MelNet: A Generative Model for Audio in the Frequency Domain

Sean Vasquez [1] **Mike Lewis** [1]

Abstract

Capturing high-level structure in audio waveforms is challenging because a single second of audio spans tens of thousands of timesteps. While long-range dependencies are difficult to model directly in the time domain, we show that they can be more tractably modelled in two-dimensional time-frequency representations such as spectrograms. By leveraging this representational advantage, in conjunction with a highly expressive probabilistic model and a multiscale generation procedure, we design a model capable of generating high-fidelity audio samples which capture structure at timescales that time-domain models have yet to achieve. We apply our model to a variety of audio generation tasks, including unconditional speech generation, music generation, and text-to-speech synthesis—showing improvements over previous approaches in both density estimates and human judgments.

1. Introduction

Audio waveforms have complex structure at drastically varying timescales, which presents a challenge for generative models. Local structure must be captured to produce high-fidelity audio, while long-range dependencies spanning tens of thousands of timesteps must be captured to generate audio which is globally consistent. Existing generative models of waveforms such as WaveNet [47] and SampleRNN [34] are well-adapted to model local dependencies, but as these models typically only backpropagate through a fraction of a second, they are unable to capture high-level structure that emerges on the scale of several seconds.

We introduce a generative model for audio which captures longer-range dependencies than existing end-to-end models. We primarily achieve this by modelling 2D time-frequency representations such as spectrograms rather than 1D time-domain waveforms (Figure 1). The temporal axis of a spectrogram is orders of magnitude more compact than that of a waveform, meaning dependencies that span tens of thousands of timesteps in waveforms only span hundreds of timesteps in spectrograms. In practice, this enables our spectrogram models to generate unconditional speech and music samples with consistency over multiple seconds whereas time-domain models must be conditioned on intermediate features to capture structure at similar timescales. Additionally, it enables fully end-to-end text-to-speech—a task which has yet to be proven feasible with time-domain models.

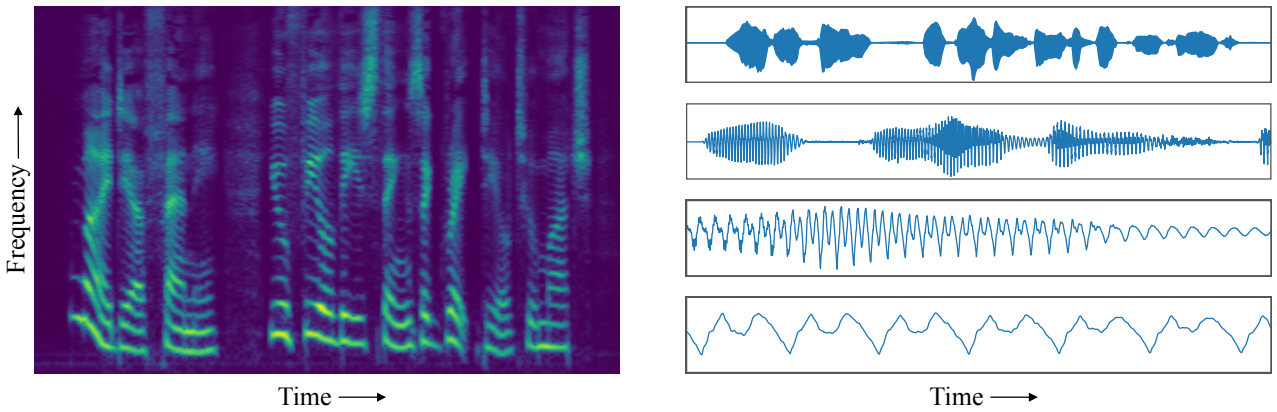
Modelling spectrograms can simplify the task of capturing global structure, but can weaken a model's ability to capture local characteristics that correlate with audio fidelity. Producing high-fidelity audio has been challenging for existing spectrogram models, which we attribute to the lossy nature of spectrograms and oversmoothing artifacts which result from insufficiently expressive models. To reduce information loss, we model high-resolution spectrograms which have the same dimensionality as their corresponding time-domain signals. To limit oversmoothing, we use a highly expressive autoregressive model which factorizes the distribution over both the time and frequency dimensions.

Modelling both fine-grained details and high-level structure in high-dimensional distributions is known to be challenging for autoregressive models. To capture both local and global structure in spectrograms with hundreds of thousands of dimensions, we employ a multiscale approach which generates spectrograms in a coarse-to-fine manner. A low-resolution, subsampled spectrogram that captures high-level structure is generated initially, followed by an iterative upsampling procedure that adds high-resolution details.

Combining these representational and modelling techniques yields a highly expressive, broadly applicable, and fully end-to-end generative model of audio. Our contributions are:

- We introduce MelNet, a generative model for spectrograms which couples a fine-grained autoregressive model and a multiscale generation procedure to jointly capture local and global structure.
- We show that MelNet is able to model longer-range dependencies than existing time-domain models.
- We demonstrate that MelNet is broadly applicable to a variety of audio generation tasks—capable of unconditional speech generation, music generation, and text-to-speech synthesis, entirely end-to-end.

[1]Facebook AI Research.



(a) Spectrogram representation

(b) Waveform representation (1x, 5x, 25x, 125x magnifications)

Figure 1. Spectrogram and waveform representations of the same four-second audio signal. The waveform spans nearly 100,000 timesteps whereas the temporal axis of the spectrogram spans roughly 400. Complex structure is nested within the temporal axis of the waveform at various timescales, whereas the spectrogram has structure which is smoothly spread across the time-frequency plane.

2. Preliminaries

We briefly present background regarding spectral representations of audio. Audio is represented digitally as a one-dimensional, discrete-time signal $y = (y_1, \ldots, y_n)$. Existing generative models for audio have predominantly focused on modelling these time-domain signals directly. We instead model spectrograms, which are two-dimensional time-frequency representations which contain information about how the frequency content of an audio signal varies through time. Spectrograms are computed by taking the squared magnitude of the short-time Fourier transform (STFT) of a time-domain signal, i.e. $x = \|\text{STFT}(y)\|^2$. The value of x_{ij} (referred to as amplitude or energy) corresponds to the squared magnitude of the jth element of the frequency response at timestep i. Each slice $x_{i,*}$ is referred to as a *frame*. We assume a time-major ordering, but following convention, all figures are displayed transposed and with the frequency axis inverted.

Time-frequency representations such as spectrograms highlight how the tones and pitches within an audio signal vary through time. Such representations are closely aligned with how humans perceive audio. To further align these representations with human perception, we convert the frequency axis to the Mel scale and apply an elementwise logarithmic rescaling of the amplitudes. Roughly speaking, the Mel transformation aligns the frequency axis with human perception of pitch and the logarithmic rescaling aligns the amplitude axis with human perception of loudness.

Spectrograms are lossy representations of their corresponding time-domain signals. The Mel transformation discards frequency information and the removal of the STFT phase discards temporal information. When recovering a time-domain signal from a spectrogram, this information loss manifests as distortion in the recovered signal. To minimize these artifacts and improve the fidelity of generated audio, we model high-resolution spectrograms. The temporal resolution of a spectrogram can be increased by decreasing the STFT hop size, and the frequency resolution can be increased by increasing the number of mel channels. Generated spectrograms are converted back to time-domain signals using classical spectrogram inversion algorithms. We experiment with both Griffin-Lim [18] and a gradient-based inversion algorithm [10], and ultimately use the latter as it generally produced audio with fewer artifacts.

3. Probabilistic Model

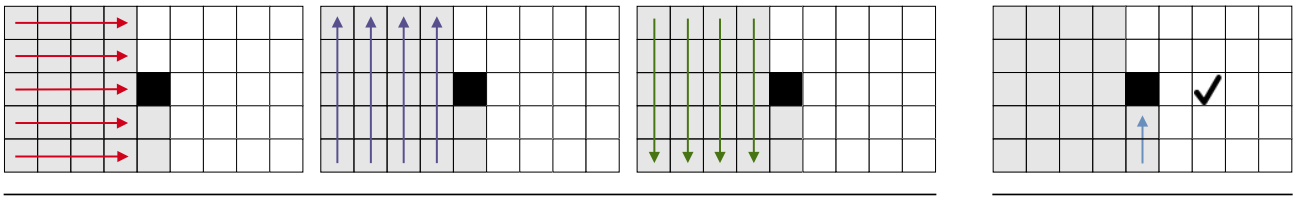
We use an autoregressive model which factorizes the joint distribution over a spectrogram x as a product of conditional distributions. Given an ordering of the dimensions of x, we define the context $x_{<ij}$ as the elements of x that precede x_{ij}. We default to a row-major ordering which proceeds through each frame $x_{i,*}$ from low to high frequency, before progressing to the next frame. The joint density is factorized as

$$p(x) = \prod_i \prod_j p(x_{ij} \mid x_{<ij};\ \theta_{ij}), \tag{1}$$

where θ_{ij} parameterizes a univariate density over x_{ij}. We model each factor distribution as a Gaussian mixture model with K components. Thus, θ_{ij} consists of $3K$ parameters corresponding to means $\{\mu_{ijk}\}_{k=1}^K$, standard deviations $\{\sigma_{ijk}\}_{k=1}^K$, and mixture coefficients $\{\pi_{ijk}\}_{k=1}^K$. The resulting factor distribution can then be expressed as

$$p(x_{ij} \mid x_{<ij};\ \theta_{ij}) = \sum_{k=1}^{K} \pi_{ijk}\, \mathcal{N}(x_{ij};\ \mu_{ijk}, \sigma_{ijk}). \tag{2}$$

Following the work on Mixture Density Networks [4] and their application to autoregressive models [15], θ_{ij} is



(a) Time-delayed stack (b) Frequency-delayed stack

Figure 2. The context $x_{<ij}$ (grey) for the element x_{ij} (black) is encoded using 4 RNNs. Three of these are used in the time-delayed stack to extract features from all preceding frames. The fourth is used in the frequency-delayed stack to extract features from all preceding elements within the current frame. Each arrow denotes an individual RNN cell and arrows of the same color use shared parameters.

modelled as the output of a neural network and computed as a function of the context $x_{<ij}$. Precisely, for some network f with parameters ψ, we have $\theta_{ij} = f(x_{<ij};\ \psi)$. A maximum-likelihood estimate for the network parameters is computed by minimizing the negative log-likelihood via gradient descent.

To ensure that the network output parameterizes a valid Gaussian mixture model, the network first computes unconstrained parameters $\{\hat{\mu}_{ijk}, \hat{\sigma}_{ijk}, \hat{\pi}_{ijk}\}_{k=1}^{K}$ as a vector $\hat{\theta}_{ij} \in \mathbb{R}^{3K}$, and enforces constraints on θ_{ij} by applying the following transformations:

$$\mu_{ijk} = \hat{\mu}_{ijk} \tag{3}$$

$$\sigma_{ijk} = \exp(\hat{\sigma}_{ijk}) \tag{4}$$

$$\pi_{ijk} = \frac{\exp(\hat{\pi}_{ijk})}{\sum_{k=1}^{K} \exp(\hat{\pi}_{ijk})}\ . \tag{5}$$

These transformations ensure the standard deviations σ_{ijk} are positive and the mixture coefficients π_{ijk} sum to one.

4. Network Architecture

To model the distribution in an autoregressive manner, we design a network which computes the distribution over x_{ij} as a function of the context $x_{<ij}$. The network architecture draws inspiration from existing autoregressive models for images [45, 49, 48, 5, 41, 36, 7]. In the same way that these models estimate a distribution pixel-by-pixel over the spatial dimensions of an image, our model estimates a distribution element-by-element over the time and frequency dimensions of a spectrogram. A noteworthy distinction is that spectrograms are not invariant to translation along the frequency axis, making the use of 2D convolution undesirable. Utilizing multidimensional recurrence instead of 2D convolution has been shown to be beneficial when modelling spectrograms in discriminative settings [32, 40], which motivates our use of an entirely recurrent architecture.

Similar to Gated PixelCNN [48], the network has multiple *stacks* of computation. These stacks extract features from different segments of the input to collectively summarize the full context $x_{<ij}$:

- The *time-delayed* stack computes features which aggregate information from all previous frames $x_{<i,*}$.
- The *frequency-delayed* stack utilizes all preceding elements within a frame, $x_{i,<j}$, as well as the outputs of the time-delayed stack, to compute features which summarize the full context $x_{<ij}$.

The stacks are connected at each layer of the network, meaning that the features generated by layer l of the time-delayed stack are used as input to layer l of the frequency-delayed stack. To facilitate the training of deeper networks, both stacks use residual connections [20]. The outputs of the final layer of the frequency-delayed stack are used to compute the unconstrained parameters $\hat{\theta}$.

4.1. Time-Delayed Stack

The time-delayed stack utilizes multiple layers of multidimensional RNNs to extract features from $x_{<i,*}$, the two-dimensional region consisting of all frames preceding x_{ij}. Each multidimensional RNN is composed of three one-dimensional RNNs: one which runs forwards along the frequency axis, one which runs backwards along the frequency axis, and one which runs forwards along the time axis. Each RNN runs along each slice of a given axis, as shown in Figure 2. The output of each layer of the time-delayed stack is the concatenation of the three RNN hidden states.

We denote the function computed at layer l of the time-delayed stack (three RNNs followed by concatenation) as $\mathcal{F}_l^t$. At each layer, the time-delayed stack uses the feature map from the previous layer, $h^t[l-1]$, to compute the subsequent feature map $\mathcal{F}_l^t\big(h^t[l-1]\big)$ which consists of the three concatenated RNN hidden states. When using residual connections, the computation of $h^t[l]$ from $h^t[l-1]$ becomes

$$h_{ij}^t[l] = W_l^t \mathcal{F}_l^t\big(h^t[l-1]\big)_{ij} + h_{ij}^t[l-1]. \tag{6}$$

To ensure the output $h_{ij}^t[l]$ is only a function of frames which lie in the context $x_{<ij}$, the inputs to the time-delayed stack are shifted backwards one step in time:

$$h_{ij}^t[0] = W_0^t x_{i-1,j}. \tag{7}$$

4.2. Frequency-Delayed Stack

The frequency-delayed stack is a one-dimensional RNN which runs forward along the frequency axis. Much like existing one-dimensional autoregressive models (language models, waveform models, etc.), the frequency-delayed stack operates on a one-dimensional sequence (a single frame) and estimates the distribution for each element conditioned on all preceding elements. The primary difference is that it is also conditioned on the outputs of the time-delayed stack, allowing it to use the full two-dimensional context $x_{<ij}$.

We denote the function computed by the frequency-delayed stack as $\mathcal{F}_l^f$. At each layer, the frequency-delayed stack takes two inputs: the the previous-layer outputs of the frequency-delayed stack, $h^f[l-1]$, and the current-layer outputs of the time-delayed stack $h^t[l]$. These inputs are summed and used as input to a one-dimensional RNN to produce the output feature map $\mathcal{F}_l^f\left(h^f[l-1],\ h^t[l]\right)$ which consists of the RNN hidden state:

$$h_{ij}^f[l] = W_l^f \mathcal{F}_l^f\left(h^f[l-1],\ h^t[l]\right)_{ij} + h_{ij}^f[l-1]. \quad (8)$$

To ensure that $h_{ij}^f[l]$ is computed using only elements in the context $x_{<ij}$, the inputs to the frequency-delayed stack are shifted backwards one step along the frequency axis:

$$h_{ij}^f[0] = W_0^f x_{i,j-1}. \quad (9)$$

At the final layer, layer L, a linear map is applied to the output of the frequency-delayed stack to produce the unconstrained parameters:

$$\hat{\theta}_{ij} = W_\theta h_{ij}^f[L]. \quad (10)$$

4.3. Centralized Stack

The recurrent state of the time-delayed stack is distributed across an array of RNN cells which tile the frequency axis. To allow for a more centralized representation, we optionally include an additional stack consisting of an RNN which, at each timestep, takes an entire frame as input and outputs a single vector consisting of the RNN hidden state. Denoting this function as $\mathcal{F}_l^c$ gives the layer update

$$h_i^c[l] = W_l^c \mathcal{F}_l^c\left(h^c[l-1]\right)_i + h_i^c[l-1]. \quad (11)$$

Similar to the time-delayed stack, the centralized stack operates on frames which are shifted backwards one step along the time axis:

$$h_i^c[0] = W_0^c x_{i-1,*}. \quad (12)$$

The output of the centralized stack is input to the frequency-delayed stack at each layer, meaning that the frequency-delayed stack is a function of three inputs: $h^f[l-1]$, $h^t[l]$, and $h^c[l]$. These three inputs are simply summed and used as input to the RNN in the frequency-delayed stack.

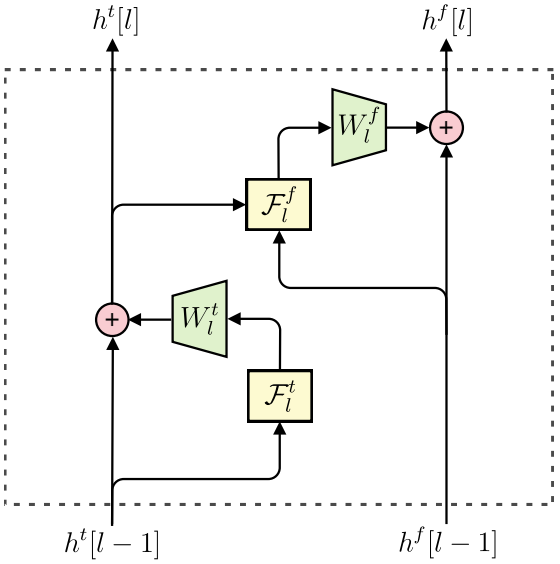


Figure 3. Computation graph for a single layer of the network. $\mathcal{F}_l^t$ and $\mathcal{F}_l^f$ are the functions computed by the time-delayed stack and frequency-delayed stack, respectively, at layer l. The outputs of these functions are projected (by the matrices W_l^t and W_l^f) and summed with the layer inputs to form residual blocks.

4.4. Conditioning

To incorporate conditioning information into the model, conditioning features z are simply projected onto the input layer along with the inputs x, altering Equations 7 and 9:

$$h_{ij}^t[0] = W_0^t x_{i-1,j} + W_z^t z_{ij} \quad (13)$$

$$h_{ij}^f[0] = W_0^f x_{i,j-1} + W_z^f z_{ij}. \quad (14)$$

Reshaping, upsampling, and broadcasting can be used as necessary to ensure the conditioning features have the same time and frequency shape as the input spectrogram, e.g. a one-hot vector representation for speaker ID would first be broadcast along both the time and frequency axes.

5. Learned Alignment

For the task of end-to-end text-to-speech, the network must learn a latent alignment between spectrogram frames $(x_{1,*}, \ldots, x_{T,*})$ and discrete character tokens $(c_1, \ldots, c_U)$. To facilitate this, we first extract character features $(\tilde{c}_1, \ldots, \tilde{c}_U)$ by embedding each character c_u and running a bidirectional RNN over the embeddings. Extracting character features eases the alignment process by allowing the network to learn both phonetic features which are important for pronunciation and higher-level semantics which must be understood to infer proper intonation and prosody.

We use an attention mechanism which is a straightforward variant of the location-based Gaussian mixture attention

introduced by Graves [15]. The attention mechanism consists of an RNN in the centralized stack which, at timestep i, computes an attention vector w_i as a weighted sum of character features $(\tilde{c}_1, \ldots, \tilde{c}_U)$. The weights correspond to a learned attention distribution $\phi_i(\,\cdot\,;\,\gamma_i)$ whose parameters γ_i are computed as a simple function g of the RNN hidden state. This is expressed by the following recurrence, where y_i represents an arbitrary input at timestep i:

$$h_i = \text{RNN}\Big([y_i,\ w_{i-1}],\ h_{i-1}\Big) \tag{15}$$

$$w_i = \sum_{u=1}^{U} \phi_i\Big(u;\ \gamma_i = g\big(h_i\big)\Big)\tilde{c}_u. \tag{16}$$

The original formulation parameterizes $\phi_i(\,\cdot\,;\,\gamma_i)$ as an unnormalized Gaussian mixture model, whereas we use a discretized mixture of logistics [41]. In either case, the distribution is parameterized by $\gamma_i = \{\kappa_i^m, \beta_i^m, \alpha_i^m\}_{m=1}^M$, corresponding to M means, scales, and mixture coefficients. We define the function g as a trainable linear mapping of the RNN hidden state h_i followed by transformations which constrain the mixture coefficients α_i^m to sum to one, the scales β_i^m to be positive, and the means κ_i^m to be monotonically increasing with i:

$$\{\hat{\kappa}_i^m, \hat{\beta}_i^m, \hat{\alpha}_i^m\}_{m=1}^M = W_g h_i \tag{17}$$

$$\kappa_i^m = \kappa_{i-1}^m + \exp(\hat{\kappa}_i^m) \tag{18}$$

$$\beta_i^m = \exp(\hat{\beta}_i^m) \tag{19}$$

$$\alpha_i^m = \frac{\exp(\hat{\alpha}_i^m)}{\sum_{m=1}^M \exp(\hat{\alpha}_i^m)}. \tag{20}$$

The resulting mixture of logistics distribution parameterized by γ_i has the distribution function

$$F_i(u;\ \gamma_i) = \sum_{m=1}^{M} \alpha_i^m \left(1 + \exp\left(\frac{\kappa_i^m - u}{\beta_i^m}\right)\right)^{-1} \tag{21}$$

which is then used to compute the discretized attention distribution

$$\phi_i(u;\ \gamma_i) = F_i(u + 0.5;\ \gamma_i) - F_i(u - 0.5;\ \gamma_i). \tag{22}$$

The network needs a criterion by which it can determine whether it has finished 'reading' the text and can terminate sampling. If we interpret $\phi_i(u;\ \gamma_i)$ as the network's belief that it is reading character c_u at timestep i, then the network's belief that it has passed the final character c_U is $\sum_{U+1}^{\infty} \phi_i(u;\ \gamma_i)$, which can be expressed in closed form as $\overline{F}_i(U + 0.5;\ \gamma_i)$ where $\overline{F}_i$ is the survival function $1 - F_i$. We stop sampling based on a simple threshold of this value, terminating at the first timestep i such that $\overline{F}_i(U + 0.5;\ \gamma_i) > \tau$. We compute an estimate for τ after the network is trained, using the empirical mean $\hat{\tau} = \frac{1}{N}\sum_{n=1}^{N} \overline{F}_{T_n}(U_n + 0.5;\ \gamma_{T_n})$.

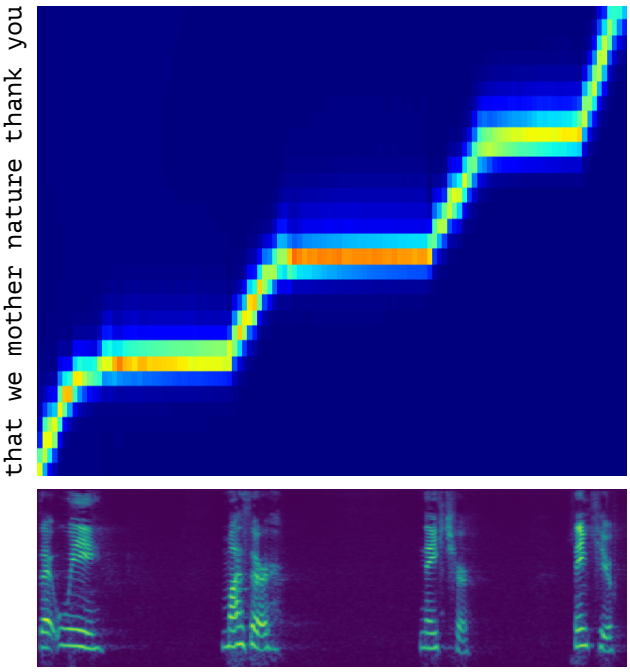


Figure 4. Learned alignment between a spectrogram and the character sequence *that we mother nature thank you*. Column i corresponds to the learned attention distribution $\phi_i(\,\cdot\,;\,\gamma_i)$. The text is read with long, deliberate pauses (*that we / mother / nature / thank you*) which appear as flat regions in the alignment.

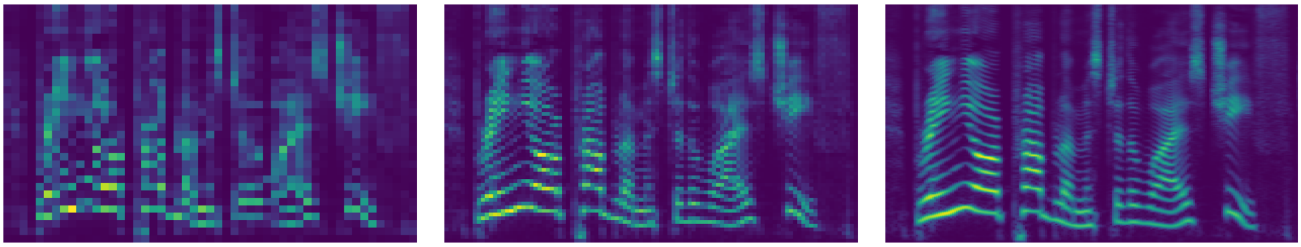
6. Multiscale Modelling

To improve audio fidelity, we generate high-resolution spectrograms which have the same dimensionality as their corresponding time-domain representations. Under this regime, a single training example has several hundreds of thousands of dimensions. Capturing global structure in such high-dimensional distributions is challenging for autoregressive models, which are biased towards capturing local dependencies. To counteract this, we utilize a multiscale approach which effectively permutes the autoregressive ordering so that a spectrogram is generated in a coarse-to-fine order.

The elements of a spectrogram x are partitioned into G tiers $x^1, \ldots, x^G$, such that each successive tier contains higher-resolution information. We define $x^{<g}$ as the union of all tiers which precede x^g, i.e. $x^{<g} = (x^1, \ldots, x^{g-1})$. The distribution is factorized over tiers:

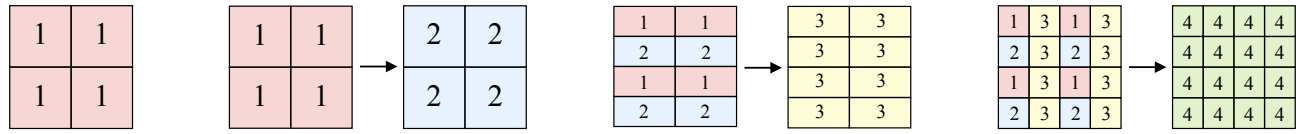
$$p(x;\ \psi) = \prod_g p(x^g \mid x^{<g};\ \psi^g), \tag{23}$$

and the distribution of each tier is further factorized element-by-element as described in Section 3. We explicitly include the parameterization by $\psi = (\psi^1, \ldots, \psi^G)$ to indicate that each tier is modelled by a separate network.



(a) Tier 1 (32×50) (b) Tiers 1–3 (64×100) (c) Tiers 1–6 (256×200)

Figure 5. A sampled spectrogram viewed at different stages of the multiscale generation procedure. The initial tier dictates high-level structure and subsequent tiers add fine-grained details. Each upsampling tier doubles the resolution of the spectrogram, resulting in the initial tier being upsampled by a factor of 32.



(a) $p(x^1;\ \psi^1)$ (b) $p(x^2 \mid x^1;\ \psi^2)$ (c) $p(x^3 \mid x^1, x^2;\ \psi^3)$ (d) $p(x^4 \mid x^1, x^2, x^3;\ \psi^4)$

Figure 6. Schematic showing how tiers of the multiscale model are interleaved and used to condition the distribution for the subsequent tier. a) The initial tier is generated unconditionally. b) The second tier is generated conditionally given the the initial tier. c) The outputs of tiers 1 and 2 are interleaved along the frequency axis and used to condition the generation of tier 3. d) Tier 3 is interleaved along the time axis with all preceding tiers and used to condition the generation of tier 4.

6.1. Training

During training, the tiers are generated by recursively partitioning a spectrogram into alternating rows along either the time or frequency axis. We define a function `split` which partitions an input into even and odd rows along a given axis. The initial step of the recursion applies the `split` function to a spectrogram x, or equivalently $x^{<G+1}$, so that the even-numbered rows are assigned to x^G and the odd-numbered rows are assigned to $x^{<G}$. Subsequent tiers are defined similarly in a recursive manner:

$$x^g,\ x^{<g} = \texttt{split}(x^{<g+1}). \tag{24}$$

At each step of the recursion, we model the distribution $p(x^g \mid x^{<g};\ \psi^g)$. The final step of the recursion models the unconditional distribution over the initial tier $p(x^1;\ \psi^1)$.

To model the conditional distribution $p(x^g \mid x^{<g};\ \psi^g)$, the network at each tier needs a mechanism to incorporate information from the preceding tiers $x^{<g}$. To this end, we add a feature extraction network which computes features from $x^{<g}$ which are used to condition the generation of x^g. We use a multidimensional RNN consisting of four one-dimensional RNNs which run bidirectionally along slices of both axes of the context $x^{<g}$. A layer of the feature extraction network is similar to a layer of the time-delayed stack, but since the feature extraction network is not causal, we include an RNN which runs backwards along the time axis and do not shift the inputs. The hidden states of the RNNs in the feature extraction network are used to condition the generation of x^g. Since each tier doubles the resolution, the features extracted from $x^{<g}$ have the same time and frequency shape as x^g, allowing the conditioning mechanism described in section 4.4 to be used straightforwardly.

6.2. Sampling

To sample from the multiscale model we iteratively sample a value for x^g conditioned on $x^{<g}$ using the learned distributions defined by the estimated network parameters $\hat{\psi} = (\hat{\psi}^1, \ldots, \hat{\psi}^G)$. The initial tier, x^1, is generated unconditionally by sampling from $p(x^1;\ \hat{\psi}^1)$ and subsequent tiers are sampled from $p(x^g \mid x^{<g};\ \hat{\psi}^g)$. At each tier, the sampled x^g is interleaved with the context $x^{<g}$:

$$x^{<g+1} = \texttt{interleave}(x^g,\ x^{<g}). \tag{25}$$

The `interleave` function is simply the inverse of the `split` function. Sampling terminates once a full spectrogram, $x^{<G+1}$, has been generated. A spectrogram generated by a multiscale model is shown in Figure 5 and the sampling procedure is visualized schematically in Figure 6.

7. Experiments

To demonstrate the MelNet is broadly applicable as a generative model for audio, we train the model on a diverse set of audio generation tasks using four publicly available datasets. We explore three unconditional audio generation tasks (single-speaker speech generation, multi-speaker

	Unconditional			Text-to-Speech	
	Blizzard	MAESTRO	VoxCeleb2	Blizzard	TED-LIUM 3
Tiers	6	4	5	6	5
Layers (Initial Tier)	12	16	16	8	12
Layers (Upsampling Tiers)	5-4-3-2-2	6-5-4	6-5-4-3	5-4-3-2-2	6-5-4-3
Hidden Size	512	512	512	512	512
GMM Mixture Components	10	10	10	10	10
Attention Mixture Components	-	-	-	10	10
Batch Size	32	16	128	32	64
Sample Rate (Hz)	22,050	22,050	16,000	22,050	16,000
Max Sample Duration (s)	10	6	6	10	10
Mel Channels	256	256	180	256	180
STFT Hop Size	256	256	180	256	180
STFT Window Size	$6 \cdot 256$	$6 \cdot 256$	$6 \cdot 180$	$6 \cdot 256$	$6 \cdot 180$

Table 1. MelNet hyperparameters. All RNNs use LSTM cells [22]. All models are trained with RMSProp [46] with a learning rate of 10^{-4} and momentum of 0.9. The initial values for all recurrent states are trainable parameters. A single hyperparameter controls the width of the network—all hidden sizes (RNN state size, residual connections, embeddings, etc.) are defined by a single value, denoted *hidden size* in the table. Only the initial tier is conditioned on text and speaker ID. Only the initial tier uses a centralized stack. When an attention cell is used in the centralized stack, it is inserted in the middle layer $(L/2)$.

speech generation, and music generation) as well as two text-to-speech tasks (single-speaker TTS and multi-speaker TTS). Generated audio samples for each task are available on the accompanying web page.[1] We include samples generated using the priming and biasing procedures described by Graves [15]. Biasing lowers the temperature of the distribution at each timestep and priming seeds the model state with a given sequence of audio prior to sampling.

7.1. Unconditional Audio Generation

Speech and music have rich hierarchies of latent structure. Speech has complex linguistic structure (phonemes, words, syntax, semantics, etc.) and music has highly compositional musical structure (notes, chords, melody and rhythm, etc.). The presence of these latent structures in generated samples can be used as a proxy for how well a generative model has learned dependencies at various timescales. As such, a qualitative analysis of unconditional samples is an insightful method of evaluating generative models of audio. We train MelNet on three unconditional audio generation tasks—single-speaker speech generation, multi-speaker speech generation, and music generation. For completeness, the sections below include brief discussions and qualitative observations regarding the generated samples. However, it is not possible to convey the many characteristics of the generated samples in text and we highly encourage the reader to listen to the audio samples and make their own judgments. In addition to qualitative analysis, we quantitatively compare MelNet to a WaveNet baseline across each of the three unconditional generation tasks.

[1] https://audio-samples.github.io

7.1.1. Single-Speaker Speech

To test the model's ability to model a single speaker in a controlled environment, we utilize the Blizzard 2013 dataset [28], which consists of audiobook narration performed in a highly animated manner by a professional speaker. We use a 140 hour subset of this dataset for which we were able to find transcriptions, making the dataset also suitable for future text-to-speech experiments. We find that MelNet frequently generates samples that contain coherent words and phrases. Even when the model generates incoherent speech, the intonation, prosody, and speaker characteristics remain consistent throughout the duration of the sample. Furthermore, the model learns to produce speech using a variety of character voices and learns to generate samples which contain elements of narration and dialogue. Biased samples tend to contain longer strings of comprehensible words but are read in a less expressive fashion. When primed with a real sequence of audio, MelNet is able to continue sampling speech which has consistent speaking style and intonation.

7.1.2. Multi-Speaker Speech

Audiobook data is recorded in a highly controlled environment. To demonstrate MelNet's capacity to model distributions with significantly more variation, we utilize the VoxCeleb2 dataset [8]. The VoxCeleb2 dataset consists of over 2,000 hours of speech data captured with real world noise including laughter, cross-talk, channel effects, music and other sounds. The dataset is also multilingual, with speech from speakers of 145 different nationalities, covering a wide range of accents, ages, ethnicities and languages.

When trained on the VoxCeleb2 dataset, we find that MelNet is able to generate unconditional samples with significant variation in both speaker characteristics (accent, language, prosody, speaking style) as well as acoustic conditions (background noise and recording quality). While the generated speech is generally not comprehensible, samples can often be identified as belonging to a specific language, indicating that the model has learned distinct modalities for different languages. Furthermore, it is difficult to distinguish real and fake samples which are spoken in foreign languages. For foreign languages, semantic structures are not understood by the listener and cannot be used to discriminate between real and fake. Consequently, the listener must rely largely on phonetic structure, which MelNet is able to realistically model.

7.1.3. MUSIC

To show that MelNet can model audio modalities other than speech, we apply the model to the task of unconditional music generation. We utilize the MAESTRO dataset [19], which consists of over 172 hours of solo piano performances. The samples demonstrate that MelNet learns musical structures such as melody and harmony. Furthermore, generated samples often maintain consistent tempo and contain interesting variation in volume, timbre, and rhythm.

7.1.4. HUMAN EVALUATION

Making quantitative comparisons with existing generative models such as WaveNet is difficult for various reasons. While WaveNet and MelNet both produce exact density estimates, these models cannot be directly compared using log-likelihood as they operate on different representations. We instead resort to comparing both models by evaluating their ability to model long-range dependencies. To make this comparison quantitatively, we conduct an experiment where we provide an anonymized ten second sample from both models to human evaluators and ask them to identify the sample which exhibits longer-term structure. Further details of the methodology for this experiment are provided in Appendix A.1. We conduct this experiment for each of the three unconditional audio generation tasks and report results in Table 2a. Evaluators overwhelmingly agreed that samples generated by MelNet had more coherent long-range structure than samples from WaveNet. Samples from both models are included on the accompanying web page.

In addition to comparing MelNet to an unconditional WaveNet model for music generation, we also compare to a two-stage Wave2Midi2Wave model [19] which conditions WaveNet on MIDI generated by a separately-trained Music Transformer [24]. Results, shown in Table 2b, show that despite having the advantage of directly modelling the musical notes, the two-stage model does not capture long-range structure as well as a MelNet model that is trained entirely end-to-end.

	WaveNet	MelNet
Blizzard	0.0 %	100.0 %
VoxCeleb2	0.0 %	100.0 %
MAESTRO	4.2 %	95.8 %

(a) Comparison between MelNet and WaveNet. Both models are trained in an entirely unsupervised manner.

	Wave2Midi2Wave	MelNet
MAESTRO	37.7 %	62.3 %

(b) Comparison between MelNet and Wave2Midi2Wave. Wave2Midi2Wave is a two-stage model consisting of a Music Transformer trained on labelled MIDI followed by a conditional WaveNet model. The MelNet model, on the other hand, is trained without any intermediate supervision.

Table 2. Selection rates of human evaluators when asked to identify which model generates samples with longer-term structure. Results show that MelNet captures long-range structure better than WaveNet. Furthermore, an end-to-end MelNet model outperforms a two-stage model which conditions WaveNet on generated MIDI.

7.2. Text-to-Speech Synthesis

We apply MelNet to the tasks of single-speaker and multi-speaker TTS. As was done for the unconditional tasks, we provide audio samples on the accompanying web page and provide a brief qualitative analysis of samples in the following sections. We then quantitatively evaluate our text-to-speech models on the task of density estimation.

7.2.1. SINGLE-SPEAKER TTS

To assess MelNet's ability to perform the task of single-speaker text-to-speech, we again use the audiobook data from the Blizzard 2013 dataset, including the corresponding transcriptions. As the dataset contains speech which is spoken in a highly expressive manner with significant variation, the distribution of audio given text is highly multimodal. To demonstrate that MelNet has learned to model these modalities, we include multiple speech samples for a given text. The samples demonstrate that MelNet learns to produce diverse vocalizations for the same text, many of which we are unable to easily distinguish from ground truth data. Furthermore, MelNet learns to infer speaking characteristics from text—samples which contain dialogue are read using various character voices, while narrative text is read in a relatively inexpressive manner. When primed with a sequence of audio, MelNet effectively infers speaker characteristics and can perform text-to-speech on unseen text while preserving the speaking style of the priming sequence.

7.2.2. MULTI-SPEAKER TTS

We also train MelNet on a significantly more challenging multi-speaker dataset. The TED-LIUM 3 dataset [21] consists of 452 hours of recorded TED talks. The dataset has various characteristics that make it particularly challenging. Firstly, the transcriptions are unpunctuated, unnormalized, and contain errors. Secondly, speaker IDs are noisy, as they do not discriminate between multiple speakers within a given talk, e.g. questions from interviewers and audience members. Lastly, the dataset includes significant variation in recording conditions, speaker characteristics (over 2,000 unique speakers with diverse accents), and background noise (applause and background music are common). Despite this, we find that MelNet is capable of producing realistic text-to-speech samples and can generate samples for different speakers by conditioning on different speaker IDs. Generated samples also contain speech disfluencies (e.g. umms and ahhs), repeated and skipped words, applause, laughter, and various other idiosyncrasies which result from the noisy nature of the data.

7.2.3. DENSITY ESTIMATION

Generative models trained with maximum-likelihood are most directly evaluated by the likelihood they assign to unseen data. Similar to MelNet, many existing works for end-to-end TTS are designed to model two-dimensional spectral features. However, density estimates cannot be used straightforwardly for comparison because existing TTS models generally do not use log-likelihood as a training objective. To make density estimation comparisons possible, we instead define a set of surrogate models which encode assumptions made by existing TTS models and compare the density estimates of these models to our own:

- **Diagonal Gaussian** The vast majority of end-to-end TTS systems such as Tacotron [53], DeepVoice [2], VoiceLoop [44], Char2Wav [43], and ClariNet [37] utilize a coarse autoregressive model, where spectral features are factorized as a product of per-frame factors. The elements within each frame are assumed to be conditionally independent and unimodal (given all preceding frames). To represent this class of models, we use a model which factorizes the distribution over frames

$$p(x) = \prod_i p(x_{i,*} \mid x_{<i,*}) \tag{26}$$

  and models each frame as a diagonal Gaussian with parameterized mean $\mu_i \in \mathbb{R}^d$ and standard deviation $\sigma_i \in \mathbb{R}^d_+$, where d is the dimension of each frame:

$$p(x_{i,*} \mid x_{<i,*}) = \mathcal{N}\big(x_{i,*};\ \mu_i, \text{diag}(\sigma_i^2)\big). \tag{27}$$

- **VAE: Global z** Subsequent works [1, 23] have used a similar frame-level factorization, but utilized a variational autoencoder (VAE) [30] to jointly model a latent variable z which conditions the generation of each frame. The joint distribution $p(x, z)$ is decomposed as $p(z)p(x \mid z)$, where

$$p(x \mid z) = \prod_i p(x_{i,*} \mid x_{<i,*}, z). \tag{28}$$

  As before, the conditional distribution over each frame, $p(x_{i,*} \mid x_{<i,*}, z)$, is modelled as a Gaussian with diagonal covariance.

- **VAE: Local z** We also introduce a more expressive VAE which differs only in that it utilizes a sequence of latent variables $z = (z_1, \ldots, z_T)$ instead of a single global latent variable:

$$p(x \mid z) = \prod_i p(x_{i,*} \mid x_{<i,*}, z_{\le i}). \tag{29}$$

- **MelNet** To represent the model introduced in this work, we use the probabilistic model described in Section 3, as well as a Gaussian variant which simply replaces the GMM with a univariate Gaussian.

	Unconditional	Text-to-Speech
Diagonal Gaussian	$= -1.44$	$= -1.56$
VAE: Global z	≤ -1.52	≤ -1.65
VAE: Local z	≤ -1.92	≤ -1.95
MelNet: Gaussian	$= -2.29$	$= -2.31$
MelNet: GMM	$= -2.32$	$= -2.33$

Table 3. Negative log-likelihood (nats/dim; lower is better) for different probabilistic models on the tasks of unconditional and text-conditional speech generation on the Blizzard dataset.

We constrain each model to use roughly the same number of parameters and briefly tune hyperparameters to ensure each model is reasonably representative of the potential of each probabilistic model. We found that the variation resulting from hyperparameters was relatively small in comparison to the margins between different probabilistic models. Further details for these models can be found in Appendix A.3.

Results shown in Table 3 demonstrate that fine-grained autoregressive model used by MelNet can greatly improve density estimates for both unconditional speech generation and TTS. The results also demonstrate that the unimodality and independence assumptions made by existing TTS models are detrimental to density estimates. Conditioning on latent variables relaxes these independence assumptions and improves performance, though density estimates by VAE models are still inferior to a full autoregressive factorization. Furthermore, even with a fine-grained factorization, it is beneficial to utilize a multimodal distribution to model the conditional distribution over each element.

8. Related Work

The predominant line of research regarding generative models for audio has been directed towards modelling time-domain waveforms with autoregressive models [47, 34, 26]. WaveNet is a competitive baseline for audio generation, and as such, is used for comparison in many of our experiments. However, we note that the contribution of our work is in many ways complementary to that of WaveNet. MelNet is more proficient at capturing high-level structure, whereas WaveNet is capable of producing higher-fidelity audio. Several works have demonstrated that time-domain models can be used to invert spectral representations to high-fidelity audio [42, 38, 3], suggesting that MelNet could be used in concert with time-domain models such as WaveNet.

In this work, we tackle the problem of jointly learning global and local structure in an end-to-end manner. This is in contrast to various works which circumvent the problem of capturing high-level structure by conditioning waveform generation on intermediate features. Notable such examples include the application of WaveNet to the tasks of text-to-speech [47, 50, 26, 6] and MIDI-conditional music generation [19, 33]. In the case of TTS, WaveNet depends on a traditional TTS pipeline to produce finely annotated linguistic features (phones, syllables, stress, etc.) as well as pitch and timing information. In the case of MIDI-conditional music generation, WaveNet relies upon a symbolic music representation (MIDI) which contains the pitch, volume, and timing of notes. These approaches require datasets with annotated features and are dependent upon human knowledge to determine appropriate domain-specific representations. In contrast to these approaches, MelNet does not require any intermediate supervision. MelNet is capable of learning TTS in an entirely end-to-end manner, whereas waveform models have not yet demonstrated the capacity to perform TTS without the assistance of intermediate linguistic features. Additionally, we demonstrate that MelNet uncovers high-level musical structure as well as two-stage models that separately model intermediate MIDI representations [19].

Dieleman et al. [11] and van den Oord et al. [51] capture long-range dependencies in waveforms by utilizing a hierarchy of autoencoders. This approach requires multiple stages of models which must be trained sequentially, whereas the multiscale approach in this work can be parallelized over tiers. Additionally, these approaches do not directly optimize the data likelihood, nor do they admit tractable marginalization over the latent codes. We also note that the modelling techniques devised in these works can be broadly applied to autoregressive models such as ours, making their contributions largely complementary to ours.

Recent works have used generative adversarial networks (GANs) [14] to model both waveforms and spectral representations [12, 13]. As with image generation, it remains unclear whether GANs capture all modes of the data distribution. Furthermore, these approaches are restricted to generating fixed-duration segments of audio, which precludes their usage in many audio generation tasks.

Many existing end-to-end TTS models are designed to generate a single high-quality sample for a given text [2, 43, 53, 37, 44]. MelNet instead focuses on modelling the full breadth of the conditional distribution of audio given text. We use the task of density estimation to demonstrate that MelNet captures this distribution better than probabilistic models that are commonly used by existing TTS systems, and we show that unimodality and independence assumptions made by existing TTS models are overly restrictive. Utilizing a more flexible probabilistic model allows MelNet to generate spectrograms with realistic textures without oversmoothing or blurring. This enables generated spectrograms to be directly inverted to high-fidelity audio using classical spectrogram inversion algorithms, whereas existing spectrogram models which produce audio of comparable quality rely on neural vocoders to correct for oversmoothing [43, 42, 37].

The network architecture used for MelNet is heavily influenced by recent advancements in deep autoregressive models for images. Theis and Bethge [45] introduced an LSTM architecture for autoregressive modelling of 2D images and van den Oord et al. [49] introduced PixelRNN and PixelCNN and scaled up the models to handle the modelling of natural images. Subsequent works in autoregressive image modelling have steadily improved state-of-the-art for image density estimation [48, 41, 36, 5, 7]. We draw inspiration from many of these models, and ultimately design a recurrent architecture of our own which is suitable for modelling spectrograms rather than images.

We use a multidimensional recurrence in both the time-delayed stack and the upsampling tiers to extract features from two-dimensional inputs. Our multidimensional recurrence is effectively 'factorized' as it independently applies one-dimensional RNNs across each dimension. This approach differs from the tightly coupled multidimensional recurrences used by MDRNNs [17, 16] and Grid LSTMs [25] and more closely resembles the approach taken by ReNet [52]. Our approach allows for efficient training as we can extract features from an $M \times N$ grid in $\max(M, N)$ sequential recurrent steps rather than the $M + N$ sequential steps required for tightly coupled recurrences. Additionally, our approach enables the use of highly optimized one-dimensional RNN implementations.

Various approaches to image generation have succeeded in generating high-resolution, globally coherent images with hundreds of thousands of dimensions [27, 39, 31]. The methods introduced in these works are not directly transferable to waveform generation, as they exploit spatial properties

of images which are absent in one-dimensional audio signals. However, these methods are more straightforwardly applicable to two-dimensional representations such as spectrograms. Of particular relevance to our work are approaches which combine autoregressive models with multiscale modelling [49, 9, 39, 35]. We demonstrate that the benefits of a multiscale autoregressive model extend beyond the task of image generation, and can be used to generate high-resolution, globally coherent spectrograms.

9. Conclusion

We have introduced MelNet, a generative model for spectral representations of audio. MelNet combines a highly expressive autoregressive model with a multiscale modelling scheme to generate high-resolution spectrograms with realistic structure on both local and global scales. In comparison to previous works which model time-domain signals directly, MelNet is particularly well-suited to model long-range temporal dependencies. Experiments show promising results on a diverse set of tasks, including unconditional speech generation, music generation, and text-to-speech synthesis.

Acknowledgements

We thank Kyle Kastner for reviewing a draft of this paper and providing helpful feedback.

References

[1] Kei Akuzawa, Yusuke Iwasawa, and Yutaka Matsuo. Expressive speech synthesis via modeling expressions with variational autoencoder. *arXiv preprint arXiv:1804.02135*, 2018.

[2] Sercan O Arik, Mike Chrzanowski, Adam Coates, Gregory Diamos, Andrew Gibiansky, Yongguo Kang, Xian Li, John Miller, Andrew Ng, Jonathan Raiman, et al. Deep voice: Real-time neural text-to-speech. *arXiv preprint arXiv:1702.07825*, 2017.

[3] Sercan Ö Arık, Heewoo Jun, and Gregory Diamos. Fast spectrogram inversion using multi-head convolutional neural networks. *IEEE Signal Processing Letters*, 26(1):94–98, 2019.

[4] Christopher M Bishop. Mixture density networks. Technical report, Citeseer, 1994.

[5] Xi Chen, Nikhil Mishra, Mostafa Rohaninejad, and Pieter Abbeel. Pixelsnail: An improved autoregressive generative model. *arXiv preprint arXiv:1712.09763*, 2017.

[6] Yutian Chen, Yannis Assael, Brendan Shillingford, David Budden, Scott Reed, Heiga Zen, Quan Wang, Luis C Cobo, Andrew Trask, Ben Laurie, et al. Sample efficient adaptive text-to-speech. *arXiv preprint arXiv:1809.10460*, 2018.

[7] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

[8] Joon Son Chung, Arsha Nagrani, and Andrew Zisserman. Voxceleb2: Deep speaker recognition. *arXiv preprint arXiv:1806.05622*, 2018.

[9] Ryan Dahl, Mohammad Norouzi, and Jonathon Shlens. Pixel recursive super resolution. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5439–5448, 2017.

[10] Rémi Decorsière, Peter L Søndergaard, Ewen N MacDonald, and Torsten Dau. Inversion of auditory spectrograms, traditional spectrograms, and other envelope representations. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(1):46–56, 2015.

[11] Sander Dieleman, Aaron van den Oord, and Karen Simonyan. The challenge of realistic music generation: modelling raw audio at scale. In *Advances in Neural Information Processing Systems*, pages 7999–8009, 2018.

[12] Chris Donahue, Julian McAuley, and Miller Puckette. Synthesizing audio with generative adversarial networks. *arXiv preprint arXiv:1802.04208*, 2018.

[13] Jesse Engel, Kumar Krishna Agrawal, Shuo Chen, Ishaan Gulrajani, Chris Donahue, and Adam Roberts. Gansynth: Adversarial neural audio synthesis. 2018.

[14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[15] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

[16] Alex Graves and Jürgen Schmidhuber. Offline handwriting recognition with multidimensional recurrent neural networks. In *Advances in neural information processing systems*, pages 545–552, 2009.

[17] Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. Multi-dimensional recurrent neural networks. In *International conference on artificial neural networks*, pages 549–558. Springer, 2007.

[18] Daniel Griffin and Jae Lim. Signal estimation from modified short-time fourier transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 32 (2):236–243, 1984.

[19] Curtis Hawthorne, Andriy Stasyuk, Adam Roberts, Ian Simon, Cheng-Zhi Anna Huang, Sander Dieleman, Erich Elsen, Jesse Engel, and Douglas Eck. Enabling factorized piano music modeling and generation with the maestro dataset. *arXiv preprint arXiv:1810.12247*, 2018.

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[21] François Hernandez, Vincent Nguyen, Sahar Ghannay, Natalia Tomashenko, and Yannick Esteve. Tedlium 3: twice as much data and corpus repartition for experiments on speaker adaptation. *arXiv preprint arXiv:1805.04699*, 2018.

[22] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[23] Wei-Ning Hsu, Yu Zhang, Ron J Weiss, Heiga Zen, Yonghui Wu, Yuxuan Wang, Yuan Cao, Ye Jia, Zhifeng Chen, Jonathan Shen, et al. Hierarchical generative modeling for controllable speech synthesis. *arXiv preprint arXiv:1810.07217*, 2018.

[24] Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Noam Shazeer, Curtis Hawthorne, Andrew M Dai, Matthew D Hoffman, and Douglas Eck. Music transformer: Generating music with long-term structure. *arXiv preprint arXiv:1809.04281*, 2018.

[25] Nal Kalchbrenner, Ivo Danihelka, and Alex Graves. Grid long short-term memory. *arXiv preprint arXiv:1507.01526*, 2015.

[26] Nal Kalchbrenner, Erich Elsen, Karen Simonyan, Seb Noury, Norman Casagrande, Edward Lockhart, Florian Stimberg, Aaron van den Oord, Sander Dieleman, and Koray Kavukcuoglu. Efficient neural audio synthesis. *arXiv preprint arXiv:1802.08435*, 2018.

[27] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

[28] Simon King. The blizzard challenge 2011, 2011.

[29] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[30] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[31] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, pages 10236–10245, 2018.

[32] Jinyu Li, Abdelrahman Mohamed, Geoffrey Zweig, and Yifan Gong. Exploring multidimensional lstms for large vocabulary asr. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 4940–4944. IEEE, 2016.

[33] Rachel Manzelli, Vijay Thakkar, Ali Siahkamari, and Brian Kulis. Conditioning deep generative raw audio models for structured automatic music. *arXiv preprint arXiv:1806.09905*, 2018.

[34] Soroush Mehri, Kundan Kumar, Ishaan Gulrajani, Rithesh Kumar, Shubham Jain, Jose Sotelo, Aaron Courville, and Yoshua Bengio. Samplernn: An unconditional end-to-end neural audio generation model. *arXiv preprint arXiv:1612.07837*, 2016.

[35] Jacob Menick and Nal Kalchbrenner. Generating high fidelity images with subscale pixel networks and multidimensional upscaling. *arXiv preprint arXiv:1812.01608*, 2018.

[36] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Łukasz Kaiser, Noam Shazeer, and Alexander Ku. Image transformer. *arXiv preprint arXiv:1802.05751*, 2018.

[37] Wei Ping, Kainan Peng, and Jitong Chen. Clarinet: Parallel wave generation in end-to-end text-to-speech. *arXiv preprint arXiv:1807.07281*, 2018.

[38] Ryan Prenger, Rafael Valle, and Bryan Catanzaro. Waveglow: A flow-based generative network for speech synthesis. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3617–3621. IEEE, 2019.

[39] Scott Reed, Aäron van den Oord, Nal Kalchbrenner, Sergio Gómez Colmenarejo, Ziyu Wang, Dan Belov, and Nando de Freitas. Parallel multiscale autoregressive density estimation. *arXiv preprint arXiv:1703.03664*, 2017.

[40] Tara N Sainath and Bo Li. Modeling time-frequency patterns with lstm vs. convolutional architectures for lvcsr tasks. In *INTERSPEECH*, pages 813–817, 2016.

[41] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *arXiv preprint arXiv:1701.05517*, 2017.

[42] Jonathan Shen, Ruoming Pang, Ron J Weiss, Mike Schuster, Navdeep Jaitly, Zongheng Yang, Zhifeng Chen, Yu Zhang, Yuxuan Wang, Rj Skerrv-Ryan, et al. Natural tts synthesis by conditioning wavenet on mel spectrogram predictions. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4779–4783. IEEE, 2018.

[43] Jose Sotelo, Soroush Mehri, Kundan Kumar, Joao Felipe Santos, Kyle Kastner, Aaron Courville, and Yoshua Bengio. Char2wav: End-to-end speech synthesis. 2017.

[44] Yaniv Taigman, Lior Wolf, Adam Polyak, and Eliya Nachmani. Voiceloop: Voice fitting and synthesis via a phonological loop. 2018.

[45] Lucas Theis and Matthias Bethge. Generative image modeling using spatial lstms. In *Advances in Neural Information Processing Systems*, pages 1927–1935, 2015.

[46] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

[47] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[48] Aaron van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with pixelcnn decoders. In *Advances in Neural Information Processing Systems*, pages 4790–4798, 2016.

[49] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*, 2016.

[50] Aaron van den Oord, Yazhe Li, Igor Babuschkin, Karen Simonyan, Oriol Vinyals, Koray Kavukcuoglu, George van den Driessche, Edward Lockhart, Luis C Cobo, Florian Stimberg, et al. Parallel wavenet: Fast high-fidelity speech synthesis. *arXiv preprint arXiv:1711.10433*, 2017.

[51] Aaron van den Oord, Oriol Vinyals, et al. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, pages 6306–6315, 2017.

[52] Francesco Visin, Kyle Kastner, Kyunghyun Cho, Matteo Matteucci, Aaron Courville, and Yoshua Bengio. Renet: A recurrent neural network based alternative to convolutional networks. *arXiv preprint arXiv:1505.00393*, 2015.

[53] Yuxuan Wang, RJ Skerry-Ryan, Daisy Stanton, Yonghui Wu, Ron J Weiss, Navdeep Jaitly, Zongheng Yang, Ying Xiao, Zhifeng Chen, Samy Bengio, et al. Tacotron: Towards end-to-end speech synthesis. *arXiv preprint arXiv:1703.10135*, 2017.

A. Experimental Details

A.1. Human Evaluation

For each of the three unconditional audio generation tasks, we generated 50 ten-second samples from WaveNet and 50 ten-second samples from MelNet. Participants were shown an anonymized, randomly-drawn sample from each model and instructed to "select the sample which has more coherent long-term structure." We collected 50 human evaluations for each task.

A.2. WaveNet Baseline

The human evaluation experiments require samples from a baseline WaveNet model. For the Blizzard and VoxCeleb2 datasets, we use our own reimplementation. Our WaveNet model uses 8-bit μ-law encoding and models each sample with a discrete distribution. Each model is trained for 150,000 steps. We use the Adam optimizer [29] with a learning rate of 0.001 and batch size of 32. Additional hyperparameters are reported in Table 4.

	Blizzard	VoxCeleb2
Sample Rate (Hz)	22,050	16,000
Layers	50	60
Kernel Size	3	3
Dilation (at layer i)	$2^{i \bmod 10}$	$2^{i \bmod 10}$
Receptive Field (samples)	10,240	12,288
Receptive Field (ms)	464	768
Max Sample Duration (s)	2	2

Table 4. WaveNet hyperparameters.

We do not use our WaveNet implementation for human evaluation on the MAESTRO dataset. The authors that introduce this dataset provide roughly 2 minutes of audio samples on their website[2] for both unconditional WaveNet and Wave2Midi2Wave models. We generate 50 random ten-second slices from these 2 minutes and directly use them for the human evaluations.

A.3. Density Estimation

We use the Blizzard dataset for all density estimation experiments. We use low-resolution spectrograms which are typically used by existing TTS systems. These spectrograms have 80 mel channels and are computed with a STFT hop size of 512 and STFT window size of $6 \cdot 512$.

All baseline models use similar network architectures which are composed of multi-layer LSTMs with residual connections. The baseline models use 1024 hidden units whereas the MelNet models use 512 hidden units. The MelNet models do not use multiscale orderings. All models have 8-layer autoregressive decoders and the VAE models have an additional 4-layer inference network. The global VAE model uses a 512-dimensional latent vector and the local VAE model uses a sequence of 32-dimensional latent vectors. VAE models are trained with KL annealing over the first epoch. When evaluating density estimates for text-to-speech synthesis, all models use the attention mechanism described in Section 5.

[2] https://goo.gl/magenta/maestro-examples