2023 IEEE INTERNATIONAL WORKSHOP ON MACHINE LEARNING FOR SIGNAL PROCESSING, SEPT. 17–20, 2023, ROME, ITALY

LOW-COMPLEXITY STREAMING SPEECH SUPER-RESOLUTION

Erfan Soltanmohammadi, Paris Smaragdis, and Michael M. Goodwin

Amazon Web Services, Inc.

ABSTRACT

Speech super-resolution is the process of estimating the missing frequency content of a speech signal from its existing band-limited frequency content. The loss of frequency components is a common occurrence that can be because of a low sampling rate, low-quality microphones, or various transmission factors, and it is an increasingly common problem as bandwidth for high-quality communications is generally available, but many end devices are still using older standards and protocols. Although a number of solutions exist for this problem, we note that most are not amenable to real-world use, due to computational or algorithmic constraints. In this paper we present a compact, efficient, and minimal-latency solution to speech super-resolution that is suitable for use with real-time streaming data. We propose a novel causal architecture that can be easily deployed for real-world use. We additionally propose a novel adversarial training process and an initialization procedure that speeds up convergence and results in improved outputs. Objective and subjective results show that our proposed model outperforms the latest solutions in this space, despite being significantly smaller and faster.

Index Terms— speech super-resolution, bandwidth extension, speech synthesis

1. INTRODUCTION

The frequency content of telephony speech, as standardized, is limited between 300Hz and 3.4kHz. This small bandwidth of 3.1kHz is enough for humans to understand the speech. However, the speech signal with this bandwidth lacks fidelity and can cause listener fatigue [1].

Modern voice communications are now increasingly relying on internet networking, which can provide a higher bandwidth and afford a much better audio quality. However, since many users are connecting through telephony devices or call in through telephone networks, we often encounter callers with low-quality signals. Since increased bandwidth is not a limiting factor anymore, this situation can be improved with the use of speech super-resolution algorithms that can improve the audio quality of telephony callers when they connect to voice-over-IP calls. The necessary processing would estimate the low and high frequency contents that are missing in the telephony speech signal as illustrated in Fig. 1.

Although this is an old problem and there have been many bandwidth expansion algorithms presented in the context of signal processing, in recent years, there have been multiple proposed algorithms that produced superior results by making use of neural networks. Inspired by deep learning-based image super-resolution algorithms [2], Kuleshov et al. [3] propose a feedforward convolutional architecture called AudioUnet for bandwidth extension. They this architecture as an end-to-end task, and view the task as finding a map from one time series, which is the low bandwidth audio signal, into another time series, which is the high bandwidth audio signal.

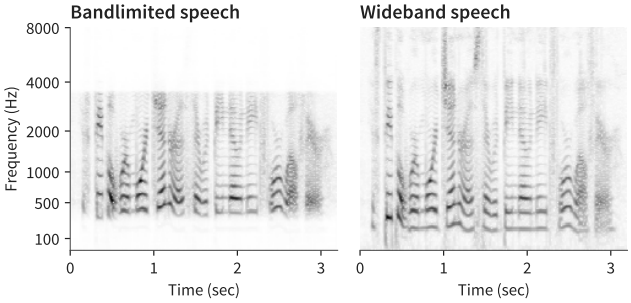

Fig. 1. Examples of band-limited and full-band speech. On the left we see a telephone voice recording, which is clearly missing content in the low and high frequencies. On the right, the same utterance in full bandwidth exhibits more signal content throughout the available frequency range. Our goal is to automatically recover the missing frequencies on the left. Note that the y axis is log-warped for legibility in the low frequencies, which under-represents the amount of missing frequency content which is more than half of the available bandwidth.

Motivated by the success of adversarial training in image super-resolution, Eskimez et al. propose and algorithm called MuGAN which uses an adversarial loss along with a log-power spectrograms loss in their deep neural network-based solution for extending the bandwidth of speech in narrowband 4kHz bandwidth signals to wideband 8kHz signals [4, 5].

Another generative adversarial network (GAN) based solution is proposed in [6]. In this method, the magnitudes of the high frequency components are estimated using a GAN architecture and the corresponding phase information is estimated using a MelGAN based vocoder [7].

Zhang et al. are first to use normalized flow structure [8] for the audio super-resolution algorithm called WSRGlow [9]. The network of WSRGlow operates on the short time Fourier transform of the input signal which allows the model to exploit both time domain and frequency domain information. These two factors accelerate the training and improve the performance of the model.

In [10], the authors propose using a diffusion probabilistic model [11] based on neural vocoders for bandwidth extension. The model is called NU-Wave and can generate natural sibilants and fricatives but does not perform well on generating harmonics of vowels. To solve this main failure of NU-Wave in generating harmonics, the authors in [12] propose NU-Wave 2 which uses short-time Fourier convolution with bandwidth spectral features to generate harmonics.

Similar to GAN-based vocoders [13], the authors in [14] propose NVSR which is a two-stage speech super-resolution architecture trained with an adversarial loss. In the approach high-resolution speech signals are constructed from the predicted high-resolution

979-8-3503-2411-2/23/$31.00

Mel-spectrograms. They use multi-resolution losses in the time and frequency domains which allows high quality reconstruction.

Despite the high-quality results of the aforementioned algorithms, their computational complexity is high enough that it makes them prohibitive for real-world deployment. Additionally, most often such models operate in a non-causal manner which limits their use for streaming data inference. As a result, in the context of real-life communications, these approaches are not providing a practical solution.

In this paper, we propose a super-resolution algorithm which provides improved output quality with a much smaller computational footprint and using causal-only operations, so it can be applicable to real-time applications. Along with extending the bandwidth, our proposed algorithm additionally compensates for the degradation that the low-quality microphones of many phones impose to the speech signal by reconstructing the low frequency missing components and equalizing the signal back to the original signal spectrum. Moreover, our algorithm also corrects quantization errors that are present in modern telephony pipelines. For example, the narrowband audio codec G.711 quantizes the audio signal with 8-bit resolution [15], which introduces some quantization noise. Our solution improves the bit resolution by reconstructing a floating-point signal that can be subsequently sampled at any desired bit depth.

2. PROPOSED MODEL

In this section we describe the model we developed. The goals of this design were to enable a compact, fast, causal, and low-latency model that can be deployed on streaming audio data. Some of the design decisions needed were obvious, e.g. we could only use causal convolutions, but others were more involved. In the following sections we outline the model structure and highlight the design decisions and trade-offs that we made. The proposed model consists of a cascade of units as shown in Fig. 2. Processing is sandwiched between an encoder-decoder pair, and most of the hard work is done by a repeating set of novel modules (MLPStreams) that operate in a higher-dimensional latent space.

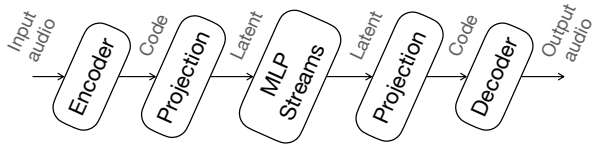

Fig. 2. The overall architecture of the proposed model

2.1. Encoder

This unit is responsible for transforming the very-correlated time domain samples into a higher dimensional representation with reduced temporal dependencies. There are a number of transforms, learned or fixed, that we use at this step. Perhaps the most obvious choice is the Short Time Fourier transform (STFT). It has the advantage of being well-understood and, by adjusting it's size and hop, can be appropriately adjusted to suit the input signal's characteristics in terms of harmonic and temporal content, while simultaneously balancing latency constraints. Although this can be an obvious choice for the encoder-decoder design, we are not making use of it as is.

The STFT does create a projection to a complex-valued space, which has significant downside; processing complex-valued data is not as straightforward and the advantages of learning in that space, as opposed to a real-only space, is still a hotly debated subject. In our experiments, we did not observe any advantage to using complex numbers, both from an optimization perspective (we could not achieve better results), but also from a computational perspective (complex number operations are currently not as well optimized as real-valued processing in most frameworks). Real-valued alternatives to the STFT would be the Modified Discrete Cosine Transform (MDCT), which exhibits similar properties as the STFT, but remains in the real number domain, or the Short Time Hartley Transform (STHT), which is a modified version of the STFT that uses the Hartley transform (essentially a real-valued version of the DFT).

Of course, at a high level most of these transforms are related to each other via simple transformations and would be expected to produce similar results. In our experiments we found that, by a small margin, the best performing solution was to concatenate the real and imaginary parts of a unitary DFT transform. Since this was the most performant solution in terms of implementation we used that for our experiments below. We can still achieve somewhat better performance, if we were to use a learned filterbank instead, however that comes at the expense of increased computation and a more difficult training process.

2.2. Projection to latent space

The aim of this unit is to transform the encoder's output to an abstract multi-dimensional space where the necessary processing will take space. To achieve this goal, we use a simple projection to the desired dimensionality, through a linear projection followed by a Parametric Rectified Linear Unit (PReLU) non-linearity [16]. This projection is not necessary if one wishes to perform processing directly on the encoder domain. In the case of using fixed transforms such as the DFT however, it is often advantageous to project to a different space that will not be constrained by the transform size (which is almost always tied to be the same as the input audio frame size). If one were to use a learned filterbank in the encoder, then this step would not be necessary since this projection could be incorporated in the learned transform.

2.3. MLPStreams

The main part of the processing is done by a sequence of repeating modules that operate in the latent space. We call these sub-units Multi-Layer Perceptron Streams (MLPStreams) and their structure is outlined in Fig. 3. The main goal in developing this type of unit in this work was to avoid the use of computationally demanding elements, such as full convolutions or transformers, while still maintaining a wide receptive field. The design of this unit is based on the idea of the ResMLP layer as presented in [17]. There are two main blocks, one processing the two-dimensional input in the left-right dimension (time), and the other in the up-down dimension (channels). In both steps, this processing is done independently in each column/row, but the interleaving of these two steps allows us to ultimately jointly transform both dimensions. Another important element in this architecture is the affine transform unit. These units implement a simple operation on their input, independently scaling and translating each dimension with learned parameters.

Now let us look inside the two parts of the MLPStream unit in more detail. First comes the temporal processing unit, that simply takes linear combinations of consecutive time samples of the inputs independently in each channel (i.e. operates in the left-right dimension only). This is done by employing a depth-wise 1D convolution layer, which for our purposes is using causal convolutions so that it

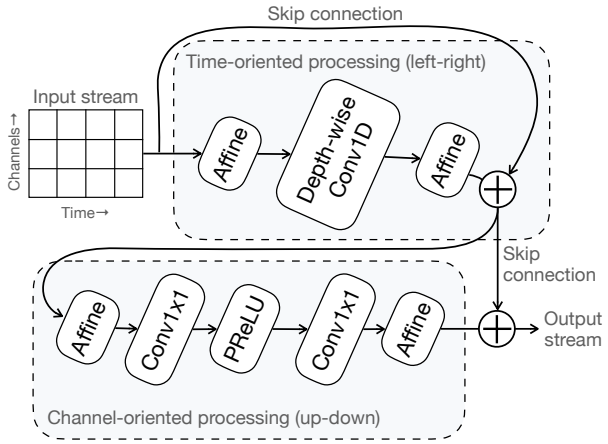

Fig. 3. The internal sub-units of the MLPStreams. Note that processing happens independently across channels and across time steps. By interleaving these two orientations we get to take all data into account with a significantly reduced amount of processing as compared to alternatives.

can be applied in real-time without latency. As compared to a more conventional 1D convolution layer, a depth-wise convolution does not apply multiple filters on all channels, instead convolving each channel independently with its own filter. By doing so it reduces the amount of computations and memory requirements considerably. We do not use a bias term for this convolution operation. Preceding and following this convolution are two affine transform layers. Although on paper these might seem redundant since their effect can be subsumed in the convolution operation, they serve an important role by independently conditioning the data for the layers that follow them. In a more traditional network architecture, these would have been normalization layers. However, due to the causality and computational constraints that we strive to satisfy, such an operation would not be usable as we would need to look into future time samples to properly normalize, or design an online normalization scheme, which itself would be significant added complexity. As in [17], the simplified processing and the strategic placing of affine transforms lets us bypass the need for any normalization layers, making our approach a viable option for deployment in real-time situations. Additionally, at inference time, the affine layers can be collapsed into the depth-wise convolution, which reduces the necessary computations by a significant amount. Finally, there is a skip connection that bypasses these layers and is averaged with their output. This additional pathway helps us better propagate gradients all the way to the start of the network.

The second part of the MLPStream is a collection of units that operates in the up-down dimension (i.e. over the channels). This has a similar structure as the first part, with affine transforms as the start and end, and a skip connection. Instead of the depth-wise convolution, which combined time samples, we now use the so-called 1x1 convolutions, which are essentially linear transforms across the input channels (but not across time). By doing so we give our network the ability to combine multiple latent dimensions, something that was not feasible in the preceding part. Just as with the depth-wise convolution, by not considering one of the two dimensions the amount of computations is significantly reduced. We use two of these 1x1 convolutions with no bias terms, and between them we include a PReLU activation function. Just as before, the skip connection averages the input and output of this segment. Just as before, at inference time, the affine transforms can be subsumed by the convolution layers, further reducing necessary computations.

In the full network, we use multiple of these MLStream units in succession. This allows us to create a larger temporal receptive field by compounding their convolutions, and it gives the network a chance to constantly switch the orientation on which it processes the data, thereby combining temporal and channel processing.

2.4. Projection to encoder space and decoding

Once processing is completed by the MLPStreams we go through a process where we undo the processing steps that brought us to the latent space. We use a postprocessing projection that helps us change the dimensionality of our latent space to that of the encoder space, thereby making our data suitable to put into the decoder. In the case where we employ a learned filterbank in the encoder and a projection to the latent space was not used, we similarly do not need to use this layer since the MLPStream outputs will already be in a suitable format to present to the decoder.

Finally, the decoder does the inverse operation of the encoder. For example, if the encoder is the STFT, then the decoder would be the inverse STFT. If the encoder is a convolutional layer, then we use another convolutional layer as the decoder. Once through the decoder the data will be in the waveform domain.

2.5. Model initialization

As with all neural net models, proper initialization can help speed up and improve training. In this particular case, there are some tricks we can use to ensure that we do not waste training iterations unnecessarily. As we will describe in more detail later, during training the network will be given bandlimited audio inputs and will be asked to produce the corresponding fullband audio output. Starting training with randomly initialized layers will require a considerable amount of training iterations until the network will be able to produce plausible sound outputs. Instead we make the observation that had the network simply produced the input as the output (i.e. it started as an identity function), it would already be at a local optimum. It wouldn't necessarily complete the desired enhancement task, but it would be properly calibrated to produce a properly scaled output that has at least some of the desired information. In effect, identify mapping will be a reasonable local optimum, certainly much better than a random network. Armed with this observation we note that all of the network components above have been designed so that they become identity functions.

For example, the encoder/decoder can be chosen to be the inverse of each other (e.g. a STFT/inverse STFT pair of transforms). The projection to the latent space, which is often of more dimensions that the encoder space, can be a padded identity matrix (and likewise, the projection back to the encoder space can simply invert that). Inside the MLPStream units, we initialize the affinity transforms to scale each dimension by 1 and translate it by 0, set all the convolutions to be an identity function, and set the parameter of the PReLU to be 1, which will effectively make it an identity function as well. Note that for our skip connections, we average the two converging paths, so that there is no extra scaling taking place. Since all of these operations are now identity functions, we know that we can initialize the network starting from a good initial condition. Doing so results in significantly accelerated training as compared to random initial conditions. Compared to Kaiming uniform initialization, which is the next best option that we found, identity initialization decreases training loss

by 3% on average after convergence is ultimately achieved. Identity initialization additionally speeds up convergence, reaching Kaiming's overall minimum loss in 60% fewer epochs..

2.6. Deploying as a streaming model

As defined, our model is causal and has minimal latency. Training is done offline to take advantage of batching and pipelining speedups, but we deploy it as a streaming model. To do so we use buffers to ensure convolution continuity across input frames. This does not add a notable amount of additional computation, but it will add some latency since in order to obtain a single filterbank frame we need to wait to have enough samples. Consequently in a typical deployment our algorithm has a latency of 120 samples (the size of the filterbank frame minus the hop size), which is 7.5ms.

2.7. Losses

During the training, the total loss that we use is a weighted sum of three different losses. In this section we explain each of them.

2.7.1. Time domain loss

The first loss we use is computed in the time domain and measures how similar each input is to the desired output. We measure that loss over a range of window frame sizes similar to [18]. For each frame size we calculate the average value in that frame and obtain the ℓ_1-norm between the input and output waveforms. We use the average loss from multiple frame sizes, and we always also use a frame size of 1, which of course becomes just the ℓ_1-norm between the input and output waveforms. We further calculate ℓ_1-norm of difference between the first order derivative of energy of the two signals in each segment. This would gives us a loss on the instantaneous phase. The average of these losses would be the loss measured on the time domain.

2.7.2. Frequency domain loss

To calculate the frequency domain loss, we find the difference between the decibel magnitude of the STFT of the target signal and the model's output in ℓ_1-norm. We additionally add pre-emphasis on the STFTs so that we can highlight the differences in the higher frequencies better. We calculate this loss for the STFTs using multiple window sizes. The average of these losses would be used as the first component of the frequency domain losses. Next, we calculate ℓ_1-norm of the difference between the Mel-spectrograms of the target signal and the model's output. Then, the frequency domain loss would be the average of these values.

2.7.3. Adversarial loss

After the model is fully trained based on the previous two losses, we add an additional adversarial loss based on a multi-period discriminator [13]. Using this extra term helps us to fine-tune the network and noticeably improves the quality of the output once the other two losses have converged.

3. RESULTS

In this section, throughout a set of experiments we evaluate the performance of the proposed model and compare it with the state-of-art models.

3.1. Settings

To train our model, we set the model parameters as follows. For the encoder, we use the STFT with the square root Hanning window so that we can ensure perfect reconstruction. We use the window size of 160 samples and hop size of 40 samples which, when used in streaming mode, imposes a delay of 7.5ms. We concatenate the real and imaginary parts of the STFT, except for the imaginary values of the DC and Nyquist which are zero. This results in a 160-dimensional representation in the encoder space. We set the size of the latent space to 512, and we use 12 MLPStream units in series. The length of the depth-wise filter in each MLStream unit is set to 5 taps. For the time domain loss, we use the set of $\{1, 240, 480, 960\}$ samples as the time frame lengths. We use an overlap of 50% of frame length for the segments greater than 1. For the frequency loss, we use the set of $\{2048, 1024, 512, 256, 128, 64\}$ as the window size of the employed STFTs and set the overlap size to 75% of the window size. We set the weight of the frequency loss to be twice that of the time domain loss, since it captures the high-frequency differences better than the time domain loss.

We use 65,536-sample long audio segments for training. This would be equivalent to 4.096 seconds at 16kHz sampling rate. We use a batch size of 16. We noticed that the larger batch sizes would result in slower training and a degraded output. For training the network we use the Adam optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. We set the initial learning rate to 0.005 which we then drop by a factor of 10 every 500 epochs. We add the adversarial loss after 1,500 epochs.

For training, we use more than 40 hours of clean recordings of speech signals from the VCTK corpus [19]. We resample the recordings to 16kHz, and use them directly as the wideband target signals during the training. To create the input narrowband signals, we downsample the original signals to 8kHz signals.

To reduce overfitting and improve generalization, we apply various data augmentation operations to the training dataset. The set of operations consists of applying random gain, imposing saturation, applying a noise gate, adding pink noise, and filtering the input signal using high-pass filters with random cutoff frequencies to emulate poor frequency response microphones. We additionally quantize the signals with varying bit resolutions to simulate quantization noise and cover artifacts from common speech codecs and encoding formats.

In the next sections, we summarize the performance evaluation and the comparison of our model to the latest methods proposed for speech super-resolution. For all other methods in this section use the pre-trained models on the VCTK corpus.

3.2. Complexity and size

We first compare the complexities of models that have been recently proposed for bandwidth expansion. These include AudioUnet [3], WSRGlow [9], NU-Wave 2 [12], and NVSR [14]. We use the number of parameters when comparing the size of the models which directly translates to the amount of memory that they require in practice. The comparison is shown in Table 1. Our model is the second smallest model in this list. We then compare the computational complexity of these models. For this comparison, we use the number of required floating-point operations per second of audio (FLOP/s) during inference. Our model significantly outperforms the others, requiring 10 times fewer FLOP/s during inference compared to the next fastest model, and about 360 times fewer FLOP/s than the slowest one.

3.3. Audio Quality

Of course, computational considerations are important when it comes to a real-time deployment of such a model, but the ability to recover lost information is of utmost importance. We measure the

	Parameters (Millions)	TFLOP/s
AudioUnet	70.9	4.7
WSRGlow	229.9	1.08
NU-Wave2	1.7	1.06
NVSR	99.0	0.13
Ours	6.5	0.013

Table 1. Comparison of the size of the models based on the number of parameters in millions and their computational requirements in TeraFLOPs per second of audio processed.

performance of our model[1] as compared to a number of competing algorithms. We do so using both objective and subjective metrics.

3.3.1. Objective evaluation

For the objective evaluation, we report the average of various measures on a test set of 2 hours of speech recordings from the LibriSpeech corpus [20]. It includes 1,757 utterances, each of length 4.096 seconds. As the first objective performance metric, we use the log-spectral distance (LSD) [21] which is defined by

$$\mathrm{LSD}(\boldsymbol{x}, \hat{\boldsymbol{x}}) \triangleq \frac{1}{L}\sum_{l=0}^{L-1}\sqrt{\frac{1}{K}\sum_{k=0}^{K-1}\left(\log_{10}\frac{|\boldsymbol{X}[k,l]|^2+\epsilon}{|\hat{\boldsymbol{X}}[k,l]|^2+\epsilon}\right)^2} \quad (1)$$

where $\boldsymbol{x}$ is the target signal, $\hat{\boldsymbol{x}}$ is the model predicted signal, L is number of time blocks, l is the time block index, K is the number of frequency bins, k is the frequency bin, $\boldsymbol{X}$ is the STFT representation of $\boldsymbol{x}$, $\hat{\boldsymbol{X}}$ is the STFT of $\hat{\boldsymbol{x}}$, $|\cdot|$ denotes the magnitude of a complex number, and ϵ is a small number for numerical stability. Table 2 shows the LSD values of different methods. Lower values are better, with a theoretical lower bound at 0. As shown, our model outperforms all other models by a significant margin.

Next, we compare these models based on the Perceptual Evaluation of Speech Quality (PESQ) metric [22]. This is a routine that simulates human listener opinions in a scale of -0.5 to 4.5, with a score of 4.5 being the best. We use the PESQ score between the models' outputs and the ground truth signals. That result is also shown in Table 2, where once again our model obtains the highest score.

At this point we need to note that our model, not being a generative model, does not generate a speech output from scratch, but instead augments the input with a plausible set of missing frequency components (or other enhancements as asked to). In comparison, some of the generative models we show here generate from scratch a speech signal conditioned on the bandlimited input. This means that they are not as well positioned to perform well with objective measures that operate on the signal level. This is because they are not striving for waveform accuracy at the sample level, but instead they strive for a high likelihood that the output is a plausible speech signal. They will sound good, but not necessarily keep all of the signal elements of the input signal.

3.3.2. Subjective evaluation

To measure the subjective quality of the outputs of the models, we use result of a Mean Opinion Score (MOS) experiment obtained by the crowdsourcing methodology as described in P.808 [23]. This test will be a better approximation of how a human listener would rate these algorithms, and additionally sidesteps the generative model issues that we outlined above. We use 50 recordings randomly chosen from the test dataset. The outputs of each model are evaluated by 20 independent listeners. The average MOS score on all of the samples for each model is also shown in Table 2. MOS values range from 1 to 5, with 5 being the best rated score. Our proposed method once again achieves the highest score at 3.88 which shows it produces high-fidelity outputs even in perceptual evaluations that don't take signal details into account.

	LSD	PESQ	MOS
AudioUnet	1.68	2.05	3.65
WSRGlow	1.15	2.68	3.06
NU-Wave2	1.44	2.83	3.35
NVSR	1.13	2.78	3.75
Ours	0.85	3.68	3.88

Table 2. Comparison of reconstruction quality using objective tests LSD and PESQ, as well as a subjective test based on Mean Opinion Score (MOS).

Finally, to evaluate the effect of adversarial loss in the performance, we ran an ablation experiment. We trained our model without adversarial loss and evaluated the MOS score on the same test dataset. We received a lower MOS score of 3.78. This implies that the adversarial loss helps to improve the performance by 2.7% (0.1 points).

4. CONCLUSIONS

In this paper, we presented a new model for speech super-resolution. In addition to estimating missing high-frequency content, the proposed method also reconstructs missing low-frequency content, and can additionally correct issues with signal quantization, level imbalances, etc. Unlike previous approaches, we focused on a design that would satisfy the demanding constraints that a real-time deployment would necessitate.

Our model is small, computationally very efficient, is causal, and has minimal latency, and despite these constraints it still produces superior quality outputs as compared to competing models that are unsuitable for real-world use. As such, it can be easily deployed on-device for call enhancement, but can also run more efficiently on a server managing more simultaneous calls than alternative approaches. Although we did not talk about this here, it is possible to further simplify this model by making use of weight quantization, model compression, and other inference optimization techniques, which can produce a significantly reduced computational footprint. Due to the simple nature of this model these are easy and safe to apply, and result in efficiency that makes this model appropriate for embedded devices too.

Finally, although in this paper we focused on the specific application of super-resolution, we note that the architecture and techniques that we introduced are general enough that they can be used for other audio processing tasks as well (e.g. general audio enhancement, source separation, etc). We hope that by introducing this model we can nudge research efforts towards solutions that are suitable for deployment in real-world real-time use.

[1]The offline and streaming modes are numerically identical for our model, we are therefore presenting streaming results in the paper.

5. REFERENCES

[1] Jan-Niklas Antons, Robert Schleicher, Sebastian Arndt, Sebastian Möller, and Gabriel Curio, "Too tired for calling? a physiological measure of fatigue caused by bandwidth limitations," in *2012 Fourth International Workshop on Quality of Multimedia Experience*, 2012, pp. 63–67.

[2] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang, "Image super-resolution using deep convolutional networks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 2, pp. 295–307, 2015.

[3] Volodymyr Kuleshov, S Zayd Enam, and Stefano Ermon, "Audio super-resolution using neural nets," in *ICLR 2017 - Workshop of International Conference on Learning Representations*, 2017.

[4] Sefik Emre Eskimez and Kazuhito Koishida, "Speech super resolution generative adversarial network," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 3717–3721.

[5] Sefik Emre Eskimez, Kazuhito Koishida, and Zhiyao Duan, "Adversarial training for speech super-resolution," *IEEE Journal of Selected Topics in Signal Processing*, vol. 13, no. 2, pp. 347–358, 2019.

[6] Shichao Hu, Bin Zhang, Beici Liang, Ethan Zhao, and Simon Lui, "Phase-aware music super-resolution using generative adversarial networks," in *Interspeech, 21th Annual Conference of the International Speech Communication Association*, 2020, pp. 4074–4078.

[7] Kundan Kumar, Rithesh Kumar, Thibault de Boissiere, Lucas Gestin, Wei Zhen Teoh, Jose Sotelo, Alexandre de Brébisson, Yoshua Bengio, and Aaron C Courville, "Melgan: Generative adversarial networks for conditional waveform synthesis," in *Advances in Neural Information Processing Systems*. 2019, vol. 32, p. 14881–14892, Curran Associates, Inc.

[8] Ivan Kobyzev, Simon JD Prince, and Marcus A Brubaker, "Normalizing flows: An introduction and review of current methods," *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 11, pp. 3964–3979, 2020.

[9] Kexun Zhang, Yi Ren, Changliang Xu, and Zhou Zhao, "WSRGlow: A Glow-Based Waveform Generative Model for Audio Super-Resolution," in *Interspeech, 21st Annual Conference of the International Speech Communication Association*, 2021, pp. 1649–1653.

[10] Junhyeok Lee and Seungu Han, "Nu-wave: A diffusion probabilistic model for neural audio upsampling," in *Interspeech, 22nd Annual Conference of the International Speech Communication Association*, 2021, pp. 1634–1638.

[11] Jonathan Ho, Ajay Jain, and Pieter Abbeel, "Denoising diffusion probabilistic models," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, Eds., 2020, vol. 33, pp. 6840–6851.

[12] Seungu Han and Junhyeok Lee, "Nu-wave 2: A general neural audio upsampling model for various sampling rates," in *Interspeech, 23rd Annual Conference of the International Speech Communication Association*, 2022.

[13] Jungil Kong, Jaehyeon Kim, and Jaekyoung Bae, "Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, Eds. 2020, vol. 33, pp. 17022–17033, Curran Associates, Inc.

[14] Haohe Liu, Woosung Choi, Xubo Liu, Qiuqiang Kong, Qiao Tian, and DeLiang Wang, "Neural vocoder is all you need for speech super-resolution," in *Interspeech, 23rd Annual Conference of the International Speech Communication Association*, 2022.

[15] "G.711 : Pulse code modulation (PCM) of voice frequencies," in *The ITU Telecommunication Standardization Sector (ITU-T), Archived from the original on 2019-06-17. Retrieved 2019-11-11.*, `https://www.itu.int/rec/T-REC-G.711`.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[17] Hugo Touvron, Piotr Bojanowski, Mathilde Caron, Matthieu Cord, Alaaeldin El-Nouby, Edouard Grave, Gautier Izacard, Armand Joulin, Gabriel Synnaeve, Jakob Verbeek, et al., "Resmlp: Feedforward networks for image classification with data-efficient training," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[18] Haohe Liu, Qiuqiang Kong, Qiao Tian, Yan Zhao, DeLiang Wang, Chuanzeng Huang, and Yuxuan Wang, "Voicefixer: Toward general speech restoration with neural vocoder," 2021.

[19] Junichi Yamagishi, Christophe Veaux, and Kirsten MacDonald, "CSTR VCTK Corpus: English multi-speaker corpus for CSTR voice cloning toolkit (version 0.92) [sound]. University of Edinburgh. The Centre for Speech Technology Research (CSTR)," 2019, `https://doi.org/10.7488/ds/2645`.

[20] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur, "Librispeech: An asr corpus based on public domain audio books," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, pp. 5206–5210.

[21] Lawrence Rabiner and Biing-Hwang Juang, *Fundamentals of speech recognition*, Prentice-Hall, Inc., 1993.

[22] "ITU-T, P.862: Perceptual evaluation of speech quality (PESQ): An objective method for end-to-end speech quality assessment of narrow-band telephone networks and speech codecs," 2001.

[23] ITU-T, "Recommendation p.808: Subjective evaluation of speech quality with a crowdsourcing approach," 2018.