, N.; and Uszkoreit, J. 2018. Tensor2Tensor for Neural Machine Translation. *CoRR* abs/1803.07416. URL http://arxiv.org/abs/1803.07416.

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

Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8(3-4): 229–256.

Wu, Q.; Li, L.; and Yu, Z. 2020. TextGAIL: Generative Adversarial Imitation Learning for Text Generation. *arXiv preprint arXiv:2004.13796* .

Wu, X.; Wang, C.; and Lei, Q. 2020. Transformer-XL Based Music Generation with Multiple Sequences of Time-valued Notes. *arXiv preprint arXiv:2007.07244* .

Wu, Y.; Zhou, P.; Wilson, A. G.; Xing, E. P.; and Hu, Z. 2020. Improving GAN Training with Probability Ratio Clipping and Sample Reweighting. *arXiv preprint arXiv:2006.06900* .

Yang, L.-C.; and Lerch, A. 2020. On the evaluation of generative models in music. *Neural Computing and Applications* 32(9): 4773–4784.

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

Zhang, N. 2020. Learning Adversarial Transformer for Symbolic Music Generation. *IEEE Transactions on Neural Networks and Learning Systems* .

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