Towards a Perceptual Loss: Using a Neural Network Codec Approximation as a Loss for Generative Audio Models

Ishwarya Ananthabhotla
ishwarya@media.mit.edu
MIT Media Lab
Cambridge, MA

Sebastian Ewert
sewert@spotify.com
Spotify, Inc.
London, United Kingdom

Joseph A. Paradiso
joep@media.mit.edu
MIT Media Lab
Cambridge, MA

ABSTRACT

Generative audio models based on neural networks have led to considerable improvements across fields including speech enhancement, source separation, and text-to-speech synthesis. These systems are typically trained in a supervised fashion using simple element-wise ℓ_1 or ℓ_2 losses. However, because they do not capture properties of the human auditory system, such losses encourage modelling perceptually meaningless aspects of the output, wasting capacity and limiting performance. Additionally, while adversarial models have been employed to encourage outputs that are statistically indistinguishable from ground truth and have resulted in improvements in this regard, such losses do not need to explicitly model perception as their task; furthermore, training adversarial networks remains an unstable and slow process. In this work, we investigate an idea fundamentally rooted in psychoacoustics. We train a neural network to emulate an MP3 codec as a differentiable function. Feeding the output of a generative model through this MP3 function, we remove signal components that are perceptually irrelevant before computing a loss. To further stabilize gradient propagation, we employ intermediate layer outputs to define our loss, as found useful in image domain methods. Our experiments using an autoencoding task show an improvement over standard losses in listening tests, indicating the potential of psychoacoustically motivated models for audio generation.

CCS CONCEPTS

• **Computing methodologies** → *Neural networks.*

KEYWORDS

perceptual loss function, perception, neural networks, audio, audio coding

ACM Reference Format:
Ishwarya Ananthabhotla, Sebastian Ewert, and Joseph A. Paradiso. 2019. Towards a Perceptual Loss: Using a Neural Network Codec Approximation as a Loss for Generative Audio Models. In *Proceedings of the 27th ACM International Conference on Multimedia (MM '19), October 21–25, 2019, Nice, France.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3343031.3351148

MM '19, October 21–25, 2019, Nice, France

ACM ISBN 978-1-4503-6889-6/19/10…$15.00
https://doi.org/10.1145/3343031.3351148

1 INTRODUCTION

In recent years, generative models based on neural networks have seen a steep improvement in performance. A major remaining challenge in designing generative models, however, is evaluation (e.g. asking, "Does this face look natural?" in image generation). This difficulty extends to the design of meaningful loss functions used to train a model, as the loss is used as a proxy measure for the final evaluation.

In this context, designing trainable objective functions that reflect auditory perception remains an interesting yet open problem. In state-of-the-art audio generation models, such as WaveNet [11], SampleRNN [10], or Tacotron [15], the most commonly employed loss functions are sample-level distance metrics, such as a mean-squared-error (ℓ_2) or mean-absolute-error (ℓ_1) metric. Adversarial loss models have been suggested as a potential improvement over such naive losses [3, 13]. Here, instead of prescribing the output in all detail, the generative model is encouraged to produce results that are statistically indistinguishable from real data. Despite considerable potential for this concept, there are several limitations and disadvantages; First, GANs remain notoriously difficult to train. Further, in contrast to the image domain, only a few examples of successful audio-oriented GANs have been reported in the literature, suggesting more intrinsic challenges. Most importantly, however, the discriminator networks used in GANs to distinguish real from generated examples are not required to take human perception into account. In other words, the objective of matching real and model distributions is too strict, as two sounds might be perceptually equivalent while the network might exploit minimal differences to distinguish real and generated examples. This way, network capacity is wasted, training could be unnecessarily difficult, and the generative performance might be limited overall.

From an auditory modelling perspective, several computational models have been proposed to approximate human perception of audio quality – examples include the Perceptual Evaluation of Audio Quality (PEAQ) [14], the Perceptual Evaluation of Speech Quality (PESQ) [12], and the Perceptual Evaluation of Audio methods for Source Separation (PEASS) [5]. Such models employ a series of signal processing steps to model various aspects of the human auditory system. While these models were certainly useful in various problem spaces, they are highly task-specific and might not generalize beyond the types of artefacts they were originally designed to capture. Furthermore, their design typically does not correspond to a differentiable function that can be employed *during* training.

As an alternative, we propose the use of existing, extremely well tuned psychoacoustic models as found in audio compression codecs such as MP3 or AAC. More precisely, such codecs were designed to identify and eliminate signal components that are perceptually

less relevant to save bandwidth during transmission, exploiting in particular masking in time and frequency [1]. In our approach, we thus apply a codec to both the generated audio as well as some target to remove irrelevant components before using a metric to compare the two. Operating in the time domain in this way, we can incorporate any available, potentially task specific codec without requiring any knowledge of the form or properties of the underlying auditory filtering or the hardcoded psychoacoustic model.

For use as part of a neural network, we would need a codec that can be expressed as a differentiable function – which is not available for commercial codecs. We thus propose training a separate network to approximate a low bit-rate codec. This way, we not only retain our ability to employ any available codec but also obtain a fully differentiable function approximation of the underlying codec.

In preliminary experiments, we found that while this concept can already be effective it can also be unstable during training. We thus additionally incorporate the idea of a *feature loss* that has been popular in image domain methods [2, 8]. More precisely, we do not only compare the generated and target audio based on the final output of our codec network, but additionally based on all intermediate layer activations. As shown previously, this procedure effectively stabilizes the gradient computation through fixed networks used as function approximations [4]. Further, we can compare the generated and target audio at several semantic levels, which additionally improved our results.

For our experiments, we employ a simple autoencoding task as a proxy for a wider range of generative audio tasks, while eliminating the task specific complexities to accelerate experimentation. The listening tests we conducted indicate a considerable improvement in audio quality using our *codec-loss network* over a baseline ℓ_1 loss. Overall, our contributions in this work are summarized as follows:

(1) We design a fully convolutional network to emulate the FFMPEG implementation of a 16kbps MP3 codec.
(2) We employ this network in combination with intermediate layer outputs to construct a more perceptually relevant loss function. The network is referred to as the *loss network*.
(3) We construct a simple autoencoder framework, simulating a coding and decoding task, to demonstrate the utility of our approach. This network is referred to as the *encoder network*.
(4) We present the results of a crowd-sourced A/B listening test, in which we compare the outputs of an encoder network trained with our codec-loss to one trained with a standard ℓ_1 metric. We demonstrate a quantifiable improvement in the perceived reconstruction quality of the segments resulting from our approach.

2 RELATED WORK

The idea of using a separately trained *loss network* is well established, and was first investigated in the image domain. Work done in [2, 8] show the use of activation outputs from a secondary neural network, originally trained to perform a task unrelated to the primary network, in computing a *feature loss* term to achieve image style transfer. The idea was subsequently explored further in the image domain [4], and was recently also developed in the audio domain. Pre-print work in [6] and [9] demonstrate the use of feature losses for denoising and bandwidth extension tasks in speech, sourced from a secondary network performing a classification and autoencoding task, respectively. Drawing inspiration from this work, we go one step further to construct a loss function from a loss network designed to approximate audio coding, which is a task directly related to perception as opposed to an unrelated task as in [6]. Doing this allows us to compute both a feature loss term and a *prediction loss* term, or a comparison of the outputs of the loss network.

Additionally, there have been a few attempts to directly incorporate perceptual evaluation toolkits and models into neural networks. Work in [17] demonstrates improvements in speech enhancement by approximating STOI within the loss function, while in [16], gradients are estimated for the STOI and PESQ metrics at each training iteration in the absence of a differentiable approximation. The latter approach is shown to slow down training significantly due to the gradient estimation of external PESQ and STOI implementations at each step, while the former does not directly incorporate perceptual information – it is an approximation of an approximation for speech perception. Both approaches are also not generalizable beyond speech data. In this work, we propose a method that incorporates a generic model of audio perception and enables on-line training as a differentiable representation.

3 DATASET

The data used for both the loss network training task and encoder network experiments (detailed below) is sourced from an internal dataset, consisting of 10,000 lossless musical tracks sampled at 44100Hz and spanning a variety of genres. The data is partitioned into train, validation, and test data subsets in a 70:20:10 ratio for each task, and preprocessed as necessary depending on the nature of the task. This is further detailed in the following sections.

4 LOSS NETWORK

As our custom loss function, we aim to train a network G such that for any lossless audio segment x_l, $G(x_l)$ produces x_c, a 16kbps MP3 encoded version of x_l.

We train two versions of such a loss network – one that operates on magnitude spectrogram input (discarding phase information), and one that operates directly on time domain input (retaining all phase information).

4.1 Magnitude STFT Domain

Our model architecture is a U-Network, closely following the work done in [7]. It consists of a series of 2D convolutional layers with a stride of 2 followed by a series of 2D transposed convolution layers, with skip connections between these downsampling and upsampling layers. An overview of our model architecture is provided in Figure 1, with our final hyperparameters listed in Table 1.

We first pre-process our dataset to downsample the tracks to 16,000Hz and sum them to have a single channel (in the interest of limiting the network size and computation time), and generate a secondary, compressed version of each track using FFMPEG's MP3 implementation. We then train our network to operate on pairs of lossless and coded magnitude spectrograms, using an ℓ_1 error metric. During training, we extract patches of time domain audio consisting of 66048 samples, and compute a magnitude STFT

Parameter	Magnitude Spectrogram	Time Domain
Number of Layers	5	8
W	128	16384
H	512	N/A
F	32	32
Batch Normalization	All layers	N/A
Dropout	50% (first 3 upsampling layers)	N/A
Kernel Size (Downsampling)	(5,5), Stride=2	15, Stride=1
Kernel Size (Upsampling)	(5,5), Stride=2	9, Stride=2
Activation	*ReLu*, *sigmoid* in final layer	*ReLu*, *tanh* in final layer
Learning Rate	0.001	0.0001
Decay	5e-6	5e-6
Batch Size	32	16
Optimizer	Adam	Adam

Table 1: A list of the model architecture parameters and hyperparameters used in training the magnitude spectral-domain and time domain loss networks.

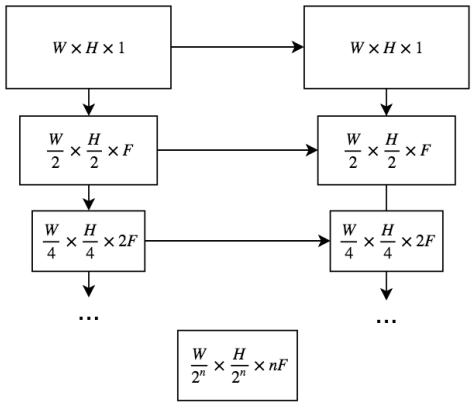

Figure 1: An illustration of the model architecture used for our loss networks.

representation with an FFT size of 1024 and hop length of 512, which are used as input.

4.2 Time Domain

In our experiments, we noted that modeling a codec in the spectral domain accelerated training, required less computational resources, and resulted in better test set accuracies than modeling a codec directly in the time domain – however, it is important to note that phase information is not considered in this process. As a more thorough approach, we additionally present the results from our attempt to train a loss function in the time domain, though we acknowledge a reduction in accuracy as a result of limited access to compute infrastructure.

We similarly preprocess the dataset by downsampling the audio to 16,000Hz and summing to a single channel. We extract time domain patches of audio consisting of 16384 samples, with 50% overlap, and feed these segments as input to our network. We again design a U-Network closely resembling our STFT loss network, and apply an ℓ_1 error metric as our loss. Each downsampling layer consists of a 1D convolution with a stride of 1, followed by a decimation of 2; each upsampling layer consists of a 1D transposed convolution. The model architecture and hyperparameters can be found in Figure 1 and Table 1 respectively.

4.3 Evaluation

Both networks are trained for approximately 36-48 hours on a single TI-1080 GPU. To indicate the effectiveness of either of these approaches, we show a series of examples in Figure 2, which visually compare the results of our neural network approximation to ground truth, i.e. the MP3 coded signal. In the spectral images in the third row, we see that the magnitude spectrogram codec approximation learns not only the low-pass filter at the appropriate cutoff frequency, but also the gaps in the spectrum that are the result of masking strategies employed in the coding process. With the time domain approximation (fourth row), however, we observe a noisier approximation of the masking process, but posit that better accuracy can be obtained with deeper networks and longer training times. As a result, we also do not expect the time domain loss function to perform comparably to the spectral loss function in our experimental setting discussed below, but refer to future experiments. Audio samples from both implementations (with the ground truth phase applied to the magnitude spectrum results) are available at ishwaryaanant.github.io/codec-perceptual-loss.

5 ENCODER NETWORK

The codec-loss functions described in Section 4 can be employed for a various tasks that entail generating audio data. As a proof-of-concept, we consider a simple coding and decoding task. Specifically, we consider a simple autoencoder setting, in which a given audio segment is compressed into a lower dimensional internal representation (coding stage), from which the original audio content is then to be reconstructed (decoding stage).

The model consists of a series of 1D convolutional and decimating layers for for the coding stage, followed by 1D transpose convolutions for the decoding stage. Model parameters are chosen such that the bottleneck layer contains 50% of the total number of variables of the input representation. We emphasize that the

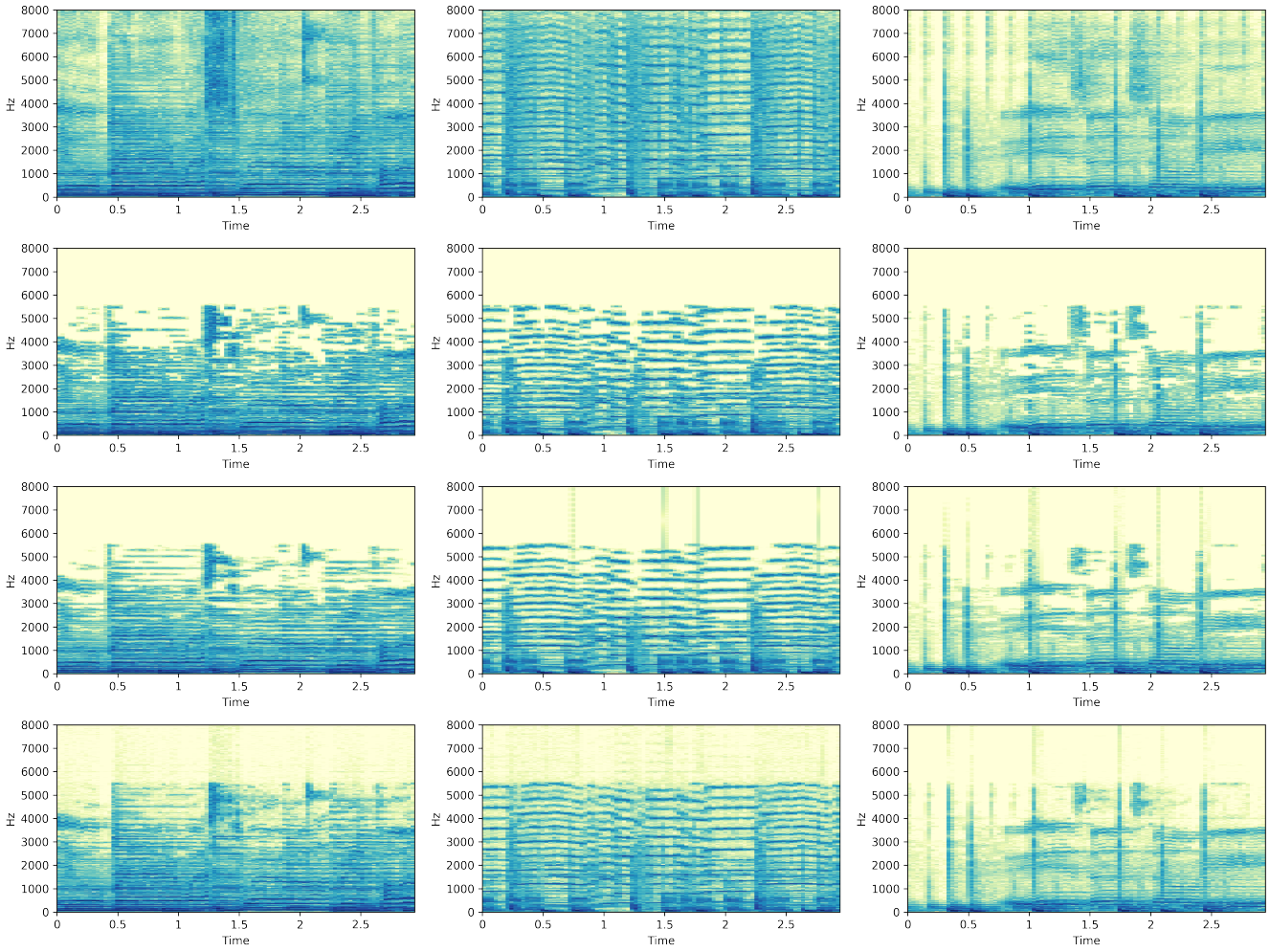

Figure 2: Example predictions from the two loss network configurations for three different tracks. Top to bottom: Lossless, FFMPEG codec (ideal), Magnitude STFT, Time Domain. We observe a detailed reconstruction of the coding behavior in the magnitude STFT result, while we observe a slightly noisier reconstruction in the time domain results.

objective of this encoder network is to create a lightweight testbed for evaluating our loss function within reasonable training times, and *not* to suggest a state-of-the-art, deep-learning driven codec.

6 COMBINING THE ENCODER AND LOSS NETWORKS: EXPERIMENTS

To study the benefits of our proposed loss function, we conduct a series of experiments combining the proposed encoder network with four different loss configurations. The models are defined below, and illustrated in Figure 3. We additionally give the exact model architecture parameters for each configuration in Table 2.

Model A: Baseline As a starting point, the encoder network is trained with the following loss:

$$L_A = ||x - x_p||_1 \tag{1}$$

Parameter	Encoder Network
Number of Layers	8
Input Size	66048 (A,B,C), 16384 (D)
Number of Filters	12
Kernel Size (Downsampling)	9, Stride=1
Kernel Size (Upsampling)	9, Stride=1
Activation	*ReLu*, *tanh* in final layer
Learning Rate	0.001
Decay	5e-6
Batch Size	32
Optimizer	Adam

Table 2: Model architecture parameters and hyperparameters employed in our encoder network.

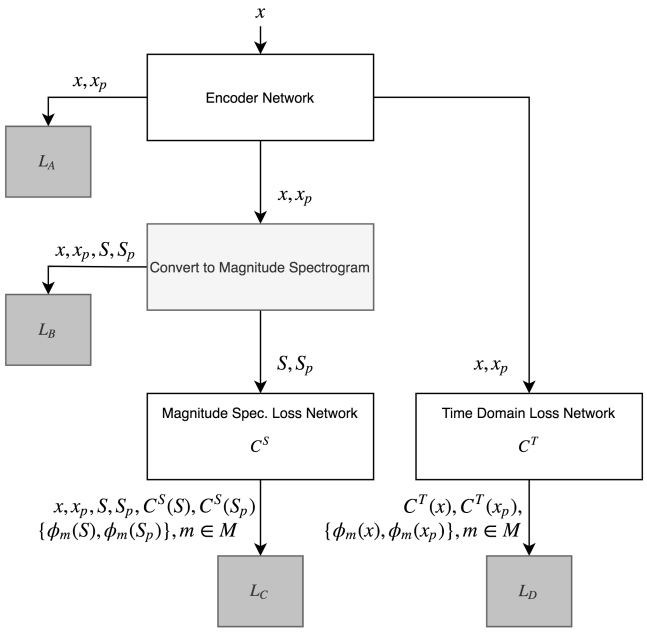

Figure 3: An overview of our experiments combining the encoder networks and the loss networks; L_A, L_B, L_C, L_C correspond to Models A, B, C, and D respectively.

where the time domain input to the encoder network is x, and the prediction of the encoder network given x is defined as x_p. This is a standard autoencoder using an ℓ_1 loss, which we use as a baseline.

Model B: Magnitude STFT Baseline To isolate the benefits of computing the loss in the *coded* magnitude spectrogram domain from simply computing it in the original magnitude spectrogram domain, we consider a model with the loss network defined as:

$$L_B = \lambda_a||x - x_p||_1 + \lambda_b||S(t,f) - S_p(t,f)||_1, t \in T, f \in F \quad (2)$$

where $S(t,f)$ is the magnitude spectrogram representation of a signal x, and λ_a and λ_b are weighting terms. We include a time domain error term here and in the subsequent model to encourage the network to also optimize for phase content, since this is discarded in the magnitude spectral loss functions (provided that enough free capacity is available in the encoder network).

Model C: Magnitude STFT Codec Loss Function We then design an experiment to evaluate our first custom loss function, trained to generate the magnitude spectrogram of an audio file encoded via MP3. For this model, our loss is defined as:

$$\begin{aligned} L_C =& \lambda_a||x - x_p||_1 \\ &+\lambda_b||C^S(S(t,f)) - C^S(S_p(t,f))||_1 \\ &+\lambda_c \sum_{m=1}^{m=M} \lambda_m||\phi_m(S(t,f)) - \phi_m(S_p(t,f)||_1, \\ & t \in T, f \in F \end{aligned} \quad (3)$$

where C^S is the magnitude spectrogram loss network, and ϕ_m is defined as the network activation of the m-*th* layer of the loss network. λ_m represents a weighting term for each of these layers, while λ_a, λ_b, and λ_c are the weighting terms for each component of the loss. The loss function consists of a *prediction loss* term, a *feature loss* term, and the time domain error term as above.

Model D: Time Domain Codec Loss Function Finally, we conduct an experiment to evaluate our second custom loss function, trained to approximate an MP3 codec directly in the time domain. Here, our loss is defined as:

$$\begin{aligned} L_D =& \lambda_a||C^T(x) - C^T(x_p)||_1 \\ &+\lambda_b \sum_{m=1}^{m=M} \lambda_m||\phi_m(x) - \phi_m(x_p)||_1 \end{aligned} \quad (4)$$

where C^T is the time domain loss network. We include a prediction loss and a feature loss term, but exclude the time domain error term, as this custom loss function inherently constrains phase information.

6.1 Training

We pre-process the dataset by downsampling all tracks to 16,000Hz, and summing to a single channel. Additionally, we intentionally band-limit the data to 3000Hz using a zero-phase sinc interpolation method, so that the spectrum width for all samples is within the range of the codec approximation we are using as our loss function, and to greatly simplify the task of the encoding network (see Section 8 for a discussion of this choice). We train Model A, B, C by extracting time domain audio patches consisting of 66048 samples and feeding these as inputs to our encoding network. In Model B and C, the ground truth sample x and the resulting encoder prediction x_p are then converted to a magnitude spectrogram representation S and S_p, using an FFT size of 1024 and hop size of 512. In Model B, these values are used directly to compute L_B, as above; in Model C, they are fed to the loss network C^S to calculate the L_C. In Model D, due to the smaller size of loss network C^T, we extract time domain audio patches consisting of 16384 samples and feed these to our encoder network. x and x_p from this network are then directly input to C^T to compute L_D.

Hyperparameters for each model are given in Table 2. Baseline models A and B are trained for 36-48 hours, while the custom loss models C and D are trained for 96 - 120 hours, all on a single TI-1080 GPU.

7 RESULTS

Model A	Model B	Model C	Model D
59.3	55.6	62.7	45.6

Table 3: Per-model SNR, dB

In Figure 4, we show examples of outputs from each of the four model configurations, compared with the original, lossless input for reference. Audio samples can also be found at http://ishwaryaanant.github.io/codec-perceptual-loss. In Table 3, we provide the *SNR* computed and averaged across non-overlapping frames of size 66048 from all tracks within the test set. We see that the *SNR* for Model C is slightly higher than the baseline models. At first, this

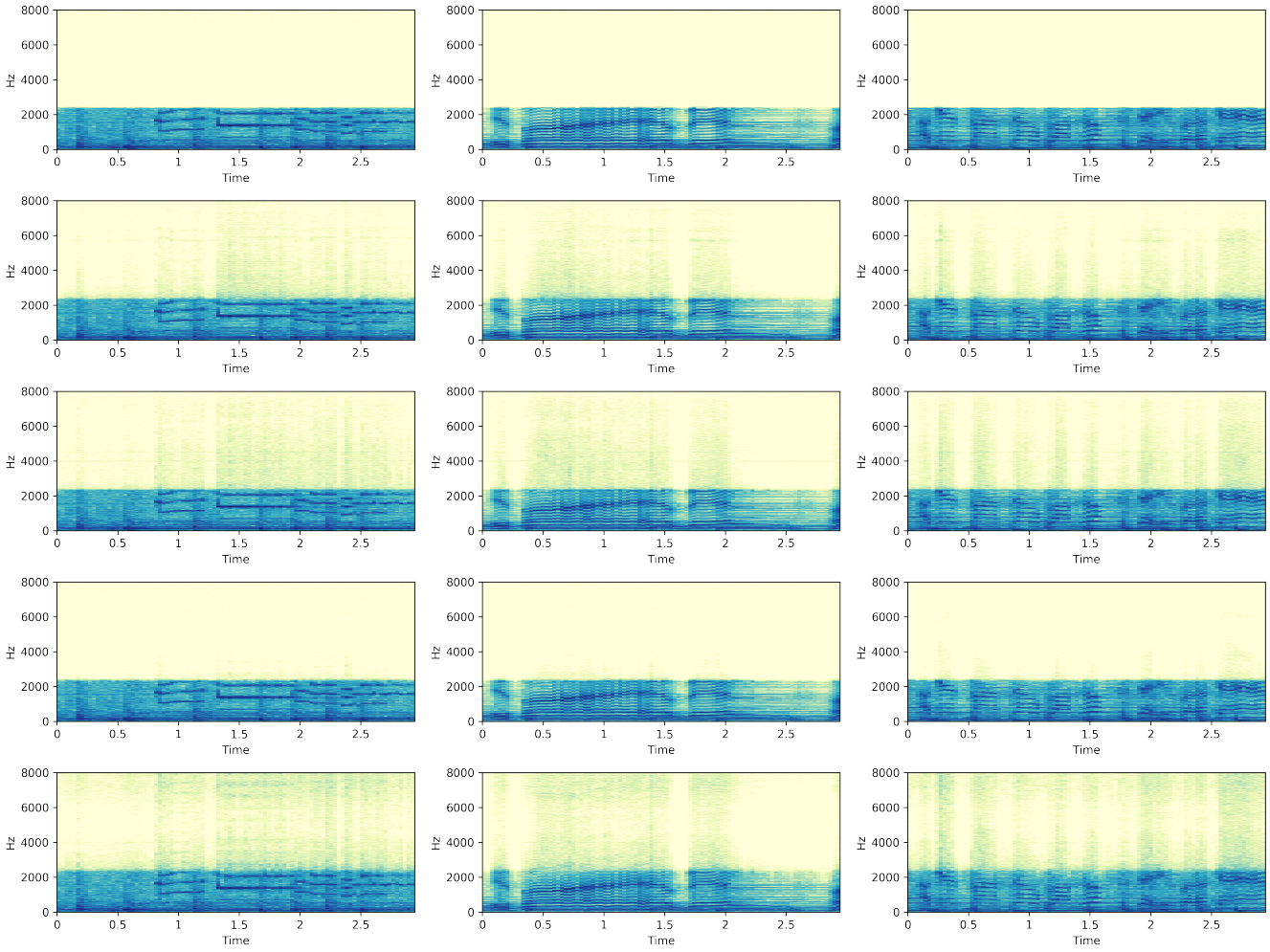

Figure 4: Example predictions from our combined encoder network and loss network experiments from three different tracks. Top to bottom: Lossless, Model A, Model B, Model C, Model D. We observe the output most perceptually comparable to the lossless examples resulting from Model C.

may appear counter-intuitive as we aim at improving the perceptual quality with model C over A and B, which might have come at the cost of a reduced energy-based quality, i.e. lower SNR values. Note however that we utilize an ℓ_1 loss for models A and B, which does not directly optimize for SNR (in contrast to an ℓ_2 loss). Further, we hypothesize that the introduction of a feature loss term might have enabled improved training behavior which is independent of any additional perceptual benefits, though this would need to be verified more thoroughly in future experiments.

We next conduct a crowd-sourced A/B test to rank the four models with regards to audio quality. We select five tracks across differing musical genres, and select an arbitrary 5-second segment from each. We then create a set of samples consisting of this original audio excerpt and the network predictions of Models A-D on this excerpt. For the test, we create pair-wise comparisons that present each of these excerpts against every other excerpt within a set, and shuffle all comparisons across tracks. For each of the 50 comparisons presented to them, participants are asked to select the sample with the better audio quality, or choose "I Don't Know" if they are unable to decide. We recruited 14 participants from our institutions, and a demonstration of this experiment can be found at http://audio-mafia.media.mit.edu/crowd-codec-ab.

In Table 4, we give the number of times an excerpt corresponding to a particular model was rated higher than the excerpt it was presented against, as a fraction of the total number of times it appeared across the experiment (20 times). We average this quantity across all participants and provide the mean and standard deviation. We additionally show a variant of a confusion matrix showing the number of times a pair of excerpts from specific model types were rated using the "I Don't Know" option, as a fraction of the number of times the pairing appears across the experiment (5 times). This quantity, also averaged across participants, is given in Table 5.

From the statistics, we infer that Model C (Magnitude STFT loss function) outperforms both baselines models (A and B), and is rated

comparably to the lossless reference example. Subjectively, we note that participants observed a reduction in noise and distortion in the Model C results as compared to the Model A and B results. We additionally note that the time domain loss function model (D) does not improve upon the baseline performance. We discuss this behavior further below.

	Original	Model A	Model B	Model C	Model D
Mean	**0.65**	0.2	0.46	**0.68**	0.13
Std	**0.20**	0.07	0.08	**0.11**	0.05

Table 4: The table indicates the number of times a sample corresponding to a model was rated higher than another sample it was presented against, as a fraction of the total number of times the sample is presented in the listening test. We show the mean and standard deviation when averaged across all participants.

	Original	Model A	Model B	Model C	Model D
Orig.	N/A	0.12, 0.24	0.04, 0.08	**0.92, 0.2**	0.08, 0.16
Model A	0.12, 0.24	N/A	0.16, 0.2	0.08, 0.16	0.2, 0.25
Model B	0.04, 0.08	0.16, 0.2	N/A	0.08, 0.1	0.04, 0.08
Model C	**0.92, 0.2**	0.08, 0.16	0.08, 0.1	N/A	0.0, 0.0
Model D	0.08, 0.16	0.2, 0.25	0.04, 0.08	0.0, 0.0	N/A

Table 5: The table indicates the number of a times an pair of samples was rated with the "I Don't Know" option, suggesting similar audio quality, as a fraction of the total number of presentations of the pair in the listening test. We give the mean and standard deviation with averaged across all participants.

8 DISCUSSION AND FUTURE WORK

8.1 Computing Loss without Phase Information

In this work, we present a version of the loss network that is trained in the magnitude spectrogram domain, and motivated by a better training behavior and the need for fewer computational resources than its time domain counterpart. However, when using a time domain encoder, the phase information from the audio samples fed through the network is not inherently optimized in the loss network. While we include a time domain comparison term in our loss function when using the magnitude spectrum loss network as a means of mitigating this to an extent, we observe an interesting behaviour when we train our encoder network with broadband audio samples. In Figure 5 (a), we show a track limited to a sample rate of 10000Hz instead of the 6000Hz in our experiments, used as the ground truth input to the encoder. Figure 5 (b) demonstrates the baseline ℓ_1 reconstruction (Model A), where only a fraction of the spectrum width is recovered by the network. In both (c) and (d), corresponding to the magnitude spectrogram baseline (Model B) and magnitude spectrogram loss function (Model C) respectively, we can observe that the network attempts to interpolate the spectral content to its full width using the magnitude information; however, without any additional phase information, we observe that the network implements this behaviour by generating a series of broadband pulses that manifest in the reconstructed signal as noisy "clicks". The reconstruction resulting from the time domain loss function (Model D), however, shown in (e), does not exhibit these artifacts and improves upon the Model A reconstruction somewhat (which is a promising result suggesting the value of the time domain loss network). Overall, these observations point us to several interesting avenues for future work. First, we might improve results further by increasing the capacity of the encoder network and time domain loss network with deeper networks, allowing us to operate on broadband signals or to eliminate the conversion to the magnitude spectrogram domain entirely. Further, we might consider a better spectral phase representation that would allow us to train with the magnitude and phase jointly.

8.2 The Challenges of Time Domain Modeling

As previously mentioned, we were limited in our exploration of time domain modeling in both the loss function training phase and task application phase by computational overhead, and acknowledge the scope for improvement in model performance in this direction. We additionally notice aliasing behavior in the output spectrum (visible in Figure 4) when training with the time domain loss function regardless of the downsampling or upsampling strategy employed by the autoencoder network, similar to the reporting in [9]. We were unable to mitigate this by fixed anti-aliasing filters within the autoencoder network, and consider exploring this anomaly in greater detail to develop an improved time domain loss function.

9 CONCLUSION

In this work, we demonstrate the feasibility of a fully differentiable, perceptually motivated loss function, designed as a neural network approximation of a low bitrate MP3 codec. We demonstrate that using a weighted combination of this network's layer activation outputs and predictions as an objective function improves the performance of a secondary autoencoding task as compared to an ℓ_1 baseline, and discuss the tradeoff between modeling such a system in the time and frequency domains. Through this work, we suggest that the more general paradigm of capitalizing on psychoacoustic information built into audio coding or other processes may be a valuable tool for introducing online perceptual evaluation methods to deep neural networks.

REFERENCES

[1] Marina Bosi and Richard E Goldberg. 2012. *Introduction to digital audio coding and standards*. Vol. 721. Springer Science & Business Media.

[2] Qifeng Chen and Vladlen Koltun. 2017. Photographic image synthesis with cascaded refinement networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 1511–1520.

[3] Chris Donahue, Julian McAuley, and Miller Puckette. 2019. Adversarial Audio Synthesis. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

[4] Alexey Dosovitskiy and Thomas Brox. 2016. Generating images with perceptual similarity metrics based on deep networks. In *Advances in Neural Information Processing Systems (NIPS)*. 658–666.

[5] Valentin Emiya, Emmanuel Vincent, Niklas Harlander, and Volker Hohmann. 2010. The PEASS Toolkit-Perceptual Evaluation methods for Audio Source Separation.

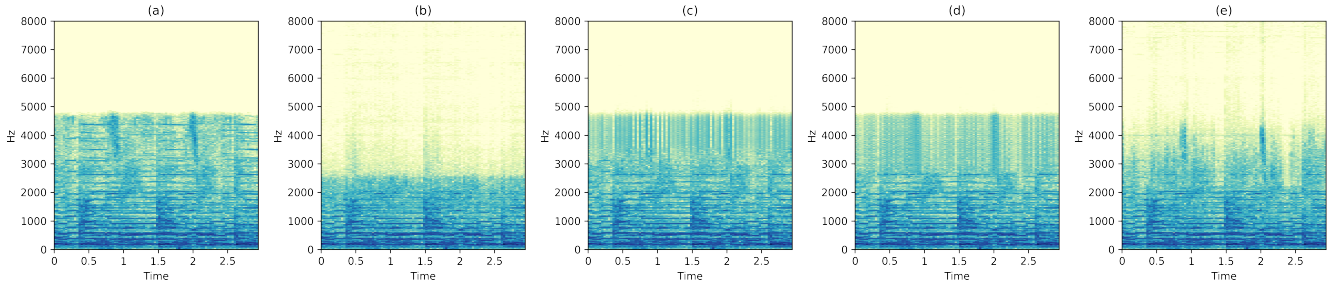

Figure 5: Examples of conducting the experiments with broadband inputs, forcing the network to interpolate phase as a result of transferring from the time domain to the spectral domain. Left to right: (a) broadband, lossless input; (b) Model A, (c) Model B, (d) Model C, (e) Model D.

In *Proceedings of the International Conference on Latent Variable Analysis and Signal Separation (LVA/ICA)*.

[6] Francois G Germain, Qifeng Chen, and Vladlen Koltun. 2018. Speech denoising with deep feature losses. *arXiv preprint arXiv:1806.10522* (2018).

[7] Andreas Jansson, Eric Humphrey, Nicola Montecchio, Rachel Bittner, Aparna Kumar, and Tillman Weyde. 2017. Singing voice separation with deep U-Net convolutional networks. In *Proceedings of the International Society for Music Information Retrieval (ISMIR)*.

[8] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. 2016. Perceptual losses for real-time style transfer and super-resolution. In *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 694–711.

[9] Sung Kim and Visvesh Sathe. 2019. Adversarial Audio Super-Resolution with Unsupervised Feature Losses. https://openreview.net/forum?id=H1eH4n09KX

[10] Soroush Mehri, Kundan Kumar, Ishaan Gulrajani, Rithesh Kumar, Shubham Jain, Jose Sotelo, Aaron Courville, and Yoshua Bengio. 2017. SampleRNN: An unconditional end-to-end neural audio generation model. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

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

[12] Antony W Rix, John G Beerends, Michael P Hollier, and Andries P Hekstra. 2001. Perceptual evaluation of speech quality (PESQ)-a new method for speech quality assessment of telephone networks and codecs. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. 749–752.

[13] Daniel Stoller, Sebastian Ewert, and Simon Dixon. 2018. Adversarial Semi-Supervised Audio Source Separation applied to Singing Voice Extraction. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. Calgary, Canada, 2391–2395.

[14] Thilo Thiede, William C Treurniet, Roland Bitto, Christian Schmidmer, Thomas Sporer, John G Beerends, and Catherine Colomes. 2000. PEAQ - The ITU standard for objective measurement of perceived audio quality. *Journal of the Audio Engineering Society* 48, 1/2 (2000), 3–29.

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

[16] Hui Zhang, Xueliang Zhang, and Guanglai Gao. 2018. Training supervised speech separation system to improve STOI and PESQ directly. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 5374–5378.

[17] Yan Zhao, Buye Xu, Ritwik Giri, and Tao Zhang. 2018. Perceptually guided speech enhancement using deep neural networks. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 5074–5078.