It's Raw! Audio Generation with State-Space Models

Karan Goel[1] **Albert Gu**[1] **Chris Donahue**[1] **Christopher Ré**[1]

Abstract

Developing architectures suitable for modeling raw audio is a challenging problem due to the high sampling rates of audio waveforms. Standard sequence modeling approaches like RNNs and CNNs have previously been tailored to fit the demands of audio, but the resultant architectures make undesirable computational tradeoffs and struggle to model waveforms effectively. We propose SASHIMI, a new multi-scale architecture for waveform modeling built around the recently introduced S4 model for long sequence modeling. We identify that S4 can be unstable during autoregressive generation, and provide a simple improvement to its parameterization by drawing connections to Hurwitz matrices. SASHIMI yields state-of-the-art performance for unconditional waveform generation in the autoregressive setting. Additionally, SASHIMI improves non-autoregressive generation performance when used as the backbone architecture for a diffusion model. Compared to prior architectures in the autoregressive generation setting, SASHIMI generates piano and speech waveforms which humans find more musical and coherent respectively, e.g. 2× better mean opinion scores than WaveNet on an unconditional speech generation task.[1] On a music generation task, SASHIMI outperforms WaveNet on density estimation and speed at both training and inference even when using 3× fewer parameters.

1. Introduction

Generative modeling of raw audio *waveforms* is a challenging frontier for machine learning due to their high-dimensionality—waveforms contain tens of thousands of timesteps per second and exhibit long-range behavior at multiple timescales. A key problem is developing architectures for modeling waveforms with the following properties:

1. **Globally coherent** generation, which requires modeling unbounded contexts with long-range dependencies.
2. **Computational efficiency** through parallel training, and fast autoregressive and non-autoregressive inference.
3. **Sample efficiency** through a model with inductive biases well suited to high-rate waveform data.

Among the many training methods for waveform generation, autoregressive (AR) modeling is a fundamentally important approach. AR models learn the distribution of future variables conditioned on past observations, and are central to recent advances in machine learning for language and image generation (Brown et al., 2020; Ramesh et al., 2021; Bommasani et al., 2021). With AR models, computing the exact likelihood is tractable, which makes them simple to train, and lends them to applications such as lossless compression (Kleijn et al., 2018) and posterior sampling (Jayaram & Thickstun, 2021). When generating, they can condition on arbitrary amounts of past context to sample sequences of unbounded length—potentially even longer than contexts observed during training. Moreover, architectural developments in AR waveform modeling can have a cascading effect on audio generation more broadly. For example, WaveNet—the earliest such architecture (van den Oord et al., 2016)—remains a central component of state-of-the-art approaches for text-to-speech (TTS) (Li et al., 2019), unconditional generation (Lakhotia et al., 2021), and non-autoregressive (non-AR) generation (Kong et al., 2021).

Despite notable progress in AR modeling of (relatively) short sequences found in domains such as natural language (e.g. 1K tokens), it is still an open challenge to develop architectures that are effective for the much longer sequence lengths of audio waveforms (e.g. 1M samples). Past attempts have tailored standard sequence modeling approaches like CNNs (van den Oord et al., 2016), RNNs (Mehri et al., 2017), and Transformers (Child et al., 2019) to fit the demands of AR waveform modeling, but these approaches have limitations. For example, RNNs lack computational efficiency because they cannot be parallelized during training, while CNNs cannot achieve global

[1] Department of Computer Science, Stanford University. Correspondence to: Karan Goel <kgoel@cs.stanford.edu>, Albert Gu <albertgu@stanford.edu>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

[1] Examples: https://hazyresearch.stanford.edu/sashimi-examples

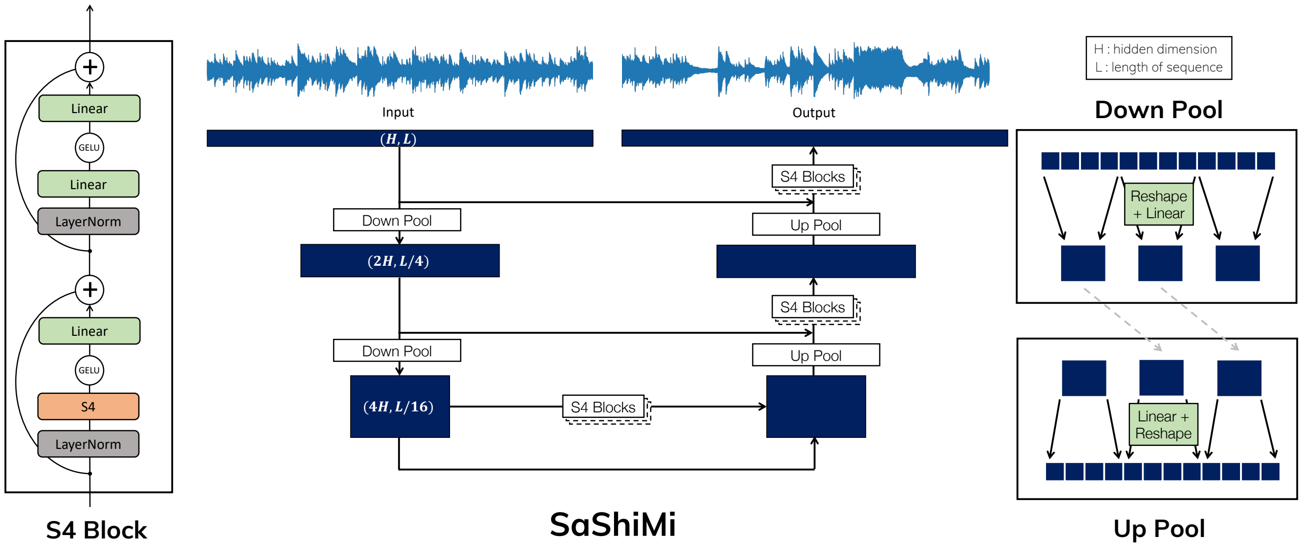


Figure 1. SaShiMi consists of a simple repeated block combined with a multiscale architecture. (*Left*) The basic S4 block is composed of an S4 layer combined with standard pointwise linear functions, non-linearities, and residual connections. (*Center*) Dark blue rectangles illustrate the shape of inputs. The input is progressively transformed to shorter and wider sequences through pooling layers, then transformed back with stacks of S4 blocks. Longer range residual connections are included to help propagate signal through the network. (*Right*) Pooling layers are position-wise linear transformations with shifts to ensure causality.

coherence because they are fundamentally constrained by the size of their receptive field.

We introduce **SaShiMi**, a new architecture for modeling waveforms that yields state-of-the-art performance on unconditional audio generation benchmarks in both the AR and non-AR settings. SaShiMi is designed around recently developed deep state space models (SSM), specifically S4 (Gu et al., 2022). SSMs have a number of key features that make them ideal for modeling raw audio data. Concretely, S4:

1. Incorporates a principled approach to modeling long range dependencies with strong results on long sequence modeling, including raw audio classification.
2. Can be computed either as a CNN for efficient parallel training, or an RNN for fast autoregressive generation.
3. Is implicitly a continuous-time model, making it well-suited to signals like waveforms.

To realize these benefits of SSMs inside SaShiMi, we make 3 technical contributions. First, we observe that while stable to train, S4's recurrent representation cannot be used for autoregressive generation due to numerical instability. We identify the source of the instability using classical state space theory, which states that SSMs are stable when the state matrix is Hurwitz, which is not enforced by the S4 parameterization. We provide a simple improvement to the S4 parameterization that theoretically ensures stability.

Second, SaShiMi incorporates pooling layers between blocks of residual S4 layers to capture hierarchical information across multiple resolutions. This is a common technique in neural network architectures such as standard CNNs and multi-scale RNNs, and provides empirical improvements in both performance and computational efficiency over isotropic stacked S4 layers.

Third, while S4 is a causal (unidirectional) model suitable for AR modeling, we provide a simple bidirectional relaxation to flexibly incorporate it in non-AR architectures. This enables it to better take advantage of the available global context in non-AR settings.

For AR modeling in audio domains with unbounded sequence lengths (e.g. music), SaShiMi can train on much longer contexts than existing methods including WaveNet (sequences of length 128K vs 4K), while simultaneously having better test likelihood, faster training and inference, and fewer parameters. SaShiMi outperforms existing AR methods in modeling the data ($>$ 0.15 bits better negative log-likelihoods), with substantial improvements ($+0.4$ points) in the musicality of long generated samples (16s) as measured by mean opinion scores. In unconditional speech generation, SaShiMi achieves superior global coherence compared to previous AR models on the difficult SC09 dataset both quantitatively (80% higher inception score) and qualitatively (2× higher audio quality and digit intelligibility opinion scores by human evaluators).

Finally, we validate that SaShiMi is a versatile backbone for non-AR architectures. Replacing the WaveNet backbone with SaShiMi in the state-of-the-art diffusion model DiffWave improves its quality, sample efficiency, and robustness to hyperparameters with no additional tuning.

Our Contributions. The central contribution of this paper is showing that deep neural networks using SSMs are a strong alternative to conventional architectures for modeling audio waveforms, with favorable tradeoffs in training speed, generation speed, sample efficiency, and audio quality.

- We technically improve the parameterization of S4, ensuring its stability when switching into recurrent mode at generation time.
- We introduce SASHIMI, an SSM-based architecture with high efficiency and performance for unconditional AR modeling of music and speech waveforms.
- We show that SASHIMI is easily incorporated into other deep generative models to improve their performance.

2. Related Work

This work focuses primarily on the task of generating raw audio waveforms without conditioning information. Most past work on waveform generation involves conditioning on localized intermediate representations like spectrograms (Shen et al., 2018; Kumar et al., 2019; Prenger et al., 2019), linguistic features (van den Oord et al., 2016; Kalchbrenner et al., 2018; Bińkowski et al., 2020), or discrete audio codes (van den Oord et al., 2017; Dieleman et al., 2018b; Dhariwal et al., 2020; Lakhotia et al., 2021). Such intermediaries provide copious information about the underlying content of a waveform, enabling generative models to produce globally-coherent waveforms while only modeling local structure.

In contrast, modeling waveforms in an unconditional fashion requires learning both local and global structure with a single model, and is thus more challenging. Past work in this setting can be categorized into AR approaches (van den Oord et al., 2016; Mehri et al., 2017; Child et al., 2019), where audio samples are generated one at a time given previous audio samples, and non-AR approaches (Donahue et al., 2019; Kong et al., 2021), where entire waveforms are generated in a single pass. While non-AR approaches tend to generate waveforms more efficiently, AR approaches have two key advantages. First, unlike non-AR approaches, they can generate waveforms of unbounded length. Second, they can tractably compute exact likelihoods, allowing them to be used for compression (Kleijn et al., 2018) and posterior sampling (Jayaram & Thickstun, 2021).

In addition to these two advantages, new architectures for AR modeling of audio have the potential to bring about a cascade of improvements in audio generation more broadly. For example, while the WaveNet architecture was originally developed for AR modeling (in both conditional and unconditional settings), it has since become a fundamental piece of infrastructure in numerous audio generation systems. For instance, WaveNet is commonly used to *vocode* intermediaries such as spectrograms (Shen et al., 2018) or discrete audio codes (van den Oord et al., 2017) into waveforms, often in the context of text-to-speech (TTS) systems. Additionally, it serves as the backbone for several families of non-AR generative models of audio in both the conditional and unconditional settings:

(i) Distillation: Parallel WaveNet (van den Oord et al., 2018) and ClariNet (Ping et al., 2019) distill parallelizable flow models from a teacher WaveNet model.

(ii) Likelihood-based flow models: WaveFlow (Ping et al., 2020), WaveGlow (Prenger et al., 2019), and FloWaveNet (Kim et al., 2019) all use WaveNet as a core component of reversible flow architectures.

(iii) Autoencoders: WaveNet Autoencoder (Engel et al., 2017) and WaveVAE (Peng et al., 2020), which use WaveNets in their encoders.

(iv) Generative adversarial networks (GAN): Parallel WaveGAN (Yamamoto et al., 2020) and GAN-TTS (Bińkowski et al., 2020), which use WaveNets in their discriminators.

(v) Diffusion probabilistic models: WaveGrad (Chen et al., 2021) and DiffWave (Kong et al., 2021) learn a reversible noise diffusion process on top of dilated convolutional architectures.

In particular, we point out that DiffWave represents the state-of-the-art for unconditional waveform generation, and incorporates WaveNet as a black box.

Despite its prevalence, WaveNet is unable to model long-term structure beyond the length of its receptive field (up to 3s), and in practice, may even fail to leverage available information beyond a few tens of milliseconds (Shen et al., 2018). Hence, we develop an alternative to WaveNet which can leverage unbounded context. We focus primarily on evaluating our proposed architecture SASHIMI in the fundamental AR setting, and additionally demonstrate that, like WaveNet, SASHIMI can also transfer to non-AR settings.

3. Background

We provide relevant background on autoregressive waveform modeling in Section 3.1, state-space models in Section 3.2 and the recent S4 model in Section 3.3, before introducing SASHIMI in Section 4.

3.1. Autoregressive Modeling of Audio

Given a distribution over waveforms $x = (x_0, \dots, x_{T-1})$, autoregressive generative models model the joint distribution as the factorized product of conditional probabilities

$$p(x) = \prod_{t=0}^{T-1} p(x_t | x_0, \dots, x_{t-1}).$$

Autoregressive models have two basic modes:

Training: Given a sequence of samples $x_0, \dots, x_{T-1}$, maximize the likelihood

$$p(x_0, \dots, x_{T-1}) = \sum_{i=0}^{T-1} p(x_i | x_0, \dots, x_{i-1}) = \sum_{i=0}^{T-1} \ell(y_i, x_{i+1})$$

where ℓ is the cross-entropy loss function.

Inference (Generation): Given $x_0, \dots, x_{t-1}$ as context, sample from the distribution represented by $y_{t-1} = p(x_t \mid x_0, \dots, x_{t-1})$ to produce the next sample x_t.

We remark that by the training mode, autoregressive models are equivalent to *causal sequence-to-sequence maps* $x_0, \dots, x_{T-1} \mapsto y_0, \dots, y_{T-1}$, where x_k are input samples to model and y_k represents the model's guess of $p_(x_{k+1} \mid x_0, \dots, x_k)$. For example, when modeling a sequence of categorical inputs over k classes, typically $x_k \in \mathbb{R}^d$ are embeddings of the classes and $y_k \in \mathbb{R}^k$ represents a categorical distribution over the classes.

The most popular models for autoregressive audio modeling are based on CNNs and RNNs, which have different tradeoffs during training and inference. A CNN layer computes a convolution with a parameterized kernel

$$K = (k_0, \dots, k_{w-1}) \qquad y = K * x \tag{1}$$

where w is the width of the kernel. The *receptive field* or *context size* of a CNN is the sum of the widths of its kernels over all its layers. In other words, modeling a context of size T requires learning a number of parameters proportional to T. This is problematic in domains such as audio which require very large contexts.

A variant of CNNs particularly popular for modeling audio is the *dilated convolution* (DCNN) popularized by WaveNet (van den Oord et al., 2016), where each kernel K is non-zero only at its endpoints. By choosing kernel widths carefully, such as in increasing powers of 2, a DCNN can model larger contexts than vanilla CNNs.

RNNs such as SampleRNN (Mehri et al., 2017) maintain a hidden state h_t that is sequentially computed from the previous state and current input, and models the output as a function of the hidden state

$$h_t = f(h_{t-1}, x_t) \qquad y_t = g(h_t) \tag{2}$$

The function f is also known as an RNN cell, such as the popular LSTM (Hochreiter & Schmidhuber, 1997).

CNNs and RNNs have efficiency tradeoffs as autoregressive models. CNNs are *parallelizable*: given an input sequence $x_0, \dots, x_{T-1}$, they can compute all y_k at once, making them efficient during training. However, they become awkward at inference time when only the output at a single timestep y_t is needed. Autoregressive stepping requires specialized caching implementations that have higher complexity requirements than RNNs.

On the other hand, RNNs are *stateful*: The entire context $x_0, \dots, x_t$ is summarized into the hidden state h_t. This makes them efficient at inference, requiring only constant time and space to generate the next hidden state and output. However, this inherent sequentiality leads to slow training and optimization difficulties (the vanishing gradient problem (Hochreiter et al., 2001; Pascanu et al., 2013)).

3.2. State Space Models

A recent class of deep neural networks was developed that have properties of both CNNs and RNNs. The state space model (SSM) is defined in continuous time by the equations

$$\begin{aligned} h'(t) &= Ah(t) + Bx(t) \\ y(t) &= Ch(t) + Dx(t) \end{aligned}. \tag{3}$$

To operate on discrete-time sequences sampled with a step size of Δ, SSMs can be computed with the recurrence

$$h_k = \overline{A}h_{k-1} + \overline{B}x_k \qquad y_k = Ch_k + Dx_k \tag{4}$$

$$\overline{A} = (I - \Delta/2 \cdot A)^{-1}(I + \Delta/2 \cdot A) \tag{5}$$

where $\overline{A}$ is the *discretized state matrix* and $\overline{B}$ has a similar formula. The intuition behind discretization is to convert the (continuous-time) differential equation (3) to a discrete-time recurrence by simulating the equation. For example, a naive method would be Euler integration

$$\begin{aligned} h(k\Delta) &\approx h((k-1)\Delta) + \Delta h'((k-1)\Delta) \\ &= (I + \Delta A)h((k-1)\Delta) + (\Delta B)x((k-1)\Delta) \end{aligned}$$

which simplifies to the the equation (4) with $\overline{A} = I + \Delta A$. Equation (5) represents a more numerically accurate discretization, of which there are many well-studied variants.

An important property of SSMs is that the recurrence Eq. (4) is equivalent to a convolution by a particular kernel $\overline{K}$

$$\overline{K} = (C\overline{B}, C\overline{AB}, C\overline{A}^2\overline{B}, \dots) \qquad y = \overline{K} * x. \tag{6}$$

This can be derived simply by unrolling equation (4); more details are in prior work (Gu et al., 2021; 2022).

Thus, SSMs can be viewed as particular instantiations of CNNs and RNNs that inherit their efficiency at both training and inference and overcome their limitations. As an RNN, (4) is a special case of (2) where f and g are linear, giving it much simpler structure that avoids the optimization issues found in RNNs. As a CNN, (6) is a special case of (1) with an unbounded convolution kernel, overcoming the context size limitations of vanilla CNNs.

3.3. S4

S4 is a particular instantiation of SSM that parameterizes A as a *diagonal plus low-rank* (DPLR) matrix, $A = \Lambda + pq^*$ (Gu et al., 2022). This parameterization has two key properties. First, this is a structured representation that allows faster computation—S4 uses a special algorithm to compute the convolution kernel $\overline{K}$ (6) very quickly. Second, this parameterization includes certain special matrices called HiPPO matrices (Gu et al., 2020), which theoretically and empirically allow the SSM to capture long-range dependencies better. In particular, HiPPO specifies a special equation $h'(t) = Ah(t) + Bx(t)$ with closed formulas for A and B. This particular A matrix can be written in DPLR form, and S4 initializes its A and B matrices to these.

We note that building SASHIMI around S4 is similar in spirit to the use of linear oscillations for waveform generation in prior work, e.g. in the Neural Source Filter (Wang et al., 2019) or differentiable digital signal processing (DDSP) models (Engel et al., 2020). SSMs like S4, and consequently SASHIMI can be viewed as a generalization of models that rely on harmonic oscillation, and are able to directly learn the appropriate basis functions for raw waveforms.

4. Model

SASHIMI consists of two main components. First, S4 layers are the core component of our neural network architecture, to capture long context while being fast at both training and inference. We provide a simple improvement to S4 that addresses instability at generation time (Section 4.1). Second, SASHIMI connects stacks of S4 layers together in a simple multi-scale architecture (Section 4.2).

4.1. Stabilizing S4 for Recurrence

We use S4's representation and algorithm as a black box, with one technical improvement: we use the parameterization $\Lambda - pp^*$ instead of $\Lambda + pq^*$. This amounts to essentially tying the parameters p and q (and reversing a sign).

To justify our parameterization, we first note that it still satisfies the main properties of S4's representation (Section 3.3). First, this is a special case of a DPLR matrix, and can still use S4's algorithm for fast computation. Moreover, we show that the HiPPO matrices still satisfy this more restricted structure; in other words, we can still use the same initialization which is important to S4's performance.

Proposition 4.1. *All three HiPPO matrices from (Gu et al., 2020) are unitarily equivalent to a matrix of the form $A = \Lambda - pp^*$ for diagonal Λ and $p \in \mathbb{R}^{N \times r}$ for $r = 1$ or $r = 2$. Furthermore, all entries of Λ have real part 0 (for HiPPO-LegT and HiPPO-LagT) or $-\frac{1}{2}$ (for HiPPO-LegS).*

Next, we discuss how this parameterization makes S4 stable. The high-level idea is that stability of SSMs involves the spectrum of the state matrix A, which is more easily controlled because $-pp^*$ is a negative semidefinite matrix (i.e., we know the signs of its spectrum).

Definition 4.2. A *Hurwitz matrix* A is one where every eigenvalue has negative real part.

Hurwitz matrices are also called stable matrices, because they imply that the SSM (3) is asymptotically stable. In the context of discrete time SSMs, we can easily see why A needs to be a Hurwitz matrix from first principles with the following simple observations.

First, unrolling the RNN mode (equation (4)) involves powering up $\overline{A}$ repeatedly, which is stable if and only if all eigenvalues of $\overline{A}$ lie inside or on the unit disk. Second, the transformation (5) maps the complex left half plane (i.e. negative real part) to the complex unit disk. Therefore computing the RNN mode of an SSM (e.g. in order to generate autoregressively) requires A to be a Hurwitz matrix.

However, controlling the spectrum of a general DPLR matrix is difficult; empirically, we found that S4 matrices generally became non-Hurwitz after training. We remark that this stability issue only arises when using S4 during autoregressive generation, because S4's convolutional mode during training does not involve powering up $\overline{A}$ and thus does not require a Hurwitz matrix. Our reparameterization makes controlling the spectrum of $\overline{A}$ easier. Note that the goal of this reparameterization is only to ensure stability, and we do not expect it to impact model performance, which we verify in Section 5.2.

Proposition 4.3. *A matrix $A = \Lambda - pp^*$ is Hurwitz if all entries of Λ have negative real part.*

Proof. We first observe that if $A + A^*$ is negative semidefinite (NSD), then A is Hurwitz. This follows because $0 > v^*(A + A^*)v = (v^*Av) + (v^*Av)^* = 2\mathfrak{Re}(v^*Av) = 2\lambda$ for any (unit length) eigenpair (λ, v) of A.

Next, note that the condition implies that $\Lambda + \Lambda^*$ is NSD (it is a real diagonal matrix with non-positive entries). Since the matrix $-pp^*$ is also NSD, then so is $A + A^*$. □

Proposition 4.3 implies that with our tied reparameterization of S4, controlling the spectrum of the learned A matrix becomes simply controlling the the diagonal portion Λ. This is a far easier problem than controlling a general DPLR matrix, and can be enforced by regularization or reparameteration (e.g. run its entries through an $\exp$ function). In practice, we found that not restricting Λ and letting it learn freely led to stable trained solutions.

Table 1. Summary of music and speech datasets used for unconditional AR generation experiments.

Category	Dataset	Total Duration	Chunk Length	Sampling Rate	Quantization	Splits (train-val-test)
Music	Beethoven	10 hours	8s	16kHz	8-bit linear	Mehri et al. (2017)
Music	YouTubeMix	4 hours	8s	16kHz	8-bit mu-law	88% − 6% − 6%
Speech	SC09	5.3 hours	1s	16kHz	8-bit mu-law	Warden (2018)

4.2. SaShiMi Architecture

Figure 1 illustrates the complete SaShiMi architecture.

S4 Block. SaShiMi is built around repeated deep neural network blocks containing our modified S4 layers, following the same original S4 model. Compared to Gu et al. (2022), we add additional pointwise linear layers after the S4 layer in the style of the *feed-forward network* in Transformers or the *inverted bottleneck layer* in CNNs (Liu et al., 2022). Model details are in Appendix A.

Multi-scale Architecture. SaShiMi uses a simple architecture for autoregressive generation that consolidates information from the raw input signal at multiple resolutions. The SaShiMi architecture consists of multiple tiers, with each tier composed of a stack of residual S4 blocks. The top tier processes the raw audio waveform at its original sampling rate, while lower tiers process downsampled versions of the input signal. The output of lower tiers is upsampled and combined with the input to the tier above it in order to provide a stronger conditioning signal. This architecture is inspired by related neural network architectures for AR modeling that incorporate multi-scale characteristics such as SampleRNN and PixelCNN++ (Salimans et al., 2017).

The pooling is accomplished by simple reshaping and linear operations. Concretely, an input sequence $x \in \mathbb{R}^{T\times H}$ with context length T and hidden dimension size H is transformed through these shapes:

$$(\textbf{Down-pool})(T, H) \xrightarrow{\text{reshape}} (T/p, p\cdot H) \xrightarrow{\text{linear}} (T/p, q\cdot H)$$

$$(\textbf{Up-pool})(T, H) \xrightarrow{\text{linear}} (T, p\cdot H/q) \xrightarrow{\text{reshape}} (T\cdot p, H/q).$$

Here, p is the *pooling factor* and q is an *expansion factor* that increases the hidden dimension while pooling. In our experiments, we always fix $p = 4, q = 2$ and use a total of just two pooling layers (three tiers).

We additionally note that in AR settings, the up-pooling layers must be shifted by a time step to ensure causality.

Bidirectional S4. Like RNNs, SSMs are causal with an innate time dimension (equation (3)). For non-autoregressive tasks, we consider a simple variant of S4 that is bidirectional. We simply pass the input sequence through an S4 layer, and also reverse it and pass it through an independent second S4 layer. These outputs are concatenated and passed through a positionwise linear layer as in the standard S4 block.

$$y = \mathsf{Linear}(\mathsf{Concat}(\mathsf{S4}(x), \mathsf{rev}(\mathsf{S4}(\mathsf{rev}(x)))))$$

Table 2. Results on AR modeling of Beethoven, a benchmark task from Mehri et al. (2017)—SaShiMi outperforms all baselines while training faster.

Model	Context	NLL	@200K steps	@10 hours
SampleRNN*	1024	1.076	–	–
WaveNet*	4092	1.464	–	–
SampleRNN†	1024	1.125	1.125	1.125
WaveNet†	4092	1.032	1.088	1.352
SaShiMi	**128000**	**0.946**	**1.007**	**1.095**

*Reported in Mehri et al. (2017) †Our replication

Table 3. Effect of context length on the performance of SaShiMi on Beethoven, controlling for computation and sample efficiency. SaShiMi is able to leverage information from longer contexts.

Context Size	Batch Size	NLL	
		200K steps	10 hours
1 second	8	1.364	1.433
2 seconds	4	1.229	1.298
4 seconds	2	1.120	1.234
8 seconds	1	**1.007**	**1.095**

We show that bidirectional S4 outperforms causal S4 when autoregression is not required (Section 5.3).

5. Experiments

We evaluate SaShiMi on several benchmark audio generation and unconditional speech generation tasks in both AR and non-AR settings, validating that SaShiMi generates more globally coherent waveforms than baselines while having higher computational and sample efficiency.

Baselines. We compare SaShiMi to the leading AR models for unconditional waveform generation, SampleRNN and WaveNet. In Section 5.3, we show that SaShiMi can also improve non-AR models.

Datasets. We evaluate SaShiMi on datasets spanning music and speech generation (Table 1).

- **Beethoven.** A benchmark music dataset (Mehri et al., 2017), consisting of Beethoven's piano sonatas.

- **YouTubeMix.** Another piano music dataset (DeepSound, 2017) with higher-quality recordings than Beethoven.

Table 4. Negative log-likelihoods and mean opinion scores on YouTubeMix. As suggested by Dieleman et al. (2018b), we encourage readers to form their own opinions by referring to the sound examples in our supplementary material.

Model	Test NLL	MOS (fidelity)	MOS (musicality)
SampleRNN	1.723	**2.98 ± 0.08**	1.82 ± 0.08
WaveNet	1.449	2.91 ± 0.08	2.71 ± 0.08
SaShiMi	**1.294**	2.84 ± 0.09	**3.11 ± 0.09**
Dataset	-	3.76 ± 0.08	4.59 ± 0.07

- **SC09.** A benchmark speech dataset (Donahue et al., 2019), consisting of 1-second recordings of the digits "zero" through "nine" spoken by many different speakers.

All datasets are quantized using 8-bit quantization, either linear or μ-law, depending on prior work. Each dataset is divided into non-overlapping chunks; the SampleRNN baseline is trained using TBPTT, while WaveNet and SaShiMi are trained on entire chunks. All models are trained to predict the negative log-likelihood (NLL) of individual audio samples; results are reported in base 2, also known as bits per byte (BPB) because of the one-byte-per-sample quantization. All datasets were sampled at a rate of 16kHz. Table 1 summarizes characteristics of the datasets and processing.

5.1. Unbounded Music Generation

Because music audio is not constrained in length, AR models are a natural approach for music generation, since they can generate samples longer than the context windows they were trained on. We validate that SaShiMi can leverage longer contexts to perform music waveform generation more effectively than baseline AR methods.

We follow the setting of Mehri et al. (2017) for the Beethoven dataset. Table 2 reports results found in prior work, as well as our reproductions. In fact, our WaveNet baseline is much stronger than the one implemented in prior work. SaShiMi substantially improves the test NLL by 0.09 BPB compared to the best baseline. Table 3 ablates the context length used in training, showing that SaShiMi significantly benefits from seeing longer contexts, and is able to effectively leverage extremely long contexts (over 100k steps) when predicting next samples.

Next, we evaluate all baselines on YouTubeMix. Table 4 shows that SaShiMi substantially outperforms SampleRNN and WaveNet on NLL. Following Dieleman et al. (2018b) (protocol in Appendix C.4), we measured mean opinion scores (MOS) for audio fidelity and musicality for 16s samples generated by each method (longer than the training context). All methods have similar fidelity, but SaShiMi substantially improves musicality by around 0.40 points, validating that it can generate long samples more coherently than other methods.

Figure 2. **(Sample Efficiency)** Plot of validation NLL (in bits) vs. wall clock time (hours) on the SC09 dataset.

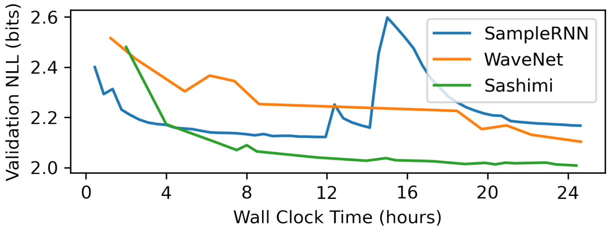


Figure 2 shows that SaShiMi trains stably and more efficiently than baselines in wall clock time. Appendix B, Figure 5 also analyzes the peak throughput of different AR models as a function of batch size.

Table 5. Architectural ablations and efficiency tradeoffs on YouTubeMix. (*Top*) AR models and baselines at different sizes. (*Bottom*) Ablating the pooling layers of SaShiMi.

Model	NLL	Time/epoch	Throughput	Params
SampleRNN−2 tier	1.762	800s	112K samples/s	51.85M
SampleRNN−3 tier	1.723	850s	116K samples/s	35.03M
WaveNet−512	1.467	1000s	185K samples/s	2.67M
WaveNet−1024	1.449	1435s	182K samples/s	4.24M
SaShiMi−2 layers	1.446	205s	596K samples/s	1.29M
SaShiMi−4 layers	1.341	340s	316K samples/s	2.21M
SaShiMi−6 layers	1.315	675s	218K samples/s	3.13M
SaShiMi−8 layers	1.294	875s	129K samples/s	4.05M
Isotropic S4−4 layers	1.429	1900s	144K samples/s	2.83M
Isotropic S4−8 layers	1.524	3700s	72K samples/s	5.53M

5.2. Model ablations: Slicing the SaShiMi

We validate our technical improvements and ablate SaShiMi's architecture.

Stabilizing S4. We consider how different parameterizations of S4's representation affect downstream performance (Section 4.1). Recall that S4 uses a special matrix $A = \Lambda + pq^*$ specified by HiPPO, which theoretically captures long-range dependencies (Section 3.3). We ablate various parameterizations of a small SaShiMi model (2 layers, 500 epochs on YouTubeMix). Learning A yields consistent improvements, but becomes unstable at generation. Our reparameterization allows A to be learned while preserving stability, agreeing with the analysis in Section 4.1. A visual illustration of the spectral radii of the learned $\overline{A}$ in the new parameterization is provided in Figure 3.

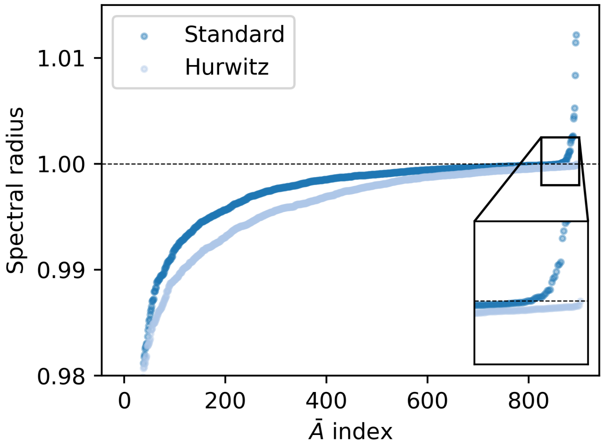
Learned	Frozen	NLL	Stable generation
−	$\Lambda + pq^*$	1.445	✓
$\Lambda + pq^*$	−	1.420	✗
$\Lambda - pp^*$	−	1.419	✓

Multi-scale architecture. We investigate the effect of

Table 6. (**SC09**) Automated metrics and human opinion scores. (*Top*) SaShiMi is the first AR model that can unconditionally generate high quality samples on this challenging dataset of fixed-length speech clips with highly variable characteristics. (*Middle*) As a flexible architecture for general waveform modeling, SaShiMi sets a true state-of-the-art when combined with a recent diffusion probabilistic model.

Model	Params	NLL	FID ↓	IS ↑	mIS ↑	AM ↓	Human (κ) Agreement	MOS		
								Quality	Intelligibility	Diversity
SampleRNN	35.0M	2.042	8.96	1.71	3.02	1.76	0.321	1.18 ± 0.04	1.37 ± 0.02	2.26 ± 0.10
WaveNet	4.2M	1.925	5.08	2.27	5.80	1.47	0.408	1.59 ± 0.06	1.72 ± 0.03	2.70 ± 0.11
SaShiMi	4.1M	**1.891**	**1.99**	**4.12**	**24.57**	**0.90**	**0.832**	$\mathbf{3.29 \pm 0.07}$	$\mathbf{3.53 \pm 0.04}$	$\mathbf{3.26 \pm 0.09}$
WaveGAN	19.1M	-	2.03	4.90	36.10	0.80	0.840	2.98 ± 0.07	3.27 ± 0.04	3.25 ± 0.09
DiffWave	24.1M	-	1.92	5.26	51.21	0.68	0.917	4.03 ± 0.06	4.15 ± 0.03	$\mathbf{3.45 \pm 0.09}$
w/ SaShiMi	23.0M	-	**1.42**	**5.94**	**69.17**	**0.59**	**0.953**	$\mathbf{4.20 \pm 0.06}$	$\mathbf{4.33 \pm 0.03}$	3.28 ± 0.11
Train	-	-	0.00	8.56	292.5	0.16	0.921	4.04 ± 0.06	4.27 ± 0.03	3.59 ± 0.09
Test	-	-	0.02	8.33	257.6	0.19				

Figure 3. **(S4 Stability)** Comparison of spectral radii for all $\overline{A}$ matrices in a SaShiMi model trained with different S4 parameterizations. The instability in the standard S4 parameterization is solved by our Hurwitz parameterization.



SaShiMi's architecture (Section 4.2) against isotropic S4 layers on YouTubeMix. Controlling for parameter counts, adding pooling in SaShiMi leads to substantial improvements in computation and modeling (Table 5, Bottom).

Efficiency tradeoffs. We ablate different sizes of the SaShiMi model on YouTubeMix to show its performance tradeoffs along different axes.

Table 5 (Top) shows that a single SaShiMi model simultaneously outperforms all baselines on quality (NLL) and computation at both training and inference, with a model more than 3× smaller. We note that for WaveNet, we use a fast, cached implementation for generation (Paine et al., 2016). Moreover, SaShiMi improves monotonically with depth, suggesting that quality can be further improved at the cost of additional computation.

5.3. Unconditional Speech Generation

The SC09 spoken digits dataset is a challenging unconditional speech generation benchmark, as it contains several axes of variation (words, speakers, microphones, alignments). Unlike the music setting (Section 5.1), SC09 contains audio of *bounded* length (1-second utterances). To date, AR waveform models trained on this benchmark have yet to generate spoken digits which are consistently intelligible to humans.[2] In contrast, non-AR approaches are capable of achieving global coherence on this dataset, as first demonstrated by WaveGAN (Donahue et al., 2019).

Although our primary focus thus far has been the challenging testbed of AR waveform modeling, SaShiMi can also be used as a flexible neural network architecture for audio generation more broadly. We demonstrate this potential by integrating SaShiMi into DiffWave (Kong et al., 2021), a diffusion-based method for non-AR waveform generation which represents the current state-of-the-art for SC09. DiffWave uses the original WaveNet architecture as its backbone—here, we simply replace WaveNet with a SaShiMi model containing a similar number of parameters.

We compare SaShiMi to strong baselines on SC09 in both the AR and non-AR (via DiffWave) settings by measuring several standard quantitative and qualitative metrics such as Frechét Inception Distance (FID) and Inception Score (IS) (Appendix C.3). We also conduct a qualitative evaluation where we ask several annotators to label the generated digits and then compute their inter-annotator agreement. Additionally, as in Donahue et al. (2019), we ask annotators for their subjective opinions on overall audio quality, intelligibility, and speaker diversity, and report MOS for each axis. Results for all models appear in Table 6.

[2] While AR waveform models can produce intelligible speech in the context of TTS systems, this capability requires conditioning on rich intermediaries like spectrograms or linguistic features.

Autoregressive. SASHIMI substantially outperforms other AR waveform models on all metrics, and achieves 2× higher MOS for both quality and intelligibility. Moreover, annotators agree on labels for samples from SASHIMI far more often than they do for samples from other AR models, suggesting that SASHIMI generates waveforms that are more globally coherent on average than prior work. Finally, SASHIMI achieves higher MOS on all axes compared to WaveGAN while using more than 4× fewer parameters.

Non-autoregressive. Integrating SASHIMI into DiffWave substantially improves performance on all metrics compared to its WaveNet-based counterpart, and achieves a new overall state-of-the-art performance on all quantitative and qualitative metrics on SC09. We note that this result involved *zero tuning* of the model or training parameters (e.g. diffusion steps or optimizer hyperparameters) (Appendix C.2). This suggests that SASHIMI could be useful not only for AR waveform modeling but also as a new drop-in architecture for many audio generation systems which currently depend on WaveNet (see Section 2).

We additionally conduct several ablation studies on our hybrid DiffWave and SASHIMI model, and compare performance earlier in training and with smaller models (Table 7). When paired with DiffWave, SASHIMI is much more sample efficient than WaveNet, matching the performance of the best WaveNet-based model with half as many training steps. Kong et al. (2021) also observed that DiffWave was extremely sensitive with a WaveNet backbone, performing poorly with smaller models and becoming unstable with larger ones. We show that, when using WaveNet, a small DiffWave model fails to model the dataset, however it works much more effectively when using SASHIMI. Finally, we ablate our non-causal relaxation, showing that this bidirectional version of SASHIMI performs much better than its unidirectional counterpart (as expected).

6. Discussion

Our results indicate that SASHIMI is a promising new architecture for modeling raw audio waveforms. When trained on music and speech datasets, SASHIMI generates waveforms that humans judge to be more musical and intelligible respectively compared to waveforms from previous architectures, indicating that audio generated by SASHIMI has a higher degree of global coherence. By leveraging the dual convolutional and recurrent forms of S4, SASHIMI is more computationally efficient than past architectures during both training and inference. Additionally, SASHIMI is consistently more sample efficient to train—it achieves better quantitative performance with fewer training steps. Finally, when used as a drop-in replacement for WaveNet, SASHIMI improved the performance of an existing state-of-the-art model for unconditional generation, indicating a potential for SASHIMI to create a ripple effect of improving audio generation more broadly.

Acknowledgments

We gratefully acknowledge the support of NIH under No. U54EB020405 (Mobilize), NSF under Nos. CCF1763315 (Beyond Sparsity), CCF1563078 (Volume to Velocity), and 1937301 (RTML); ARL under No. W911NF-21-2-0251 (Interactive Human-AI Teaming); ONR under No. N000141712266 (Unifying Weak Supervision); ONR N00014-20-1-2480: Understanding and Applying Non-Euclidean Geometry in Machine Learning; N000142012275 (NEPTUNE); NXP, Xilinx, LETI-CEA, Intel, IBM, Microsoft, NEC, Toshiba, TSMC, ARM, Hitachi, BASF, Accenture, Ericsson, Qualcomm, Analog Devices, Google Cloud, Salesforce, Total, the HAI-Google & AWS Cloud Credits for Research program, the Stanford Data Science Initiative (SDSI), and members of the Stanford DAWN project: Facebook, Google, and VMWare. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views, policies, or endorsements, either expressed or implied, of NIH, ONR, or the U.S. Government.

References

Bińkowski, M., Donahue, J., Dieleman, S., Clark, A., Elsen, E., Casagrande, N., Cobo, L. C., and Simonyan, K. High fidelity speech synthesis with adversarial networks. In *International Conference on Learning Representations*, 2020.

Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

Chen, N., Zhang, Y., Zen, H., Weiss, R. J., Norouzi, M., and Chan, W. Wavegrad: Estimating gradients for waveform generation. In *International Conference on Learning Representations*, 2021.

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

Dauphin, Y. N., Fan, A., Auli, M., and Grangier, D. Language modeling with gated convolutional networks. In

International conference on machine learning, pp. 933–941. PMLR, 2017.

DeepSound. Samplernn. `https://github.com/deepsound-project/samplernn-pytorch`, 2017.

Dhariwal, P., Jun, H., Payne, C., Kim, J. W., Radford, A., and Sutskever, I. Jukebox: A generative model for music. *arXiv preprint arXiv:2005.00341*, 2020.

Dieleman, S., van den Oord, A., and Simonyan, K. The challenge of realistic music generation: modelling raw audio at scale. *ArXiv*, abs/1806.10474, 2018a.

Dieleman, S., van den Oord, A., and Simonyan, K. The challenge of realistic music generation: modelling raw audio at scale. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 8000–8010, 2018b.

Donahue, C., McAuley, J., and Puckette, M. Adversarial audio synthesis. In *International Conference on Learning Representations*, 2019.

Engel, J., Resnick, C., Roberts, A., Dieleman, S., Norouzi, M., Eck, D., and Simonyan, K. Neural audio synthesis of musical notes with wavenet autoencoders. In *International Conference on Machine Learning*, pp. 1068–1077. PMLR, 2017.

Engel, J., Hantrakul, L., Gu, C., and Roberts, A. Ddsp: Differentiable digital signal processing. *arXiv preprint arXiv:2001.04643*, 2020.

Gu, A., Dao, T., Ermon, S., Rudra, A., and Ré, C. Hippo: Recurrent memory with optimal polynomial projections. *Advances in Neural Information Processing Systems*, 33, 2020.

Gu, A., Johnson, I., Goel, K., Saab, K., Dao, T., Rudra, A., and Ré, C. Combining recurrent, convolutional, and continuous-time models with the structured learnable linear state space layer. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Gu, A., Goel, K., and Ré, C. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*, 2022.

Gurumurthy, S., Kiran Sarvadevabhatla, R., and Venkatesh Babu, R. Deligan: Generative adversarial networks for diverse and limited data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 166–174, 2017.

Hendrycks, D. and Gimpel, K. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

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

Hochreiter, S., Bengio, Y., Frasconi, P., Schmidhuber, J., et al. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.

Jayaram, V. and Thickstun, J. Parallel and flexible sampling from autoregressive models via langevin dynamics. In *The International Conference on Machine Learning (ICML)*, 2021.

Kalchbrenner, N., Elsen, E., Simonyan, K., Noury, S., Casagrande, N., Lockhart, E., Stimberg, F., van den Oord, A., Dieleman, S., and Kavukcuoglu, K. Efficient neural audio synthesis. In *International Conference on Machine Learning*, pp. 2410–2419. PMLR, 2018.

Kim, S., Lee, S.-G., Song, J., Kim, J., and Yoon, S. Flowavenet: A generative flow for raw audio. In *International Conference on Machine Learning*, pp. 3370–3378. PMLR, 2019.

Kleijn, W. B., Lim, F. S., Luebs, A., Skoglund, J., Stimberg, F., Wang, Q., and Walters, T. C. Wavenet based low rate speech coding. In *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 676–680. IEEE, 2018.

Kong, Z., Ping, W., Huang, J., Zhao, K., and Catanzaro, B. Diffwave: A versatile diffusion model for audio synthesis. In *International Conference on Learning Representations*, 2021.

Kumar, K., Kumar, R., de Boissiere, T., Gestin, L., Teoh, W. Z., Sotelo, J., de Brébisson, A., Bengio, Y., and Courville, A. C. Melgan: Generative adversarial networks for conditional waveform synthesis. *Advances in Neural Information Processing Systems*, 32, 2019.

Lakhotia, K., Kharitonov, E., Hsu, W.-N., Adi, Y., Polyak, A., Bolte, B., Nguyen, T.-A., Copet, J., Baevski, A., Mohamed, A., et al. Generative spoken language modeling from raw audio. *arXiv preprint arXiv:2102.01192*, 2021.

Li, N., Liu, S., Liu, Y., Zhao, S., and Liu, M. Neural speech synthesis with transformer network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 6706–6713, 2019.

Liu, Z., Mao, H., Wu, C.-Y., Feichtenhofer, C., Darrell, T., and Xie, S. A convnet for the 2020s. *arXiv preprint arXiv:2201.03545*, 2022.

Mehri, S., Kumar, K., Gulrajani, I., Kumar, R., Jain, S., Sotelo, J., Courville, A., and Bengio, Y. Samplernn: An unconditional end-to-end neural audio generation model. In *International Conference on Learning Representations*, 2017.

Neekhara, P., Donahue, C., Puckette, M., Dubnov, S., and McAuley, J. Expediting tts synthesis with adversarial vocoding. In *INTERSPEECH*, 2019.

Paine, T. L., Khorrami, P., Chang, S., Zhang, Y., Ramachandran, P., Hasegawa-Johnson, M. A., and Huang, T. S. Fast wavenet generation algorithm. *arXiv preprint arXiv:1611.09482*, 2016.

Pascanu, R., Mikolov, T., and Bengio, Y. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pp. 1310–1318, 2013.

Peng, K., Ping, W., Song, Z., and Zhao, K. Non-autoregressive neural text-to-speech. In *International conference on machine learning*, pp. 7586–7598. PMLR, 2020.

Ping, W., Peng, K., and Chen, J. Clarinet: Parallel wave generation in end-to-end text-to-speech. In *International Conference on Learning Representations*, 2019.

Ping, W., Peng, K., Zhao, K., and Song, Z. Waveflow: A compact flow-based model for raw audio. In *International Conference on Machine Learning*, pp. 7706–7716. PMLR, 2020.

Prenger, R., Valle, R., and Catanzaro, B. Waveglow: A flow-based generative network for speech synthesis. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3617–3621. IEEE, 2019.

Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., and Sutskever, I. Zero-shot text-to-image generation. *arXiv preprint arXiv:2102.12092*, 2021.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. Improved techniques for training gans. *Advances in neural information processing systems*, 29:2234–2242, 2016.

Salimans, T., Karpathy, A., Chen, X., and Kingma, D. P. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. In *International Conference on Learning Representations*, 2017.

Shen, J., Pang, R., Weiss, R. J., Schuster, M., Jaitly, N., Yang, Z., Chen, Z., Zhang, Y., Wang, Y., Skerrv-Ryan, R., et al. Natural tts synthesis by conditioning wavenet on mel spectrogram predictions. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4779–4783. IEEE, 2018.

van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

van den Oord, A., Vinyals, O., and Kavukcuoglu, K. Neural discrete representation learning. *arXiv preprint arXiv:1711.00937*, 2017.

van den Oord, A., Li, Y., Babuschkin, I., Simonyan, K., Vinyals, O., Kavukcuoglu, K., Driessche, G., Lockhart, E., Cobo, L., Stimberg, F., et al. Parallel wavenet: Fast high-fidelity speech synthesis. In *International conference on machine learning*, pp. 3918–3926. PMLR, 2018.

Wang, X., Takaki, S., and Yamagishi, J. Neural source-filter-based waveform model for statistical parametric speech synthesis. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5916–5920. IEEE, 2019.

Warden, P. Speech commands: A dataset for limited-vocabulary speech recognition. *ArXiv*, abs/1804.03209, 2018.

Xie, S., Girshick, R., Dollár, P., Tu, Z., and He, K. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1492–1500, 2017.

Yamamoto, R., Song, E., and Kim, J.-M. Parallel wavegan: A fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6199–6203. IEEE, 2020.

Zhou, Z., Cai, H., Rong, S., Song, Y., Ren, K., Zhang, W., Wang, J., and Yu, Y. Activation maximization generative adversarial nets. In *International Conference on Learning Representations*, 2018.

A. Model Details

A.1. S4 Stability

We prove Proposition 4.1. We build off the S4 representation of HiPPO matrices, using their decomposition as a normal plus low-rank matrix which implies that they are unitarily similar to a diagonal plus low-rank matrix. Then we show that the low-rank portion of this decomposition is in fact negative semidefinite, while the diagonal portion has non-positive real part.

Proof of Proposition 4.1. We consider the diagonal plus low-rank decompositions shown in Gu et al. (2022) of the three original HiPPO matrices Gu et al. (2020), and show that the low-rank portions are in fact negative semidefinite.

HiPPO-LagT. The family of generalized HiPPO-LagT matrices are defined by

$$\boldsymbol{A}_{nk} = \begin{cases} 0 & n < k \\ -\frac{1}{2} - \beta & n = k \\ -1 & n > k \end{cases}$$

for $0 \leq \beta \leq \frac{1}{2}$, with the main HiPPO-LagT matrix having $\beta = 0$.

It can be decomposed as

$$\boldsymbol{A} = -\begin{bmatrix} \frac{1}{2}+\beta & & & & \cdots \\ 1 & \frac{1}{2}+\beta & & & \\ 1 & 1 & \frac{1}{2}+\beta & & \\ 1 & 1 & 1 & \frac{1}{2}+\beta & \\ \vdots & & & & \ddots \end{bmatrix} = -\beta I - \begin{bmatrix} & -\frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & \\ \frac{1}{2} & & -\frac{1}{2} & -\frac{1}{2} & \cdots \\ \frac{1}{2} & \frac{1}{2} & & -\frac{1}{2} & \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & & \\ \vdots & & & & \ddots \end{bmatrix} - \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 & \cdots \\ 1 & 1 & 1 & 1 & \\ 1 & 1 & 1 & 1 & \\ 1 & 1 & 1 & 1 & \\ \vdots & & & & \ddots \end{bmatrix}.$$

The first term is skew-symmetric, which is unitarily similar to a (complex) diagonal matrix with pure imaginary eigenvalues (i.e., real part 0). The second matrix can be factored as pp^* for $p = 2^{-1/2}\begin{bmatrix}1 & \cdots & 1\end{bmatrix}^*$. Thus the whole matrix A is unitarily similar to a matrix $\Lambda - pp^*$ where the eigenvalues of Λ have real part between $-\frac{1}{2}$ and 0.

HiPPO-LegS. The HiPPO-LegS matrix is defined as

$$\boldsymbol{A}_{nk} = -\begin{cases} (2n+1)^{1/2}(2k+1)^{1/2} & \text{if } n > k \\ n+1 & \text{if } n = k \\ 0 & \text{if } n < k \end{cases}.$$

It can be decomposed as Adding $\frac{1}{2}(2n+1)^{1/2}(2k+1)^{1/2}$ to the whole matrix gives

$$-\frac{1}{2}I - S - pp^*$$

$$S_{nk} = \begin{cases} \frac{1}{2}(2n+1)^{1/2}(2k+1)^{1/2} & \text{if } n > k \\ 0 & \text{if } n = k \\ -\frac{1}{2}(2n+1)^{1/2}(2k+1)^{1/2} & \text{if } n < k \end{cases}$$

$$p_n = (n + \frac{1}{2})^{1/2}$$

Note that S is skew-symmetric. Therefore A is unitarily similar to a matrix $\Lambda - pp^*$ where the eigenvalues of Λ have real part $-\frac{1}{2}$.

HiPPO-LegT.

Up to the diagonal scaling, the LegT matrix is

$$\boldsymbol{A} = -\begin{bmatrix} 1 & -1 & 1 & -1 & \cdots \\ 1 & 1 & -1 & 1 & \\ 1 & 1 & 1 & -1 & \\ 1 & 1 & 1 & 1 & \\ \vdots & & & & \ddots \end{bmatrix} = -\begin{bmatrix} 0 & -1 & 0 & -1 & \cdots \\ 1 & 0 & -1 & 0 & \\ 0 & 1 & 0 & -1 & \\ 1 & 0 & 1 & 0 & \\ \vdots & & & & \ddots \end{bmatrix} - \begin{bmatrix} 1 & 0 & 1 & 0 & \cdots \\ 0 & 1 & 0 & 1 & \\ 1 & 0 & 1 & 0 & \\ 0 & 1 & 0 & 1 & \\ \vdots & & & & \ddots \end{bmatrix}.$$

The first term is skew-symmetric and the second term can be written as pp^* for

$$p = \begin{bmatrix} 1 & 0 & 1 & 0 & \cdots \\ 0 & 1 & 0 & 1 & \cdots \end{bmatrix}^\top$$

□

A.2. Model Architecture

S4 Block Details The first portion of the S4 block is the same as the one used in Gu et al. (2022).

$$\begin{aligned} y &= x \\ y &= \mathsf{LayerNorm}(y) \\ y &= \mathsf{S4}(y) \\ y &= \phi(y) \\ y &= Wy + b \\ y &= x + y \end{aligned}$$

Here ϕ is a non-linear activation function, chosen to be GELU (Hendrycks & Gimpel, 2016) in our implementation. Note that all operations aside from the S4 layer are *position-wise* (with respect to the time or sequence dimension).

These operations are followed by more position-wise operations, which are standard in other deep neural networks such as Transformers (where it is called the feed-forward network) and CNNs (where it is called the inverted bottleneck layer).

$$\begin{aligned} y &= x \\ y &= \mathsf{LayerNorm}(y) \\ y &= W_1 y + b_1 \\ y &= \phi(y) \\ y &= W_2 y + b_2 \\ y &= x + y \end{aligned}$$

Here $W_1 \in \mathbb{R}^{d\times ed}$ and $W_2 \in \mathbb{R}^{ed\times d}$, where e is an expansion factor. We fix $e = 2$ in all our experiments.

B. Additional Results

We provide details of ablations, including architecture ablations and efficiency benchmarking.

B.0.1. YouTubeMix

We conduct architectural ablations and efficiency benchmarking for all baselines on the YouTubeMix dataset.

Architectures. SampleRNN-2 and SampleRNN-3 correspond to the 2- and 3-tier models described in Appendix C.2 respectively. WaveNet-512 and WaveNet-1024 refer to models with 512 and 1024 skip channels respectively with all other details fixed as described in Appendix C.2. SaShiMi-$\{2, 4, 6, 8\}$ consist of the indicated number of S4 blocks in each tier of the architecture, with all other details being the same.

Isotropic S4. We also include an isotropic S4 model to ablate the effect of pooling in SaShiMi. Isotropic S4 can be viewed as SaShiMi without any pooling (i.e. no additional tiers aside from the top tier). We note that due to larger memory usage for these models, we use a sequence length of 4s for the 4 layer isotropic model, and a sequence length of 2s for the 8 layer isotropic model (both with batch size 1), highlighting an additional disadvantage in memory efficiency.

Throughput Benchmarking. To measure peak throughput, we track the time taken by models to generate 1000 samples at batch sizes that vary from 1 to 8192 in powers of 2. The throughput is the total number of samples generated by a model in 1 second. Figure 4 shows the results of this study in more detail for each method.

Diffusion model ablations. Table 7 reports results for the ablations described in Section 5.3. Experimental details are provided in Appendix C.2.

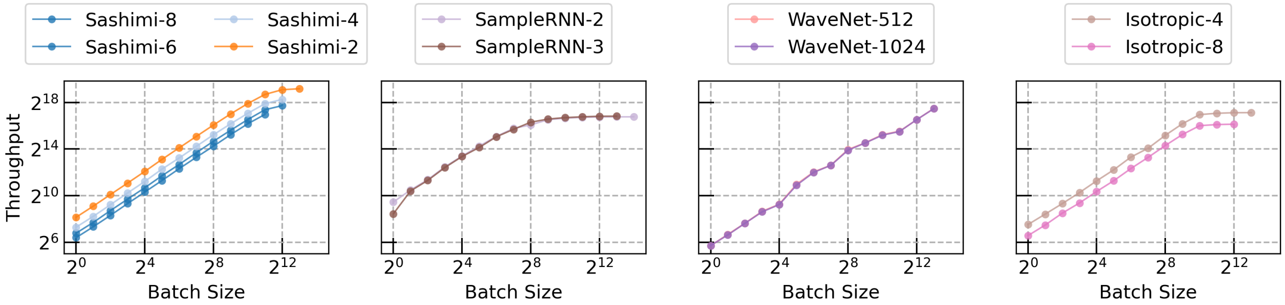


Figure 4. Log-log plot of throughput vs. batch size. Throughput scales near linearly for SASHIMI. By contrast, SampleRNN throughput peaks at smaller batch sizes, while WaveNet shows sublinear scaling with throughput degradation at some batch sizes. Isotropic variants have far lower throughput than SASHIMI.

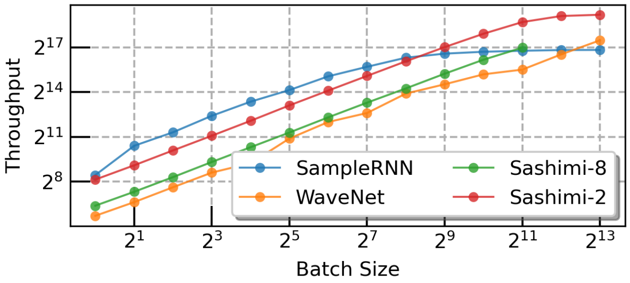


Figure 5. Log-log plot of throughput vs. batch size. SASHIMI-2 improves peak throughput over WaveNet and SampleRNN by 3× and 5× respectively.

C. Experiment Details

We include experimental details, including dataset preparation, hyperparameters for all methods, details of ablations as well as descriptions of automated and human evaluation metrics below.

C.1. Datasets

A summary of dataset information can be found in Table 1. Across all datasets, audio waveforms are preprocessed to 16kHz using `torchaudio`.

Beethoven. The dataset consists of recordings of Beethoven's 32 piano sonatas. We use the version of the dataset shared by Mehri et al. (2017), which can be found here. Since we compare to numbers reported by Mehri et al. (2017), we use linear quantization for all (and only) Beethoven experiments. We attempt to match the splits used by the original paper by reference to the code provided here.

YouTubeMix. A 4 hour dataset of piano music taken from `https://www.youtube.com/watch?v=EhO_MrRfftU`. We split the audio track into `.wav` files of 1 minute each, and use the first 88% files for training, next 6% files for validation and final 6% files for testing.

SC09. The Speech Commands dataset (Warden, 2018) contains many spoken words by thousands of speakers under various recording conditions including some very noisy environments. Following prior work (Donahue et al., 2019; Kong et al., 2021) we use the subset that contains spoken digits "zero" through "nine". This SC09 dataset contains 31,158 training utterances (8.7 hours in total) by 2,032 speakers, where each audio has length 1 second sampled at 16kHz. the generative models need to model them without any conditional information.

The datasets we used can be found on Huggingface datasets: Beethoven, YouTubeMix, SC09.

Table 7. (**SC09 diffusion models**.) Beyond AR, SaShiMi can be flexibly combined with other generative modeling approaches, improving on the state-of-the-art DiffWave model by simply replacing the architecture. (*Top*) A parameter-matched SaShiMi architecture with *no tuning* outperforms the best DiffWave model. (*Middle*) SaShiMi is consistently better than WaveNet at all stages of training; a model trained on half the samples matches the best DiffWave model. (*Bottom*) The WaveNet backbone is extremely sensitive to architecture parameters such as size and dilation schedule; a small model fails to learn. We also ablate the bidirectional S4 layer, which outperforms the unidirectional one.

Architecture	Params	Training Steps	FID ↓	IS ↑	mIS ↑	AM ↓	NDB ↓	Human (κ) Agreement	MOS		
									Quality	Intelligibility	Diversity
SaShiMi	23.0M	800k	1.42	5.94	69.17	0.59	0.88	0.953	4.20 ± 0.06	4.33 ± 0.03	3.28 ± 0.11
WaveNet	24.1M	1000k	1.92	5.26	51.21	0.68	0.88	0.917	4.03 ± 0.06	4.15 ± 0.03	3.45 ± 0.09
SaShiMi	23.0M	500k	2.08	5.68	51.10	0.66	0.76	0.923	3.99 ± 0.06	4.13 ± 0.03	3.38 ± 0.10
WaveNet	24.1M	500k	2.25	4.68	34.55	0.80	0.90	0.848	3.53 ± 0.07	3.69 ± 0.03	3.30 ± 0.08
SaShiMi (uni.)	7.1M	500k	2.70	3.62	17.96	1.03	0.90	0.829	3.08 ± 0.07	3.29 ± 0.04	3.26 ± 0.08
SaShiMi	7.5M	500k	1.70	5.00	40.27	0.72	0.90	0.934	3.83 ± 0.07	4.00 ± 0.03	3.34 ± 0.09
WaveNet	6.8M	500k	4.53	2.80	9.02	1.30	0.94	0.446	1.85 ± 0.08	1.90 ± 0.03	3.03 ± 0.10

C.2. Models and Training Details

For all datasets, SaShiMi, SampleRNN and WaveNet receive 8-bit quantized inputs. During training, we use no additional data augmentation of any kind. We summarize the hyperparameters used and any sweeps performed for each method below.

C.2.1. Details for Autoregressive Models

All methods in the AR setting were trained on single V100 GPU machines.

SaShiMi. We adapt the S4 implementation provided by Gu et al. (2022) to incorporate parameter tying for pq^*. For simplicity, we do not train the low-rank term pp^*, timescale dt and the B matrix throughout our experiments, and let Λ be trained freely. We find that this is actually stable, but leads to a small degradation in performance compared to the original S4 parameterization. Rerunning all experiments with our updated Hurwitz parameterization–which constrains the real part of the entries of Λ using an $\exp$ function–would be expensive, but would improve performance. For all datasets, we use feature expansion of $2\times$ when pooling, and use a feedforward dimension of $2\times$ the model dimension in all inverted bottlenecks in the model. We use a model dimension of 64. For S4 parameters, we only train Λ and C with the recommended learning rate of 0.001, and freeze all other parameters for simplicity (including pp^*, B, dt). We train with $4\times \rightarrow 4\times$ pooling for all datasets, with 8 S4 blocks per tier.

On Beethoven, we learn separate Λ matrices for each SSM in the S4 block, while we use parameter tying for Λ within an S4 block on the other datasets. On SC09, we found that swapping in a gated linear unit (GLU) (Dauphin et al., 2017) in the S4 block improved NLL as well as sample quality.

We train SaShiMi on Beethoven for 1M steps, YouTubeMix for 600K steps, SC09 for 1.1M steps.

SampleRNN. We adapt an open-source PyTorch implementation of the SampleRNN backbone, and train it using truncated backpropagation through time (TBPTT) with a chunk size of 1024. We train 2 variants of SampleRNN: a 3-tier model with frame sizes $8, 2, 2$ with 1 RNN per layer to match the 3-tier RNN from Mehri et al. (2017) and a 2-tier model with frame sizes $16, 4$ with 2 RNNs per layer that we found had stronger performance in our replication (than the 2-tier model from Mehri et al. (2017)). For the recurrent layer, we use a standard GRU model with orthogonal weight initialization following Mehri et al. (2017), with hidden dimension 1024 and feedforward dimension 256 between tiers. We also use weight normalization as recommended by Mehri et al. (2017).

We train SampleRNN on Beethoven for 150K steps, YouTubeMix for 200K steps, SC09 for 300K steps. We found that SampleRNN could be quite unstable, improving steadily and then suddenly diverging. It also appeared to be better suited to training with linear rather than mu-law quantization.

WaveNet. We adapt an open-source PyTorch implementation of the WaveNet backbone, trained using standard backpropagation. We set the number of residual channels to 64, dilation channels to 64, end channels to 512. We use 4 blocks of dilation with 10 layers each, with a kernel size of 2. Across all datasets, we sweep the number of skip channels among $\{512, 1024\}$. For optimization, we use the AdamW optimizer, with a learning rate of 0.001 and a plateau learning rate scheduler that has a patience of 5 on the validation NLL. During training, we use a batch size of 1 and pad each batch on the left with zeros equal to the size of the receptive field of the WaveNet model (4093 in our case).

We train WaveNet on Beethoven for 400K steps, YouTubeMix for 200K steps, SC09 for 500K steps.

C.2.2. Details for Diffusion Models

All diffusion models were trained on 8-GPU A100 machines.

DiffWave. We adapt an open-source PyTorch implementation of the DiffWave model. The DiffWave baseline in Table 6 is the unconditional SC09 model reported in Kong et al. (2021), which uses a 36 layer WaveNet backbone with dilation cycle $[1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048]$ and hidden dimension 256, a linear diffusion schedule $\beta_t \in [1 \times 10^4, 0.02]$ with $T = 200$ steps, and the Adam optimizer with learning rate 2×10^{-4}. The small DiffWave model reported in Table 7 has 30 layers with dilation cycle $[1, 2, 4, 8, 16, 32, 64, 128, 256, 512]$ and hidden dimension 128.

DiffWave with SaShiMi. Our large SaShiMi model has hidden dimension 128 and 6 S4 blocks per tier with the standard two pooling layers with pooling factor 4 and expansion factor 2 (Section 4.2). We additionally have S4 layers in the down-blocks in addition to the up-blocks of Figure 1. Our small SaShiMi model (Table 7) reduces the hidden dimension to 64. These architectures were chosen to roughly parameter match the DiffWave model. While DiffWave experimented with other architectures such as deep and thin WaveNets or different dilation cycles (Kong et al., 2021), we only ran a single SaShiMi model of each size. All optimization and diffusion hyperparameters were kept the same, with the exception that we manually decayed the learning rate of the large SaShiMi model at 500K steps as it had saturated and the model had already caught up to the best DiffWave model (Table 7).

C.3. Automated Evaluations

NLL. We report negative log-likelihood (NLL) scores for all AR models in bits, on the test set of the respective datasets. To evaluate NLL, we follow the same protocol as we would for training, splitting the data into non-overlapping chunks (with the same length as training), running each chunk through a model and then using the predictions made on each step of that chunk to calculate the average NLL for the chunk.

Evaluation of generated samples. Following Kong et al. (2021), we use 4 standard evaluation metrics for generative models evaluated using an auxiliary ResNeXT classifier (Xie et al., 2017) which achieved 98.3% accuracy on the test set. Note that Kong et al. (2021) reported an additional metric NDB (number of statistically-distinct bins), which we found to be slow to compute and generally uninformative, despite SaShiMi performing best.

- **Fréchet Inception Distance (FID)** (Heusel et al., 2017) uses the classifier to compare moments of generated and real samples in feature space.
- **Inception Score (IS)** (Salimans et al., 2016) measures both quality and diversity of generated samples, and favoring samples that the classifier is confident on.
- **Modified Inception Score (mIS)** (Gurumurthy et al., 2017) provides a measure of both intra-class in addition to inter-class diversity.
- **AM Score** (Zhou et al., 2018) uses the marginal label distribution of training data compared to IS.

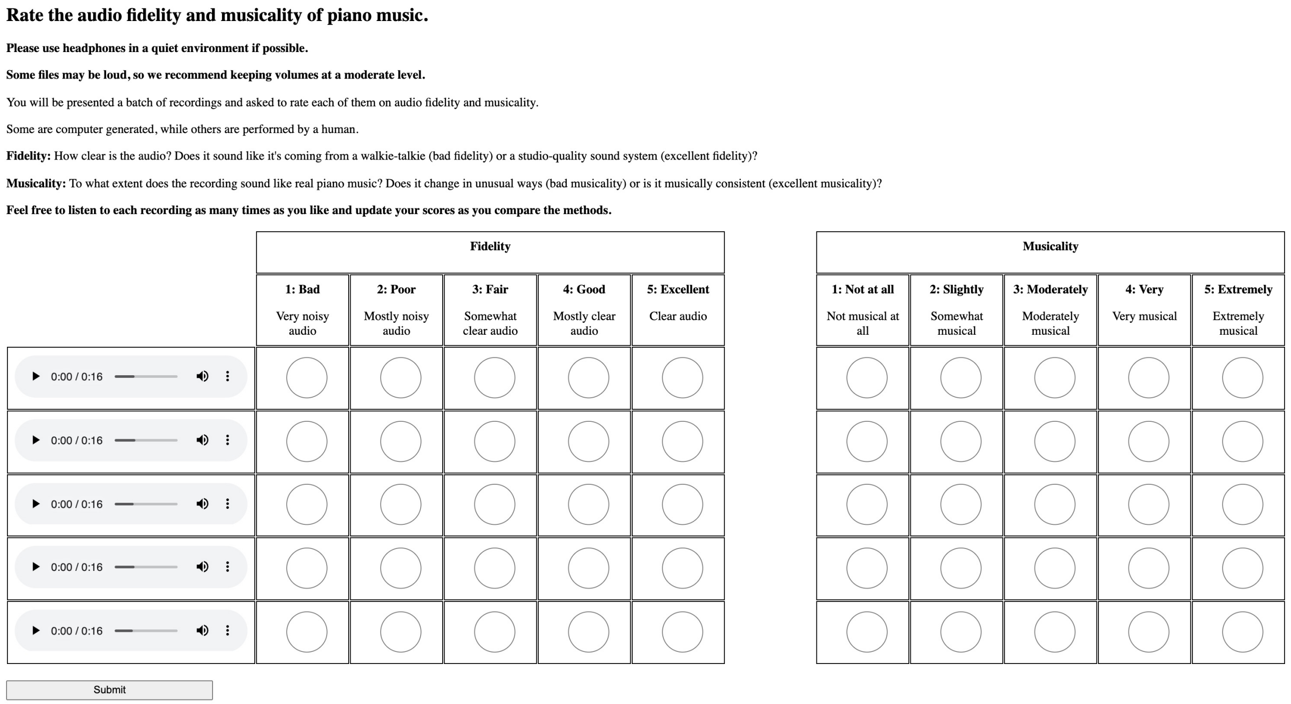
We also report the Cohen's inter-annotator agreement κ score, which is computed with the classifier as one rater and a crowdworker's digit prediction as the other rater (treating the set of crowdworkers as a single rater).

C.3.1. Evaluation Procedure for Autoregressive Models

Because autoregressive models have tractable likelihood scores that are easily evaluated, we use them to perform a form of rejection sampling when evaluating their automated metrics. Each model generated 5120 samples and ranked them by likelihood scores. The lowest-scoring 0.40 and highest-scoring 0.05 fraction of samples were thrown out. The remaining samples were used to calculate the automated metrics.

The two thresholds for the low- and high- cutoffs were found by validation on a separate set of 5120 generated samples.

Figure 6. **(YouTubeMix MOS Interface)** Crowdsourcing interface for collecting mean opinion scores (MOS) on YouTubeMix. Crowdworkers are given a collection of audio files, one from each method and the dataset. They are asked to rate each file on audio fidelity and musicality.



Rate the audio fidelity and musicality of piano music.

Please use headphones in a quiet environment if possible.

Some files may be loud, so we recommend keeping volumes at a moderate level.

You will be presented a batch of recordings and asked to rate each of them on audio fidelity and musicality.

Some are computer generated, while others are performed by a human.

Fidelity: How clear is the audio? Does it sound like it's coming from a walkie-talkie (bad fidelity) or a studio-quality sound system (excellent fidelity)?

Musicality: To what extent does the recording sound like real piano music? Does it change in unusual ways (bad musicality) or is it musically consistent (excellent musicality)?

Feel free to listen to each recording as many times as you like and update your scores as you compare the methods.

	Fidelity					Musicality				
	1: Bad Very noisy audio	**2: Poor** Mostly noisy audio	**3: Fair** Somewhat clear audio	**4: Good** Mostly clear audio	**5: Excellent** Clear audio	**1: Not at all** Not musical at all	**2: Slightly** Somewhat musical	**3: Moderately** Moderately musical	**4: Very** Very musical	**5: Extremely** Extremely musical
▶ 0:00 / 0:16	○	○	○	○	○	○	○	○	○	○
▶ 0:00 / 0:16	○	○	○	○	○	○	○	○	○	○
▶ 0:00 / 0:16	○	○	○	○	○	○	○	○	○	○
▶ 0:00 / 0:16	○	○	○	○	○	○	○	○	○	○
▶ 0:00 / 0:16	○	○	○	○	○	○	○	○	○	○

C.3.2. Evaluation Procedure for Non-autoregressive Models

Automated metrics were calculated on 2048 random samples generated from each model.

C.4. Evaluation of Mean Opinion Scores

For evaluating mean opinion scores (MOS), we repurpose scripts for creating jobs for Amazon Mechanical Turk from Neekhara et al. (2019).

C.4.1. Mean Opinion Scores for YouTubeMix

We collect MOS scores on audio fidelity and musicality, following Dieleman et al. (2018a). The instructions and interface used are shown in Figure 6.

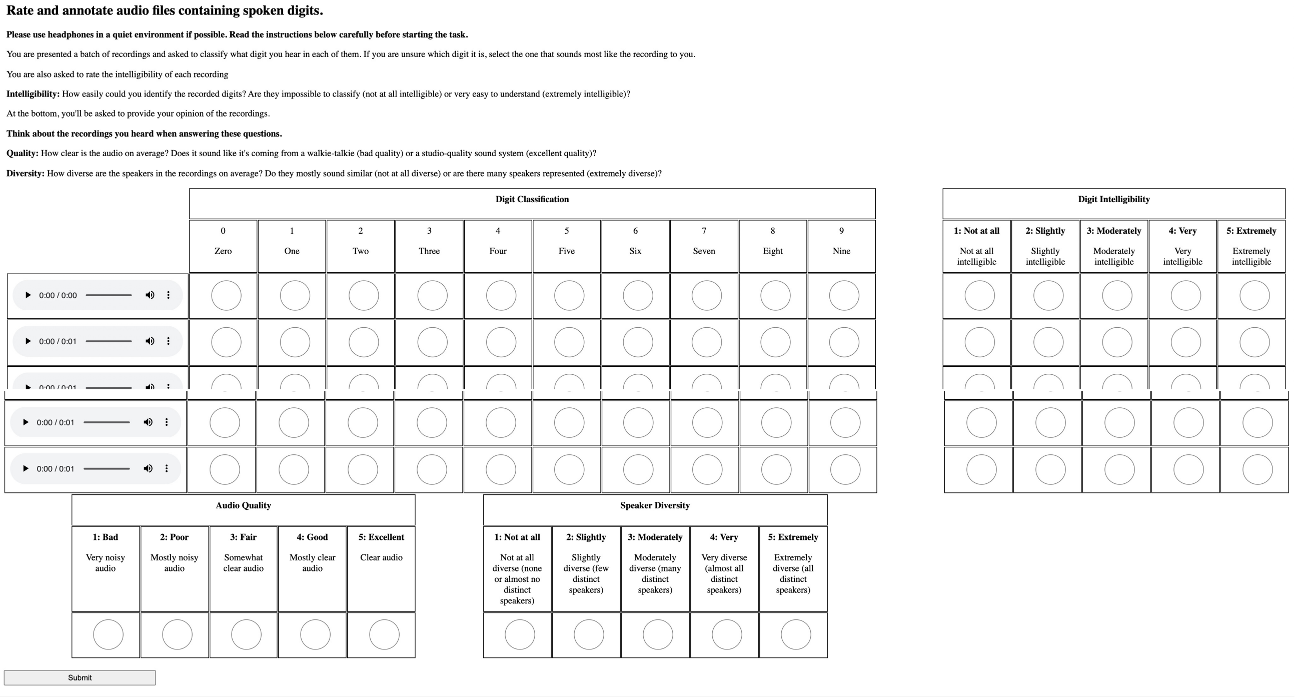
The protocol we follow to collect MOS scores for YouTubeMix is outlined below. For this study, we compare unconditional AR models, SaShiMi to SampleRNN and WaveNet.

- For each method, we generated unconditional 1024 samples, where each sample had length 16s (1.024M steps). For sampling, we directly sample from the distribution output by the model at each time step, without using any other modifications.

- As noted by Mehri et al. (2017), autoregressive models can sometimes generate samples that are "noise-like". To fairly compare all methods, we sequentially inspect the samples and reject any that are noise-like. We also remove samples that mostly consist of silences (defined as more than half the clip being silence). We carry out this process until we have 30 samples per method.

- Next, we randomly sample 25 clips from the dataset. Since this evaluation is quite subjective, we include some gold standard samples. We add 4 clips that consist mostly of noise (and should have musicality and quality MOS $<= 2$).

Figure 7. **(SC09 MOS Interface)** Crowdsourcing interface for collecting mean opinion scores (MOS) on SC09. Crowdworkers are given a collection of 10 audio files from the same method, and are asked to classify the spoken digits and rate them on intelligibility. At the bottom, they provide a single score on the audio quality and speaker diversity they perceive for the batch.



Rate and annotate audio files containing spoken digits.

Please use headphones in a quiet environment if possible. Read the instructions below carefully before starting the task.

You are presented a batch of recordings and asked to classify what digit you hear in each of them. If you are unsure which digit it is, select the one that sounds most like the recording to you.

You are also asked to rate the intelligibility of each recording

Intelligibility: How easily could you identify the recorded digits? Are they impossible to classify (not at all intelligible) or very easy to understand (extremely intelligible)?

At the bottom, you'll be asked to provide your opinion of the recordings.

Think about the recordings you heard when answering these questions.

Quality: How clear is the audio on average? Does it sound like it's coming from a walkie-talkie (bad quality) or a studio-quality sound system (excellent quality)?

Diversity: How diverse are the speakers in the recordings on average? Do they mostly sound similar (not at all diverse) or are there many speakers represented (extremely diverse)?

	Digit Classification										**Digit Intelligibility**				
	0 Zero	1 One	2 Two	3 Three	4 Four	5 Five	6 Six	7 Seven	8 Eight	9 Nine	**1: Not at all** Not at all intelligible	**2: Slightly** Slightly intelligible	**3: Moderately** Moderately intelligible	**4: Very** Very intelligible	**5: Extremely** Extremely intelligible
▶ 0:00 / 0:00	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
▶ 0:00 / 0:01	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
▶ 0:00 / 0:01	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
▶ 0:00 / 0:01	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
▶ 0:00 / 0:01	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○

Audio Quality				
1: Bad Very noisy audio	**2: Poor** Mostly noisy audio	**3: Fair** Somewhat clear audio	**4: Good** Mostly clear audio	**5: Excellent** Clear audio
○	○	○	○	○

Speaker Diversity				
1: Not at all Not at all diverse (none or almost no distinct speakers)	**2: Slightly** Slightly diverse (few distinct speakers)	**3: Moderately** Moderately diverse (many distinct speakers)	**4: Very** Very diverse (almost all distinct speakers)	**5: Extremely** Extremely diverse (all distinct speakers)
○	○	○	○	○

We include 1 clip that has variable quality but musicality MOS <= 2. Any workers who disagree with this assessment have their responses omitted from the final evaluation.

- We construct 30 batches, where each batch consists of 1 sample per method (plus a single sample for the dataset), presented in random order to a crowdworker. We use Amazon Mechanical Turk for collecting responses, paying \$0.50 per batch and collecting 20 responses per batch. We use Master qualifications for workers, and restrict to workers with a HIT approval rating above 98%. We note that it is likely enough to collect 10 responses per batch.

C.4.2. MEAN OPINION SCORES FOR SC09

Next, we outline the protocol used for collecting MOS scores on SC09. We collect MOS scores on digit intelligibility, audio quality and speaker diversity, as well as asking crowdworkers to classify digits following Donahue et al. (2019). The instructions and interface used are shown in Figure 7.

- For each method, we generate 2048 samples of 1s each. For autoregressive models (SASHIMI, SampleRNN, WaveNet), we directly sample from the distribution output by the model at each time step, without any modification. For WaveGAN, we obtained 50000 randomly generated samples from the authors, and subsampled 2048 samples randomly from this set. For the diffusion models, we run 200 steps of denoising following Kong et al. (2021).
- We use the ResNeXT model (Appendix C.3) to classify the generated samples into digit categories. Within each digit category, we choose the top-50 samples, as ranked by classifier confidence. We note that this mimics the protocol followed by Donahue et al. (2019), which we established through correspondence with the authors.
- Next, we construct batches consisting of 10 random samples (randomized over all digits) drawn from a single method (or the dataset). Each method (and the dataset) thus has 50 total batches. We use Amazon Mechanical Turk for collecting responses, paying \$0.20 per batch and collecting 10 responses per batch. We use Master qualification for workers, and restrict to workers with a HIT approval rating above 98%.

Note that we elicit digit classes and digit intelligibility scores for each audio file, while audio quality and speaker diversity are elicited once per batch.