Synthesizing Audio with Generative Adversarial Networks

Chris Donahue [1] **Julian McAuley** [2] **Miller Puckette** [1]

Abstract

While Generative Adversarial Networks (GANs) have seen wide success at the problem of synthesizing realistic images, they have seen little application to audio generation. Unlike for images, a barrier to success is that the best *discriminative* representations for audio tend to be non-invertible, and thus cannot be used to synthesize listenable outputs. In this paper, we introduce WaveGAN, a first attempt at applying GANs to raw audio synthesis in an unsupervised setting. Our experiments on speech demonstrate that WaveGAN can produce intelligible words from a small vocabulary of human speech, as well as synthesize audio from other domains such as bird vocalizations, drums, and piano. Qualitatively, we find that human judges prefer the generated examples from WaveGAN over those from a method which naïvely apply GANs on image-like audio feature representations.

1. Introduction

Synthesizing audio for specific domains has many practical applications such as text-to-speech and music production. End-to-end, learning-based approaches have recently eclipsed the performance of production parametric systems in the area of text-to-speech (Wang et al., 2017). Such methods depend on access to large quantities of transcribed recordings, but do not take advantage of additional untranscribed audio that is often available. Unsupervised approaches may be able to reduce data requirements for these methods by learning to synthesize *a priori*. However, audio signals have high temporal resolution with periodic behavior over large windows, and relevant strategies must perform effectively in high dimensions.

Generative Adversarial Networks (Goodfellow et al., 2014) are unsupervised algorithms that have shown promise at generating high-dimensional signals. Since their introduction, GANs have been refined to generate images with increasing fidelity (Radford et al., 2016; Berthelot et al., 2017; Karras et al., 2018). Despite their prevalence for image applications, GANs have yet to be demonstrated capable of synthesizing audio in an unsupervised setting.

A naïve solution for applying GANs to audio would be to operate them on image-like *spectrograms*, i.e., time-frequency representations of audio. This practice of bootstrapping image-processing algorithms for audio tasks is commonplace in the discriminative setting (Hershey et al., 2017). In the generative setting however, this approach is problematic as the most perceptually-informed representations are non-invertible, and hence cannot be listened to without lossy estimations (Griffin & Lim, 1984) or learned inversion models (Shen et al., 2018). Recent work (Oord et al., 2016; Mehri et al., 2017) has shown that neural networks can be trained with autoregression to operate on *raw audio*. Such approaches are attractive as they dispense with engineered acoustic features. However, unlike with GANs, the autoregressive setting results in slow generation as output audio samples must be fed back into the model one at a time.

In this work, we investigate both time- and frequency-domain strategies for generating slices of audio with GANs.[1] With our frequency-domain approach (SpecGAN), we first design an appropriate spectrogram representation that allows for approximate inversion, and apply the two-dimensional deep convolutional GAN (DCGAN) method (Radford et al., 2016) to these spectrograms. In WaveGAN, our time-domain approach, we flatten the SpecGAN architecture to operate in one dimension, resulting in a model with the same number of parameters and numerical operations as its two-dimensional analog.

To evaluate WaveGAN, we propose a new standard task, generating spoken examples of digits “zero” through ”nine”. We design evaluation methodology based on scores from a pre-trained classifier and human judgements. Our experiments on this task demonstrate that both WaveGAN and SpecGAN can generate examples of speech that are intelligible to humans. On criteria of sound quality and speaker diversity, human judges indicate a preference for the examples from WaveGAN compared to those from SpecGAN.

[1]Department of Music, UC San Diego [2]Department of Computer Science, UC San Diego.

[1]Sound examples: https://goo.gl/oxGXAi
Interactive notebook: https://goo.gl/ChSPp9
Code released upon publication

2. GAN Preliminaries

GANs learn mappings from low-dimensional latent vectors $\boldsymbol{z} \in \mathcal{Z}$, i.i.d. samples from known prior P_Z, to points in the space of natural data $\mathcal{X}$. In their original formulation (Goodfellow et al., 2014), a generator $G : \mathcal{Z} \mapsto \mathcal{X}$ is pitted against a discriminator $D : \mathcal{X} \mapsto [0, 1]$ in a two-player minimax game. The generator is trained to minimize the following value function, while the discriminator is trained to maximize it:

$$V(D, G) = \mathbb{E}_{\boldsymbol{x} \sim P_X}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim P_Z}[\log(1 - D(G(\boldsymbol{z})))]. \quad (1)$$

In other words, D is trained to determine if an example is real (1) or fake (0), and G is trained to fool the discriminator into thinking its output is real. Goodfellow et al. (2014) demonstrate that their proposed training algorithm for Equation 1 equates to minimizing the Jensen-Shannon divergence between P_X, the real data distribution, and P_G, the implicit distribution of the generator when $\boldsymbol{z} \sim P_Z$.

In their original formulation, GANs are notoriously difficult to train, and prone to catastrophic failure cases such as mode collapse, in which the generator outputs a single result for all $\boldsymbol{z}$. Arjovsky et al. (2017) demonstrate that, under certain conditions, minimizing Jensen-Shannon divergence of P_X and P_G with gradient descent is ill-posed as it is discontinuous and provides no usable gradient. As a smoother alternative, they suggest minimizing the Wasserstein-1 distance between distributions

$$W(P_X, P_G) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x \sim P_X}[f(x)] - \mathbb{E}_{x \sim P_G}[f(x)] \quad (2)$$

where $\|f\|_L \leq 1 : \mathcal{X} \mapsto \mathbb{R}$ is the family of functions that are 1-Lipschitz.

To minimize Wasserstein distance, they suggest a GAN training algorithm (WGAN), similar to that of Goodfellow et al. (2014), for the following value function:

$$V_{\text{WGAN}}(D_w, G) = \mathbb{E}_{\boldsymbol{x} \sim P_X}[D_w(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{z} \sim P_Z}[D_w(G(\boldsymbol{z}))]. \quad (3)$$

With this formulation, $D_w : \mathcal{X} \mapsto \mathbb{R}$ is not trained to identify examples as real or fake, but instead is trained as a function that assists in computing the Wasserstein distance. Arjovsky et al. (2017) suggest weight clipping as a means of enforcing that D_w is 1-Lipschitz. As an alternative strategy, Gulrajani et al. (2017) replace weight clipping with a gradient penalty (WGAN-GP) that also enforces the constraint. They demonstrate that their WGAN-GP strategy can successfully train a variety of model configurations where other GAN losses fail.

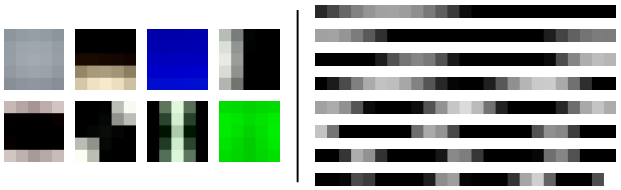

Figure 1. First eight principal components for 5x5 patches from natural images (**left**) versus those of length-25 audio slices from speech (**right**). Periodic patterns are unusual in natural images but a fundamental structure in audio.

3. WaveGAN

We motivate our design choices for WaveGAN by first discussing fundamental differences between audio and images.

3.1. Audio versus images

From a data storage perspective, time-domain audio samples are similar to the pixels in an image. However, these two types of signals have inherently different traits. In Figure 1, we show the first eight principal components for patches from natural images versus slices from speech. While the principal components of images generally capture intensity, gradient, and edge characteristics, those from audio are periodic functions that decompose the audio into constituent bands. In general, natural audio signals are more likely to exhibit periodicity than natural images.

As a consequence, correlations across large windows are commonplace in audio. For example, in a waveform sampled at 16 kHz, a 440 Hz sinusoid (the musical note A4) takes over 36 samples to complete a single cycle. This suggests that filters with large receptive fields are needed to process raw audio. This same idea motivated Oord et al. (2016) in their design of WaveNet, which uses dilated convolutions to exponentially increase the model's effective receptive field with layer depth.

3.2. WaveGAN architecture

We base our WaveGAN architecture off of DCGAN (Radford et al., 2016) which popularized usage of GANs for image synthesis. Motivated by our above discussion, we modify the transposed convolution operation from the DCGAN generator to widen its receptive field. Specifically, we use longer one-dimensional filters of length 25 instead of two-dimensional filters of size 5x5, and we upsample by a factor of 4 instead of 2 at each layer (Figure 2). We modify the discriminator in a similar way, using length-25 filters in one dimension and increasing stride from 2 to 4. These changes result in WaveGAN having the same number of parameters, numerical operations, and output dimensionality as DCGAN.

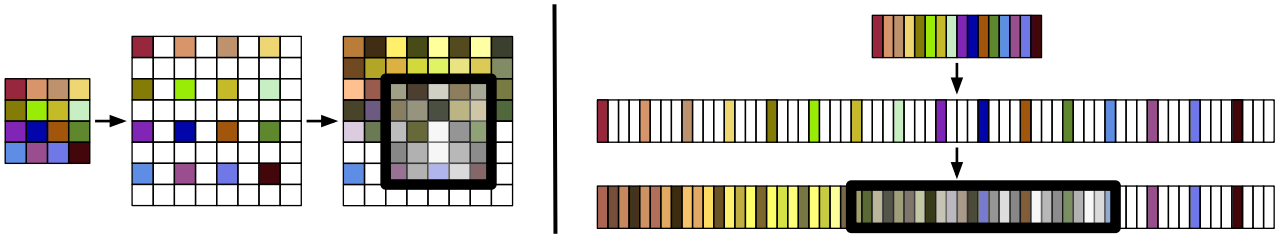

Figure 2. Depiction of the transposed convolution operation for the first layers of the DCGAN (Radford et al., 2016) (**left**) and WaveGAN (**right**) generators. DCGAN uses small (5x5), two-dimensional filters while WaveGAN uses longer (length-25), one-dimensional filters and a larger upsampling factor. The two operations have the same number of parameters and numerical operations.

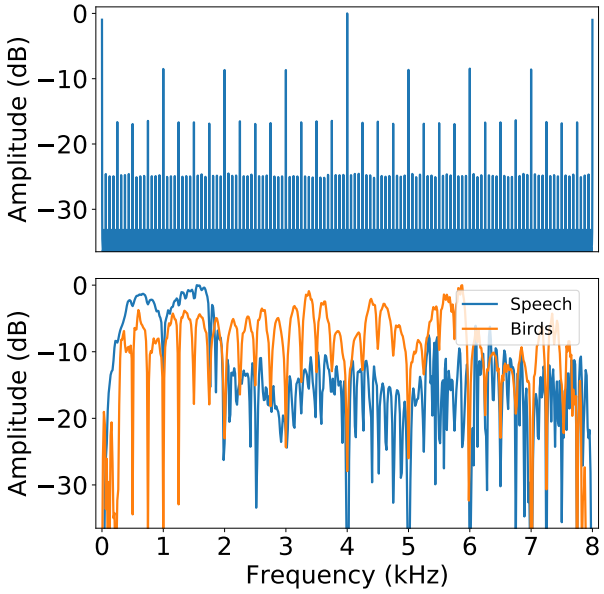

Figure 3. (**Top**): Average impulse response for 1000 random initializations of the WaveGAN generator. (**Bottom**): Response of learned post-processing filters for speech and bird vocalizations. Both filters reject frequencies corresponding to noise byproducts created by the generative procedure (top). The filter for speech boosts signal in prominent speech bands (below 2 kHz), while the filter for bird vocalizations (which are more uniformly-distributed in frequency) simply reduces noise presence.

Because DCGAN outputs 64x64 pixel images — equivalent to just 4096 audio samples — we add one additional layer to the model resulting in 16384 samples, slightly more than one second of audio at 16 kHz. While 16k samples is a sufficient length for certain sound domains (e.g. sound effects, voice commands), generalization to longer output is an avenue for future work. We requantize the real data from 16-bit PCM to 32-bit floating point, and our generator similarly outputs real-valued samples. Our experiments observe and generate monaural audio, though a trivial modification of the architecture would allow for operation on audio of arbitrary channel depth. A complete description of our model can be found in Appendix A.

3.3. Checkerboard artifacts in audio versus images

Generative models that upsample by transposed convolution are known to produce characteristic "checkerboard" artifacts in images (Odena et al., 2016), artifacts with particular spatial periodicities. The discriminator of image-generating GANs can learn to reject images with these artifacts because they are uncommon in real data (as discussed in Section 3.1). However, in the audio domain, the discriminator might not have such luxury as these artifacts correspond to frequencies which might rightfully appear in the real data.

To characterize analogous artifacts in WaveGAN, we measure its impulse response by randomly initializing it 1000 times and passing unit impulses to its first convolutional layer. In Figure 3, we plot the average of these responses in the frequency domain. The response has sharp peaks at linear multiples of the sample rates of each convolutional layer (4 kHz, 1 kHz, 250 Hz, etc.). This is in agreement with our informal observation of results from WaveGAN, which often have a pitched noise close to the musical note B (247×2^n Hz). In the remainder of this section, we discuss several methods which may help to mitigate this noise.

3.4. Learned post-processing filters

We experiment with adding a post-processing filter to the generator, giving WaveGAN a simple mechanism to filter out undesirable frequencies created by the generative process. This filter has a long window (512 samples) allowing it to represent intricate transfer functions, and the weights of the filter are learned as part of the generator's parameters. In Figure 3, we compare the post-processing filters that WaveGAN learns for human speech and bird vocalizations. The filters boost signal in regions of the frequency spectrum that are most prominent in the real data domain, and introduce notches at bands that are artifacts of the generative procedure as discussed in the previous section.

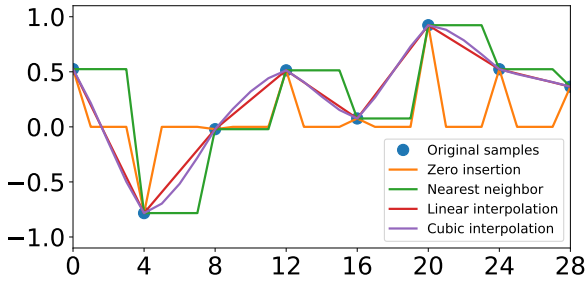

Figure 4. Depiction of the upsampling strategy used by transposed convolution (zero insertion) and other strategies which mitigate aliasing: nearest neighbor, linear and cubic interpolation.

3.5. Upsampling procedure

Transposed convolution upsamples signals by inserting zeros in between samples and applying a learned filterbank. This operation introduces aliased frequencies, copies of pre-existing frequencies shifted by multiples of the previous Nyquist rate, into the upsampled signal. While aliased frequencies are usually seen as undesirable artifacts of a bad upsampling procedure, in the generative setting their existence may be crucial for producing fine-grained details in the output. We experiment with three other upsampling strategies in WaveGAN: nearest-neighbor, linear and cubic interpolation, all of which attenuate aliased frequencies. In Figure 4, we compare these strategies visually.

3.6. Phase shuffle

As discussed in Section 3.3, the transposed convolution procedure of the WaveGAN generator produces characteristic artifacts. The discriminator may have difficulty distinguishing these artifacts from periodicities in the real data by their frequency alone. However, if the artifact frequencies always occur at a particular phase, and the real data contains these frequencies at many phases, then the discriminator could learn a trivial solution that rejects the generated samples. This may inhibit the overall optimization problem.

To discourage the discriminator from learning such a solution, we propose the *phase shuffle* operation (with hyperparameter n) which randomly perturbs the phase of each layer's activations by $-n$ to n samples before input to the next layer (Figure 5). We use reflection padding to fill in the missing samples. Phase shuffling may benefit GANs designed for image generation, though in this work we only evaluate it within the audio context.

3.7. Batch normalization

As a pilot study for this work, we trained WaveGAN using the DCGAN, LSGAN, WGAN and WGAN-GP losses (Radford et al., 2016; Mao et al., 2017; Arjovsky et al., 2017; Gulrajani et al., 2017), both with and without batch normalization (Ioffe & Szegedy, 2015). Only two of these configurations, DCGAN with batch normalization and WGAN-GP without batch normalization, resulted in reasonable output. We compare these two combinations in our experimentation.

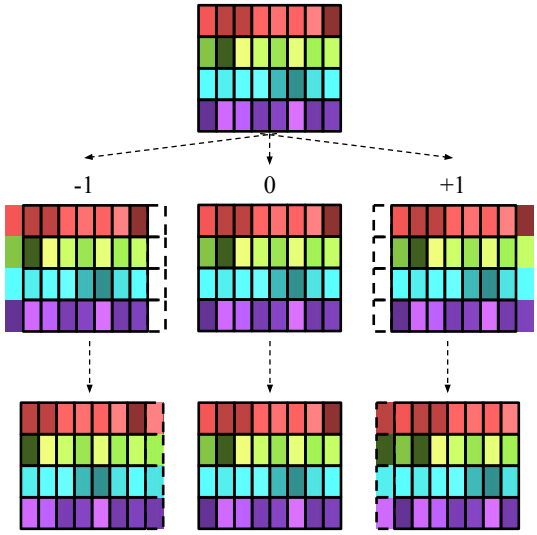

Figure 5. At each layer of the WaveGAN discriminator, the phase shuffle operation randomizes the phase of each channel by $[-n, n]$ samples, filling in the missing samples (dashed outlines) by reflection. This figure shows possible outcomes for 4 channels ($n = 1$).

4. SpecGAN

While a minority of recent research in discriminative audio classification tasks has used raw audio input (Sainath et al., 2015; Lee et al., 2017), most of these approaches operate on spectrogram representations of audio. A generative model may also benefit from operating in such a space. However, commonly-used representations in the discriminative setting aggregate frequencies into logarithmically-spaced bins and discard phase information, rendering them uninvertible.

With SpecGAN, our frequency-domain audio generation model, we design a spectrogram representation that is both well-suited to GANs designed for image generation and *can* be approximately inverted. Additionally, to facilitate direct comparison, our representation is designed to use the same dimensionality per unit of time as WaveGAN (16384 samples yield a 128x128 spectrogram).

To process audio into suitable spectrograms, we first perform the short-time Fourier transform with 16 ms windows and 8 ms stride, resulting in 129 frequency bins linearly spaced from 0 to 8 kHz. We take the magnitude of the resultant spectra and scale amplitude values logarithmically to better-align with human perception. We then normalize each frequency bin to have zero mean and unit variance, and discard the highest frequency bin. Finally, we clip the

spectra to 3 standard deviations and rescale to $[-1, 1]$.

Once our dataset has been processed into this format, we operate the DCGAN (Radford et al., 2016) algorithm on the resultant spectra. As with WaveGAN, this requires adding an extra layer to the original architecture (Appendix A). To render the resultant generated spectrograms as waveforms, we first invert the steps of spectrogram preprocessing described above, resulting in linear-amplitude magnitude spectra. We then employ the iterative Griffin-Lim algorithm (Griffin & Lim, 1984) with 16 iterations to estimate phase and produce 16384 samples of audio.

5. Experiments

Our primary experimentation focuses on the *Speech Commands Dataset* (Warden, 2017). This dataset consists of many speakers recording individual words in uncontrolled recording conditions. To be comparable to datasets often studied in image generation (LeCun et al., 1998), we propose the *Speech Commands Zero Through Nine* (SC09) subset, which reduces the vocabulary of the dataset to ten words: the digits "zero" through "nine".

These ten words represent a variety of phonemes and two consist of multiple syllables. Each recording is one second in length, and we do not attempt to align the words in time. There are 1850 utterances of each word in the training set, resulting in 5.3 hours of speech. The wide variety of alignments, speakers and recording conditions make this a challenging dataset from a generative perspective.

Our baseline configuration for WaveGAN excludes both post-processing filters and batch normalization, does not use phase shuffle, and upsamples by inserting zeros (normal transposed convolution). We also run experiments that modify this baseline configuration to use phase shuffle with $n \in [2, 4]$, upsample by nearest-neighbor, linear, and cubic strategies, include post-processing filters and use batch normalization. Based on our pilot study (Section 3.7), we use the DCGAN loss (Radford et al., 2016) when training the model with batch normalization, and otherwise we use WGAN-GP (Gulrajani et al., 2017).

For the SC09 dataset, we compare the performance of WaveGAN to that of SpecGAN. We also experiment with phase shuffling in SpecGAN (on the time axis only). Shifting the phase of the spectrogram by one timestep equates to a shift of 128 time-domain samples; as such we limit our experimentation of phase shuffle to $n = 1$ for SpecGAN.

We also perform experiments on four other datasets with different sonic characteristics from SC09 (Figure 6):

1. *Large vocab speech* (2.4 hours): Multiple speakers, clean recordings (TIMIT (Garofolo et al., 1993))
2. *Bird vocalizations* (12.2 hours): In-the-wild recordings of many species (Boesman, 2018)
3. *Single drum hits* (0.7 hours): Drum machine samples for kicks, snares, toms, and cymbals
4. *Piano* (0.3 hours): Professional performer playing a variety of Bach compositions

We train our networks using batches of size 64 on a single NVIDIA P100 GPU. During our quantitative evaluation of SC09 (discussed below), our WaveGAN networks converge by their early stopping criteria within four days (200k iterations, equivalent to 700 epochs). Our SpecGAN networks converge more quickly, within two days (175 epochs). On the other four datasets, we train WaveGAN for 200k iterations representing nearly 300 epochs for the largest dataset. Unlike with autoregressive methods (Oord et al., 2016; Mehri et al., 2017), generation with WaveGAN is fully parallel and can produce an hour of audio in less than two seconds. We list all hyperparameters in Appendix B.

6. Evaluation methodology

Evaluation of generative models is a fraught topic. Theis et al. (2016) demonstrate that quantitative measures of sample quality are poorly correlated with each other and human judgement. We use several quantitative evaluation metrics for hyperparameter validation and discussion, and also evaluate our best models with human judges.

6.1. Inception score

Salimans et al. (2016) propose the *inception score*, which uses a pre-trained Inception classifier (Szegedy et al., 2016) to measure both the diversity and semantic discriminability of generated images, finding that the measure correlates well with human judgement.

Given model scores $P(\boldsymbol{y} \mid \boldsymbol{x})$ with marginal $P(\boldsymbol{y})$, inception score is defined as $\exp(\mathbb{E}_{\boldsymbol{x}} D_{\mathrm{KL}}(P(\boldsymbol{y} \mid \boldsymbol{x}) || P(\boldsymbol{y})))$, and is estimated over a large number of samples (e.g. 50k). For data with n classes, this measure ranges from 1 to n, and is maximized when the model is completely confident about each prediction but predicts each label equally often. We will use this measure as our primary quantitative evaluation method and early stopping criteria.

To measure inception score, we train an audio classifier on SC09. Our classifier first computes a short-time Fourier transform of the input audio with 64 ms windows and 8 ms stride. This representation is projected to 128 frequency bins equally spaced on the Mel scale (Stevens et al., 1937) from 40 Hz to 7800 Hz. Amplitudes are scaled logarithmically and normalized so that each bin has zero mean and unit variance. We process this perceptually-informed representa-

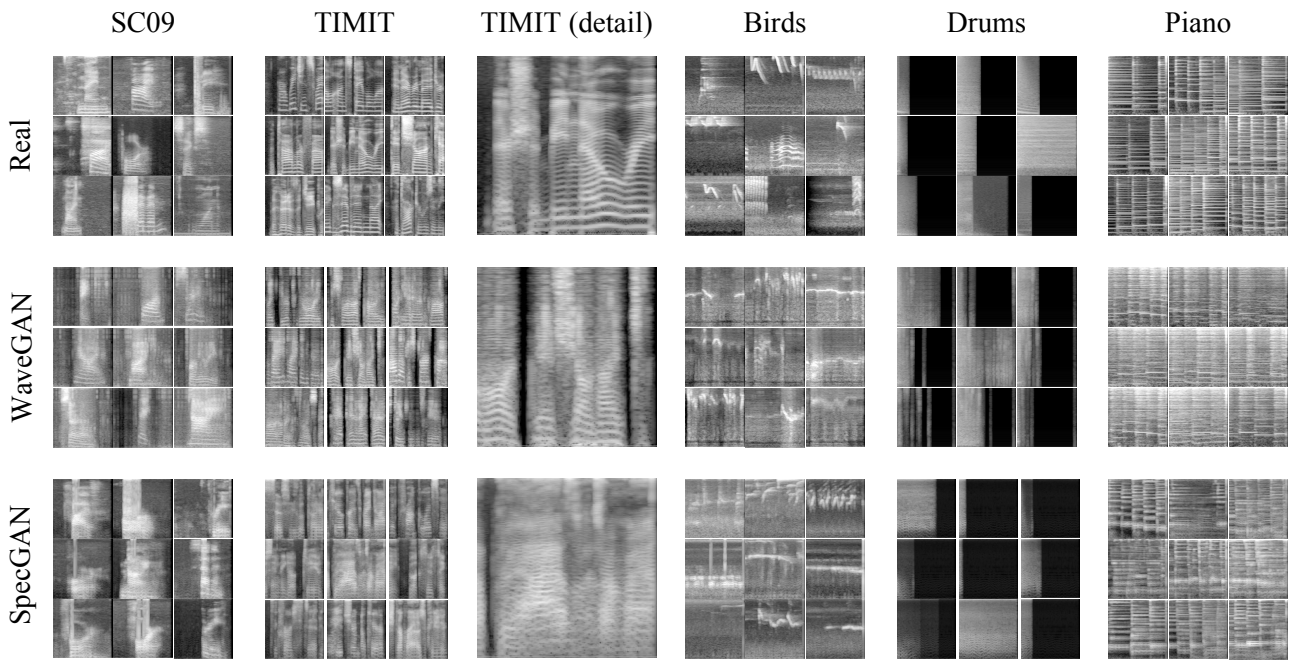

Figure 6. **Top**: Random samples from each of the five datasets used in this study, illustrating the wide variety of spectral characteristics. **Middle**: Random samples generated by WaveGAN for each domain. WaveGAN operates in the time domain but results are displayed here in the frequency domain for visual comparison. **Bottom**: Random samples generated by SpecGAN for each domain.

tion with four layers of convolution and pooling, and project the resultant features to a *softmax* layer with 10 classes. We perform early stopping using cross-entropy on the SC09 validation set, and the resultant model achieves 93% accuracy on the test set.

6.2. Nearest neighbor comparison

Inception score has two trivial failure cases in which a poor generative model can achieve a high score. Firstly, a generative model that outputs a single example of each class with uniform probability will be assigned a high score. Secondly, a generative model that overfits the training data will achieve a high score simply by outputting examples from the training data.

We use two indicators metrics to determine if a high inception score has been caused by either of these two cases. Our first indicator, $|D|_{\text{self}}$, measures the average Euclidean distance of a set of 1k examples to their nearest neighbor within the set (other than itself). A higher $|D|_{\text{self}}$ indicates higher diversity amongst samples. Because measuring Euclidean distance in time-domain audio poorly represents human perception, we evaluate distance in the same frequency-domain representation as our classifier from Section 6.1.

Our second indicator, $|D|_{\text{train}}$, measures the average Euclidean distance of 1k examples to their nearest neighbor in the real training data. If the generative model simply produces examples from the training set, this measure will be 0. We report $|D|_{\text{train}}$ and $|D|_{\text{self}}$ relative to those of the real test set.

6.3. Measuring upsampling noise

As discussed in Section 3.3, the transposed convolution upsampling procedure results in a particular pitched noise in WaveGAN outputs. To measure the amount by which various hyperparameters affect the model's ability to mitigate this noise, we calculate the weighted average of the magnitude spectrum for 1k examples from each method, using the normalized magnitude response from Figure 3 as the weights. This value approximates the amount of upsampling noise in the output, though it also includes energy that should rightfully occur at these frequencies. Accordingly, we report noise values relative to that of the real data.

6.4. Qualitative human judgements

While inception score is a useful quantitative metric for hyperparameter validation, our ultimate goal is to produce examples that are semantically meaningful to humans. To this end, we measure the ability of human annotators on *Amazon Mechanical Turk* to label the generated audio.

Using our best WaveGAN and SpecGAN models as measured by inception score, we generate 100 examples for each digit (as labeled by our classifier from Section 6.1). In

Table 1. Quantitative results for SC09 experiments comparing real and generated data. A higher inception score suggests that semantic modes of the real data distribution have been captured. $|D|_{\text{self}}$ indicates the intra-dataset diversity relative to that of the real test data. $|D|_{\text{train}}$ indicates the distance between the dataset and the training set relative to that of the test data; a low value indicates a generative model that is overfit to the training data. Noise indicates the relative amount of upsampling noise (Figure 3).

Experiment	Inception score	$\|D\|_{\text{self}}$	$\|D\|_{\text{train}}$	Noise
Real (train)	9.18 ± 0.04	1.1	0.0	1.0
Real (test)	8.01 ± 0.24	1.0	1.0	1.0
Parametric	5.02 ± 0.06	0.7	1.1	0.2
WaveGAN	4.12 ± 0.03	1.4	2.0	1.2
+ Phase shuffle $n = 2$	4.67 ± 0.01	0.8	2.3	1.8
+ Phase shuffle $n = 4$	4.54 ± 0.03	1.0	2.3	1.5
+ Nearest neighbor	3.77 ± 0.02	1.8	2.6	1.4
+ Linear interpolation	2.88 ± 0.02	1.7	2.7	1.2
+ Cubic interpolation	2.60 ± 0.01	1.3	3.6	1.1
+ Post-processing	3.92 ± 0.03	1.4	2.9	1.0
+ DCGAN/BN	2.01 ± 0.01	0.9	4.3	0.0
+ Dropout	3.93 ± 0.03	1.0	2.6	1.6
SpecGAN	6.03 ± 0.04	1.1	1.4	1.5
+ Phase shuffle $n = 1$	3.71 ± 0.03	0.8	1.6	3.1

Table 2. Pairwise accuracy for human judges on SC09 data.

Metric	Real (test)	WaveGAN	SpecGAN
Human accuracy	.976	.943	.945

batches of ten questions, we ask annotators to determine which of two examples, one with the correct digit and one chosen at random from the other digits, sounds more like a target digit. We choose this scheme so that we can conduct the questionnaire with a redundancy factor of three, casting decisions as a majority vote among three judges when calculating accuracy.

After the ten questions, each annotator is asked to assign subjective values of 1 through 5 for criteria of sound quality, ease of intelligibility, and speaker diversity. We calculate the mean opinion score of these criteria and compare results in Figure 7.

7. Results and discussion

Results for our quantitative evaluation appear in Table 1. We also evaluate our metrics on the real training data, the real test data, and a version of SC09 generated by a parametric speech synthesizer (Buchner, 2017). We also compare to SampleRNN (Mehri et al., 2017) and two public implementations of WaveNet (Oord et al., 2016), but neither method produced competitive results (none were stronger than our weakest baseline), and we excluded them from further evaluation. One possible explanation for the poor performance

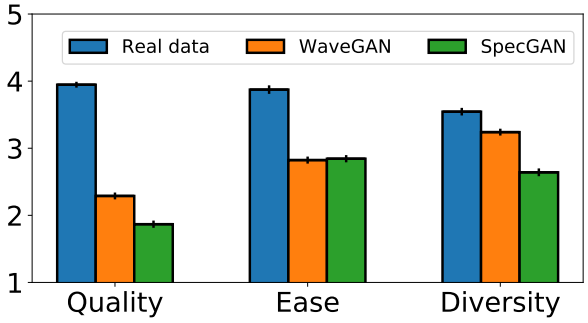

Figure 7. Mean opinion scores for *sample quality*, *ease of intelligibility* and *speaker diversity* from human judges. Error bars show standard error ($n = 300$).

of SampleRNN and WaveNet is that the single-word SC09 dataset is unlike the dense speech datasets that these methods were originally demonstrated on. Sound examples for all experiments, including SampleRNN and WaveNet, can be found at https://goo.gl/oxGXAi.

While the maximum inception score for SC09 is 10, any score higher than the test set score of 8 should be seen as evidence that a generative model has overfit. Our best WaveGAN model uses phase shuffle with $n = 2$ and achieves an inception score of 4.7. Adding either amount of phase shuffle to the discriminator improved the inception score over the baseline model. To determine if phase shuffle was improving the learning procedure simply by slowing down training, we also tried using 50% dropout in the discriminator's activations with fixed masks across timesteps. Dropout resulted in a lower inception score compared to the baseline model. More-sophisticated interpolation strategies yielded worse inception scores. Using the DCGAN loss with batch normalization produces quiet results that only occasionally resemble speech.

Most experiments produced $|D|_{\text{self}}$ (diversity) values higher than that of the test data, and all experiments produced $|D|_{\text{train}}$ (distance from training data) values higher than that of the test data. While these measures indicate that our generative models produce examples with statistics that deviate from those of the real data, neither metric indicates that the models achieve high inception scores by trivial solutions (Section 6.2).

While the WaveGAN model with a learned post-processing filter (Section 3.4) produced examples with an inception score below that of the baseline, it also produced examples with the same amount of noise as the real data. Qualitatively speaking, we found that adding a post-processing filter improved the perceived sound quality of the results especially for other domains such as bird vocalizations.

While examples from SpecGAN achieve higher inception score (6.0) than those from our best WaveGAN model (4.7),

human judges are able to label examples from the two models with identical accuracy (Table 2). However, on subjective criteria of sound quality and speaker diversity, humans indicate a significant preference for examples from WaveGAN (Figure 7). This result demonstrates the shortcomings of using measurements like inception score that employ neural networks as a proxy for human perception. It is possible that the poor qualitative ratings for examples from SpecGAN are primarily caused by the noisy Griffin-Lim procedure (Griffin & Lim, 1984) and not the generative process itself; investigation of more sophisticated reconstruction strategies is an avenue for future work.

Finally, we train our best WaveGAN and SpecGAN models (as measured by inception score on SC09) on the four other domains listed in Section 5. Results from these experiments appear in Figure 6. Somewhat surprisingly, we find that the frequency-domain spectra produced by WaveGAN (a time-domain method) are visually more consistent with the training data (e.g. in terms of sharpness) than those produced by SpecGAN. For TIMIT, a large-vocabulary speech dataset with many speakers, WaveGAN produces speech-like babbling (similar to results from unconditional autoregressive models (Oord et al., 2016)). On bird vocalizations, WaveGAN generates a variety of bird sounds but with more noise than the other domains. For drum sound effects, WaveGAN captures semantic modes such as kick and snare drums. On piano, WaveGAN produces musically-consonant motifs that, as with the training data, represent a variety of key signatures and rhythmic patterns.

8. Related work

Much of the work within generative modeling of audio is within the context of text-to-speech. Text-to-speech systems are primarily either *concatenative* or *parametric*. In concatenative systems, audio is generated by sequencing small, prerecorded portions of speech from a phonetically-indexed dictionary (Moulines & Charpentier, 1990; Hunt & Black, 1996). Parametric systems map text to salient parameters of speech, which are then synthesized by a vocoder (Dudley, 1939); see (Zen et al., 2009) for a comprehensive review. Some of these systems use learning-based approaches such as a hidden Markov models (Yoshimura, 2002; Tokuda et al., 2013), and separately-trained neural networks pipelines (Ling et al., 2015) to estimate speech parameters.

Recently, several researchers have investigated parametric speech synthesis with end-to-end neural network approaches that learn to produce vocoder features directly from text or phonetic embeddings (Arik et al., 2017; Ping et al., 2018; Sotelo et al., 2017; Wang et al., 2017; Shen et al., 2018). These vocoder features are synthesized to raw audio using off-the-shelf methods such as WORLD (Morise et al., 2016) and Griffin-Lim (Griffin & Lim, 1984), or trained neural vocoders (Sotelo et al., 2017; Shen et al., 2018; Ping et al., 2018).

Several recent approaches have explored unsupervised generation of raw audio. Oord et al. (2016) propose WaveNet, a convolutional model which learns to predict raw audio samples by autoregressive modeling. Engel et al. (2017) pose WaveNet as an autoencoder to generate examples of recordings from musical instruments. Chung et al. (2014); Mehri et al. (2017) both train recurrent autoregressive models which learn to predict raw audio samples. In contrast to our method, generation with autoregressive models can produce audio examples of unbounded length. However, generation with autoregressive models is slow as the network must be evaluated once per sample, while WaveGAN can generate all audio samples in parallel. Oord et al. (2017) eliminate the issue of slow generation by employing knowledge distillation (Hinton et al., 2014). A teacher model, trained in the autoregressive setting, is used to teach a student model, capable of generating audio in parallel by filtering noise. This approach differs from the GAN setting in that the two networks cooperate.

The application of GANs (Goodfellow et al., 2014) to audio has so far been limited to supervised learning problems in combination with traditional loss functions. Pascual et al. (2017) apply GANs to raw audio speech enhancement; the generator ingests noisy speech and is trained to output clean speech. Their approach uses an encoder-decoder generator (analogous to (Isola et al., 2017)), and combines the GAN objective with traditional L_2 reconstruction loss. Jansson et al. (2017); Michelsanti & Tan (2017); Donahue et al. (2018) all use GANs in combination with unstructured losses to map spectrograms in one domain to spectrograms in another. Chen et al. (2017) use GANs to map images into associated audio spectrograms. Mogren (2016); Yu et al. (2017) explore music generation with GANs, but operate in symbolic domains and do not learn to generate the corresponding audio.

9. Conclusion

We present WaveGAN, the first application of GANs to unsupervised audio generation. Our experiments suggest that WaveGAN captures semantically-meaningful modes (i.e., words) of the real data distribution for small-vocabulary speech. We also compare WaveGAN to a model that naïvely applies image-generating GANs to audio spectrograms; while the spectrogram model achieves a higher inception score, humans prefer the sound quality of WaveGAN. We hope that this work catalyzes future investigation of GANs for audio generation, and establish a new task with reproducible evaluation methodology to further this goal.

ACKNOWLEDGMENTS

The authors thank Peter Boesman, Sander Dieleman, and Colin Raffel for helpful conversations about this work. This work was supported by the UC San Diego Department of Computer Science. GPUs used for this work were provided by the HPC @ UC program and donations from NVIDIA.

References

Arik, Sercan, Diamos, Gregory, Gibiansky, Andrew, Miller, John, Peng, Kainan, Ping, Wei, Raiman, Jonathan, and Zhou, Yanqi. Deep Voice 2: Multi-speaker neural text-to-speech. In *NIPS*, 2017.

Arjovsky, Martin, Chintala, Soumith, and Bottou, Léon. Wasserstein GAN. In *ICML*, 2017.

Berthelot, David, Schumm, Tom, and Metz, Luke. BEGAN: Boundary equilibrium generative adversarial networks. *arXiv:1703.10717*, 2017.

Boesman, Peter. Bird recordings. https://www.xeno-canto.org/contributor/OOECIWCSWV, 2018. Accessed: 2018-01-08.

Buchner, Johannes. Synthetic speech commands dataset. https://www.kaggle.com/jbuchner/synthetic-speech-commands-dataset, 2017. Accessed: 2017-01-15.

Chen, Lele, Srivastava, Sudhanshu, Duan, Zhiyao, and Xu, Chenliang. Deep cross-modal audio-visual generation. In *Proceedings of the on Thematic Workshops of ACM Multimedia*, 2017.

Chung, Junyoung, Gulcehre, Caglar, Cho, KyungHyun, and Bengio, Yoshua. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS Deep Learning and Representation Learning Workshop*, 2014.

Donahue, Chris, Li, Bo, and Prabhavalkar, Rohit. Exploring speech enhancement with generative adversarial networks for robust speech recognition. In *ICASSP*, 2018.

Dudley, Homer. Remaking speech. *The Journal of the Acoustical Society of America*, 1939.

Engel, Jesse, Resnick, Cinjon, Roberts, Adam, Dieleman, Sander, Eck, Douglas, Simonyan, Karen, and Norouzi, Mohammad. Neural audio synthesis of musical notes with WaveNet autoencoders. In *ICML*, 2017.

Garofolo, John S, Lamel, Lori F, Fisher, William M, Fiscus, Jonathan G, Pallett, David S, Dahlgren, Nancy L, and Zue, Victor. Timit acoustic-phonetic continuous speech corpus. *Linguistic data consortium*, 1993.

Goodfellow, Ian, Pouget-Abadie, Jean, Mirza, Mehdi, Xu, Bing, Warde-Farley, David, Ozair, Sherjil, Courville, Aaron, and Bengio, Yoshua. Generative adversarial networks. In *NIPS*, 2014.

Griffin, Daniel and Lim, Jae. Signal estimation from modified short-time fourier transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 1984.

Gulrajani, Ishaan, Ahmed, Faruk, Arjovsky, Martin, Dumoulin, Vincent, and Courville, Aaron. Improved training of Wasserstein GANs. In *NIPS*, 2017.

Hershey, Shawn, Chaudhuri, Sourish, Ellis, Daniel PW, Gemmeke, Jort F, Jansen, Aren, Moore, R Channing, Plakal, Manoj, Platt, Devin, Saurous, Rif A, Seybold, Bryan, et al. CNN architectures for large-scale audio classification. In *ICASSP*, 2017.

Hinton, Geoffrey, Vinyals, Oriol, and Dean, Jeff. Distilling the knowledge in a neural network. In *NIPS Deep Learning Workshop*, 2014.

Hunt, Andrew J and Black, Alan W. Unit selection in a concatenative speech synthesis system using a large speech database. In *ICASSP*, 1996.

Ioffe, Sergey and Szegedy, Christian. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

Isola, Phillip, Zhu, Jun-Yan, Zhou, Tinghui, and Efros, Alexei A. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017.

Jansson, Andreas, Humphrey, Eric, Montecchio, Nicola, Bittner, Rachel, Kumar, Aparna, and Weyde, Tillman. Singing voice separation with deep u-net convolutional networks. In *ISMIR*, 2017.

Karras, Tero, Aila, Timo, Laine, Samuli, and Lehtinen, Jaakko. Progressive growing of GANs for improved quality, stability, and variation. In *ICLR*, 2018.

LeCun, Yann, Bottou, Léon, Bengio, Yoshua, and Haffner, Patrick. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

Lee, Jongpil, Park, Jiyoung, Kim, Keunhyoung Luke, and Nam, Juhan. Sample-level deep convolutional neural networks for music auto-tagging using raw waveforms. In *Sound and Music Computing Conference*, 2017.

Ling, Zhen-Hua, Kang, Shi-Yin, Zen, Heiga, Senior, Andrew, Schuster, Mike, Qian, Xiao-Jun, Meng, Helen M, and Deng, Li. Deep learning for acoustic modeling in parametric speech generation: A systematic review of existing techniques and future trends. *IEEE Signal Processing Magazine*, 2015.

Mao, Xudong, Li, Qing, Xie, Haoran, Lau, Raymond YK, Wang, Zhen, and Smolley, Stephen Paul. Least squares generative adversarial networks. In *ICCV*, 2017.

Mehri, Soroush, Kumar, Kundan, Gulrajani, Ishaan, Kumar, Rithesh, Jain, Shubham, Sotelo, Jose, Courville, Aaron, and Bengio, Yoshua. SampleRNN: An unconditional end-to-end neural audio generation model. In *ICLR*, 2017.

Michelsanti, Daniel and Tan, Zheng-Hua. Conditional generative adversarial networks for speech enhancement and noise-robust speaker verification. In *INTERSPEECH*, 2017.

Mogren, Olof. C-RNN-GAN: Continuous recurrent neural networks with adversarial training. In *NIPS Constructive Machine Learning Workshop*, 2016.

Morise, Masanori, Yokomori, Fumiya, and Ozawa, Kenji. WORLD: a vocoder-based high-quality speech synthesis system for real-time applications. *IEICE Transactions on Information and Systems*, 2016.

Moulines, Eric and Charpentier, Francis. Pitch-synchronous waveform processing techniques for text-to-speech synthesis using diphones. *Speech communication*, 1990.

Odena, Augustus, Dumoulin, Vincent, and Olah, Chris. Deconvolution and checkerboard artifacts. *Distill*, 2016.

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

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

Pascual, Santiago, Bonafonte, Antonio, and Serrà, Joan. SEGAN: Speech enhancement generative adversarial network. In *INTERSPEECH*, 2017.

Ping, Wei, Peng, Kainan, Gibiansky, Andrew, Arik, Sercan O, Kannan, Ajay, Narang, Sharan, Raiman, Jonathan, and Miller, John. Deep Voice 3: 2000-speaker neural text-to-speech. In *ICLR*, 2018.

Radford, Alec, Metz, Luke, and Chintala, Soumith. Unsupervised representation learning with deep convolutional generative adversarial networks. In *ICLR*, 2016.

Sainath, Tara N, Weiss, Ron J, Senior, Andrew, Wilson, Kevin W, and Vinyals, Oriol. Learning the speech front-end with raw waveform CLDNNs. In *INTERSPEECH*, 2015.

Salimans, Tim, Goodfellow, Ian, Zaremba, Wojciech, Cheung, Vicki, Radford, Alec, and Chen, Xi. Improved techniques for training GANs. In *NIPS*, 2016.

Shen, Jonathan, Pang, Ruoming, Weiss, Ron J, Schuster, Mike, Jaitly, Navdeep, Yang, Zongheng, Chen, Zhifeng, Zhang, Yu, Wang, Yuxuan, Skerry-Ryan, RJ, et al. Natural TTS synthesis by conditioning WaveNet on Mel spectrogram predictions. In *ICASSP*, 2018.

Sotelo, Jose, Mehri, Soroush, Kumar, Kundan, Santos, Joao Felipe, Kastner, Kyle, Courville, Aaron, and Bengio, Yoshua. Char2Wav: End-to-end speech synthesis. In *ICLR Workshops*, 2017.

Stevens, Stanley Smith, Volkmann, John, and Newman, Edwin B. A scale for the measurement of the psychological magnitude pitch. *The Journal of the Acoustical Society of America*, 1937.

Szegedy, Christian, Vanhoucke, Vincent, Ioffe, Sergey, Shlens, Jon, and Wojna, Zbigniew. Rethinking the inception architecture for computer vision. In *CVPR*, 2016.

Theis, Lucas, Oord, Aäron van den, and Bethge, Matthias. A note on the evaluation of generative models. In *ICLR*, 2016.

Tokuda, Keiichi, Nankaku, Yoshihiko, Toda, Tomoki, Zen, Heiga, Yamagishi, Junichi, and Oura, Keiichiro. Speech synthesis based on hidden markov models. *Proceedings of the IEEE*, 2013.

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

Warden, Pete. Speech commands dataset. https://research.googleblog.com/2017/08/launching-speech-commands-dataset.html, 2017. Accessed: 2017-12-15.

Yoshimura, Takayoshi. Simultaneous modeling of phonetic and prosodic parameters, and characteristic conversion for hmm-based text-to-speech systems. 2002.

Yu, Lantao, Zhang, Weinan, Wang, Jun, and Yu, Yong. SeqGAN: Sequence generative adversarial nets with policy gradient. In *AAAI*, 2017.

Zen, Heiga, Tokuda, Keiichi, and Black, Alan W. Statistical parametric speech synthesis. *Speech Communication*, 2009.

Table 3. WaveGAN generator architecture

Operation	Kernel Size	Output Shape
Input $\boldsymbol{z} \sim \text{Uniform}(-1, 1)$		$(n, 100)$
Dense 1	$(100, 256d)$	$(n, 256d)$
Reshape		$(n, 16, 16d)$
ReLU		$(n, 16, 16d)$
Trans Conv1D (Stride=4)	$(25, 16d, 8d)$	$(n, 64, 8d)$
ReLU		$(n, 64, 8d)$
Trans Conv1D (Stride=4)	$(25, 8d, 4d)$	$(n, 256, 4d)$
ReLU		$(n, 256, 4d)$
Trans Conv1D (Stride=4)	$(25, 4d, 2d)$	$(n, 1024, 2d)$
ReLU		$(n, 1024, 2d)$
Trans Conv1D (Stride=4)	$(25, 2d, d)$	$(n, 4096, d)$
ReLU		$(n, 4096, d)$
Trans Conv1D (Stride=4)	$(25, d, c)$	$(n, 16384, c)$
Tanh		$(n, 16384, c)$

Table 4. WaveGAN discriminator architecture

Operation	Kernel Size	Output Shape
Input $\boldsymbol{x}$ or $G(\boldsymbol{z})$		$(n, 16384, c)$
Conv1D (Stride=4)	$(25, c, d)$	$(n, 4096, d)$
LReLU ($\alpha = 0.2$)		$(n, 4096, d)$
Phase Shuffle ($n = 2$)		$(n, 4096, d)$
Conv1D (Stride=4)	$(25, d, 2d)$	$(n, 1024, 2d)$
LReLU ($\alpha = 0.2$)		$(n, 1024, 2d)$
Phase Shuffle ($n = 2$)		$(n, 1024, 2d)$
Conv1D (Stride=4)	$(25, 2d, 4d)$	$(n, 256, 4d)$
LReLU ($\alpha = 0.2$)		$(n, 256, 4d)$
Phase Shuffle ($n = 2$)		$(n, 256, 4d)$
Conv1D (Stride=4)	$(25, 4d, 8d)$	$(n, 64, 8d)$
LReLU ($\alpha = 0.2$)		$(n, 64, 8d)$
Phase Shuffle ($n = 2$)		$(n, 64, 8d)$
Conv1D (Stride=4)	$(25, 8d, 16d)$	$(n, 16, 16d)$
LReLU ($\alpha = 0.2$)		$(n, 16, 16d)$
Reshape		$(n, 256d)$
Dense	$(256d, 1)$	$(n, 1)$

A. Architecture description

In Tables 3 and 4, we list the full architectures for our WaveGAN generator and discriminator respectively. In Tables 5 and 6, we list the same for SpecGAN. In these tables, n is the batch size, d modifies model size, and c is the number of channels in the examples. All dense and convolutional layers include biases.

B. Training hyperparameters

In Table 7, we list the values of these and all other hyperparameters for our experiments, which constitute our out-of-the-box recommendations for WaveGAN and SpecGAN.

C. Feline Turing test

In Figure 8, we show results for a very formal study in which results from our experiments with bird vocalizations were played for a *felis catus* specimen.

Table 5. SpecGAN generator architecture

Operation	Kernel Size	Output Shape
Input $\boldsymbol{z} \sim \text{Uniform}(-1, 1)$		$(n, 100)$
Dense 1	$(100, 256d)$	$(n, 256d)$
Reshape		$(n, 4, 4, 16d)$
ReLU		$(n, 4, 4, 16d)$
Trans Conv1D (Stride=2)	$(5, 5, 16d, 8d)$	$(n, 8, 8, 8d)$
ReLU		$(n, 8, 8, 8d)$
Trans Conv1D (Stride=2)	$(5, 5, 8d, 4d)$	$(n, 16, 16, 4d)$
ReLU		$(n, 16, 16, 4d)$
Trans Conv1D (Stride=2)	$(5, 5, 4d, 2d)$	$(n, 32, 32, 2d)$
ReLU		$(n, 32, 32, 2d)$
Trans Conv1D (Stride=2)	$(5, 5, 2d, d)$	$(n, 64, 64, d)$
ReLU		$(n, 64, 64, d)$
Trans Conv1D (Stride=2)	$(5, 5, d, c)$	$(n, 128, 128, c)$
Tanh		$(n, 128, 128, c)$

Table 6. SpecGAN discriminator architecture

Operation	Kernel Size	Output Shape
Input $\boldsymbol{x}$ or $G(\boldsymbol{z})$		$(n, 128, 128, c)$
Conv1D (Stride=4)	$(5, 5, c, d)$	$(n, 64, 64, d)$
LReLU ($\alpha = 0.2$)		$(n, 64, 64, d)$
Conv1D (Stride=4)	$(5, 5, d, 2d)$	$(n, 32, 32, 2d)$
LReLU ($\alpha = 0.2$)		$(n, 32, 32, 2d)$
Conv1D (Stride=4)	$(5, 5, 2d, 4d)$	$(n, 16, 16, 4d)$
LReLU ($\alpha = 0.2$)		$(n, 16, 16, 4d)$
Conv1D (Stride=4)	$(5, 5, 4d, 8d)$	$(n, 8, 8, 8d)$
LReLU ($\alpha = 0.2$)		$(n, 8, 8, 8d)$
Conv1D (Stride=4)	$(5, 5, 8d, 16d)$	$(n, 4, 4, 16d)$
LReLU ($\alpha = 0.2$)		$(n, 4, 4, 16d)$
Reshape		$(n, 256d)$
Dense	$(256d, 1)$	$(n, 1)$

Table 7. WaveGAN hyperparameters

Name	Value
Input data type	16-bit PCM (requantized)
Model data type	32-bit floating point
Num channels (c)	1
Batch size (b)	64
Model size (d)	64
Phase shuffle (WaveGAN)	2
Phase shuffle (SpecGAN)	0
Loss	WGAN-GP (Gulrajani et al., 2017)
WGAN-GP λ	10
D updates per G update	5
Optimizer	Adam ($\alpha = 1\text{e}{-4}$, $\beta_1 = 0.5$, $\beta_2 = 0.9$)

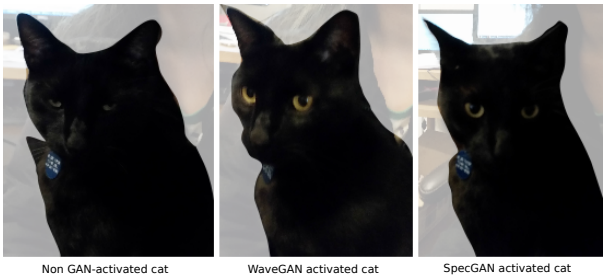

Figure 8. Compared to resting state, this cat's level of alertness increased when presented examples from WaveGAN. When presented with SpecGAN examples, it lost interest in the experiment.