Time-Scale Modification with Temporal Compressing Networks

Ernie Chu*
Research Center for Information Technology Innovation, Academia Sinica
Taipei, Taiwan
shchu@citi.sinica.edu.tw

Ju-Ting Cheng*
National Cheng Kung University
Tainan, Taiwan

Chia-Ping Chen
National Sun Yat-sen University
Kaohsiung, Taiwan
cpchen@cse.nsysu.edu.tw

ABSTRACT

We propose a novel approach for time-scale modification of audio signals. Unlike traditional methods that rely on the framing technique or the short-time Fourier transform to preserve the frequency during temporal stretching, our neural network model encodes the raw audio into a high-level latent representation, dubbed Neuralgram, where each vector represents 1024 audio sample points. Due to a sufficient compression ratio, we are able to apply arbitrary spatial interpolation of the Neuralgram to perform temporal stretching. Finally, a learned neural decoder synthesizes the time-scaled audio samples based on the stretched Neuralgram representation. Both the encoder and decoder are trained with latent regression losses and adversarial losses in order to obtain high-fidelity audio samples. Despite its simplicity, our method has comparable performance compared to the existing baselines and opens a new possibility in research into modern time-scale modification. Audio samples can be found on our website[1].

CCS CONCEPTS

• **Applied computing** → **Sound and music computing**; • **Computing methodologies** → *Temporal reasoning*.

KEYWORDS

time-scale modification, GAN, neural vocoder

ACM Reference Format:
Ernie Chu, Ju-Ting Cheng, and Chia-Ping Chen. 2023. Time-Scale Modification with Temporal Compressing Networks. In *Under review*. ACM, New York, NY, USA, 7 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

With the advancement of technologies and digitalization, we can store and reproduce multimedia content. We can even manipulate materials in ways that we couldn't imagine before. For example, image resizing and video editing change the dimensions of digital pictures spatially and temporally, respectively. Another ubiquitous application regarding audio signals is time-scale modification (TSM), which is used in our daily life. It is also known as playback speed control in video streaming platforms, such as YouTube.

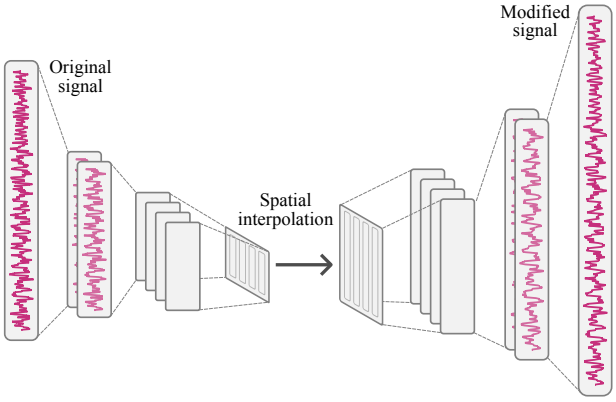

Figure 1: Overall architecture for the proposed method. TSM-Net temporally stretches an audio signal in a simple approach.

With the power of artificial intelligence (AI) and modern computation hardware, pragmatic AI tools are emerging in multimedia domains, such as image super-resolution [16] and motion estimation, motion compensation [1], etc. However, as far as we know, methods leveraging AI to refine the TSM algorithm and improve the quality of the synthetic audio, have not been well-studied.

Naive resampling of the raw audio signal to get scaled versions causes serious pitch-shifting effects because the wavelength of each frequency component scales proportionally with the duration of the overall sample. TSM has been a research topic aiming to alleviate the pitch-shifting effects when stretching audio samples. Several methods have been proposed, including time-domain approaches and spectral approaches. The ultimate goal of TSM is to synthesize high-quality audio that is perceptually indistinguishable from real-world recordings. Despite the efforts of early research and wide applications in the market, existing methods merely achieve this goal for some speech-only audio. When it comes to more complicated content like music, users face a trade-off in quality between the harmonic source and percussive source. Our aim is to develop a universal method that can accommodate all types of audio content while minimizing the occurrence of artifacts to the greatest extent possible.

*Part of the work was done while Ernie and Ju-Ting were students at National Sun Yat-sen University.
[1]https://tsmnet-mmasia23.github.io.

Under review, ,

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

In our work, we use an architecture similar to the autoencoder [13], which encodes the data into high-level (and typically low-dimensional) latent vectors and faithfully reconstructs the original data. In our model, the dimension of the latent vectors is 1024 times smaller than the original one, which means one sample in the latent vector can represent more than an entire wave in the raw audio waveform. Since the smallest unit encompasses more than one wave, we can apply arbitrary spatial interpolation on the latent vector to stretch the audio, without worrying about the changes in frequency components and the pitches. Finally, we decode the resized latent vector to obtain the time-scaled audio waveform. Our overall architecture is illustrated in Figure 4. To synthesize high-fidelity audio samples, we use the adversarial losses to train our autoencoder [7]. Multi-scale discriminative networks are employed to distinguish between real and generated data. We will delve into the details of the training in Section 3.

In summary, our main contribution is to propose a simple yet powerful data-driven method that shows comparable performance on various kinds of audio contents. We would also like to reignite the research on TSM in the Machine Learning era. Through the improvement of this fundamental tool in audio processing, we believe a wide range of applications can directly benefit.

2 RELATED WORK

In this section, we comprehensively review the existing approaches for TSM, both in the time domain and the spectral domain. Additionally, we review the fundamental tool in audio generation, the neural vocoder. This tool would be an essential component for the proposed method.

2.1 Time-domain approach

The main idea of TSM is that instead of scaling the raw waveforms on the time axis, which leads to pitch shifts due to the changes in wavelengths, we segment the audio into small chunks of fixed length, also known as frames or windows, in order to preserve the wavelength. The adjacent frames are overlapped and rearranged to minimize boundary breakage after processing, as shown in Figure 2.

The original distance between the start of each frame is called the analysis hop size. Once the frames are relocated, this distance becomes the synthesis hop size, and the ratio of the analysis hop size to the synthesis hop size determines the speed at which the audio is accelerated or decelerated [4]. Additionally, the Hann window [5] is usually applied to each analysis frame to maintain the amplitude of the overlapped areas. One of the main challenges in this approach is the harmonic alignment problem, which is illustrated in Figure 3.

When significant periodicities are present, an unconstrained ratio of the analysis to synthesis hop size can cause a discrepancy with the original waveform. Specifically, the phases of the frequency components within the frames do not synchronize properly, resulting in significant interference. Several solutions have been proposed to address the synchronization problem [9, 21, 30]. However, the resulting sound is often unnatural and includes noticeable clipping artifacts. Moreover, time-domain TSM only preserves the most prominent periodicity. For audio with a wide range of frequency compositions, such as pop music, symphony, and orchestra, less prominent sounds are often discarded in the process.

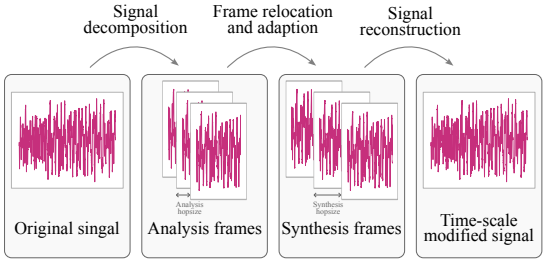

Figure 2: Generic processing pipeline of time-domain time-scale modification (TSM) procedures.

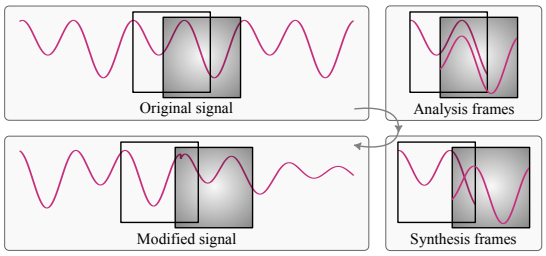

Figure 3: An illustration of the harmonic alignment problem. The black boxes demonstrate the rearrangement of the frames. An unconstrained scale ratio would lead to serious interference.

2.2 Spectral-domain approach

Another approach processes the audio within the spectral domain using the short-time Fourier transform (STFT) to convert the frequency information contained in the raw waveform into a more semantic representation in complex numbers [15]. Further, the magnitude and phase components can be derived. Unfortunately, unlike the magnitudes, which provide constructive and straightforward audio features, the phases are relatively complex and challenging to model. Moreover, due to the heavy correlation between each phase bin, an additional phase vocoder [6] is required to estimate phases and the instantaneous frequencies after carefully relocating STFT bins to prevent specific artifacts known as "phasiness." Although some refined methods [12, 19, 22] enhance both the vertical and horizontal coherence of the phase, the spectral representation is not inherently scalable, and the iterative phase propagation process in the phase vocoder poses a significant computational overhead.

2.3 Neural vocoder

Modeling audio is not a trivial task for neural networks. To illustrate this, let's consider image generation as an example. DCGAN [25] is a generative neural network used for synthesizing realistic images with dimensions of $3 \times 64 \times 64$ pixels. It needs to generate a total of 12,288 pixels. On the other hand, a stereo audio clip lasting 5 seconds, sampled at a rate of 22050 Hz, consists of 220,500 samples.

Furthermore, while each pixel in an image is stored in 8 bits, each audio sample is stored in 16 bits, providing 256 times more possible values than an image pixel. Another option to reduce complexity is to decrease the sampling rate. However, according to the Nyquist-Shannon sampling theorem, a low sampling rate leads to significant aliasing.

Models that directly generate raw audio waveforms are referred to as vocoders. A vocoder can utilize high-level abstract features, such as linguistic features or spectrograms, for conditioning. The spectrogram represents the magnitude component obtained from the output of STFT. It is easy to model due to its smooth variations in frequency composition over time. As mentioned in Section 2.1, the phases are relatively hard to estimate. Therefore, in applications such as text-to-speech (TTS) pipelines [27], the network often generates the speech spectrogram from given texts and then utilizes a vocoder to synthesize the corresponding audio waveform. Early vocoders, such as Griffin-Lim [8] and WORLD [20], were developed.

2.3.1 Autoregressive and flow-based neural vocoder. The pioneers of modern neural-based vocoders include WaveNet [29], which predicts the distribution for each audio sample conditioned on all previous ones. However, the autoregressive model runs too slowly to be applied to real-time applications. FloWaveNet [11] and WaveGlow [24] are neural vocoders that are based on bipartite transforms. They present a faster inference speed and high-quality synthetic audio but require larger models and more parameters to be as expressive as the autoregressive models, thus making them harder to train.

2.3.2 GAN-based neural vocoder. WaveGAN [3], MelGAN [14], and VocGAN [32] employ the generative adversarial network [7] training architecture in which the discriminators are used to measure the divergence of synthetic audio and the real audio and help the generator network synthesize audio samples as realistic as possible. The discriminators usually work at multiple scales to handle different frequency bands in the audio data. This kind of approach allows smaller models to generate high-fidelity audio samples.

3 METHOD

In this section, we provide a comprehensive discussion of Neuralgram, a novel representation of audio signals, and how it can be used in the TSM. Additionally, we present a brief comparison between Neuralgram and other common representations such as the spectrogram. Finally, we introduce the proposed TSM-Net architecture, which consists of an autoencoder and multi-scale discriminators. The architecture is depicted in Figure 4.

3.1 Latent representation

We propose a new representation for audio called Neuralgram to provide a novel approach to TSM. The Neuralgram is a temporally compressed feature map extracted from the middle of a neural autoencoder. A Neuralgram is applicable to TSM only when the following conditions are met:

(1) An encoder-decoder pair that faithfully reconstructs the audio waveform.
(2) A compression ratio that is high enough to put an entire sinusoid of the lowest frequency perceivable into a single sample point in the Neuralgram.

Instead of directly scaling the raw waveform, which leads to pitch shifting of the audio, we follow a process where we encode the raw waveform as a real-valued Neuralgram, then scale the Neuralgram accordingly. Finally, we decode the scaled Neuralgram to obtain temporally stretched audio while preserving the original pitch. The process is illustrated in Figure 5. Formally speaking, given an input audio x, we obtain the time-scaled signal $\hat{x}$ by

$$\hat{x} = \mathcal{A}_D(S(\mathcal{A}_E(x), r)), \tag{1}$$

where we decompose an optimized autoencoder $\mathcal{A}$ into the encoder $\mathcal{A}_E$ and the decoder $\mathcal{A}_D$, and a scaling function S configured by a factor r is applied to the Neuralgram produced by $\mathcal{A}_E$. In our context, S is a basic cubic interpolation for simplicity. However, S can also be expanded to a more advanced neural technique for super-resolution. Because each sample in the Neuralgram encodes more than an entire sinusoid for each frequency component, resizing the Neuralgram can replicate entire sinusoids in the reconstructed waveform instead of altering their wavelengths and frequencies.

3.1.1 Neuralgram vs. Spectrogram. In the literature, most of the works related to the neural vocoder use the spectrogram family as prior conditions. Why should we consider adopting a new representation? These traditional representations encode various frequency information into the same number of sample points in the latent space. This results in different upsample scaling ratios during the decoding process, as each frequency has its own wavelength. The transformation function with variable upsampling ratios is harder to approximate with convolutional generative models. On the contrary, our Neuralgram encodes different frequency components proportionally, which is intuitive for convolutional networks, making the model easier to train.

3.2 The TSM-Net model

3.2.1 Architecture. Our model is adapted from the MelGAN [14]. In the original generator, the input is a mel-spectrogram, which would be upsampled 256× to a raw audio waveform. In the proposed model, we increase the upsampling rate to 1024× to capture the entire sinusoid within a single sample point. This is because in audio with a sampling rate of 22050Hz, the lowest frequency perceived by human ears, a 20Hz sinusoid occupies 1102.5 sample points. The upsampling process involves five stages of upsampling blocks: 8×, 8×, 4×, 2×, and 2×.

Additionally, we prepend a mirror of the modified generator to the front of the generator to obtain the full autoencoder model A. Both the encoder A_E and the decoder $\mathcal{A}_D$ are initiated by an aggregating convolutional layer, followed by a deries of downsampling/upsampling stages. Each stage is composed of a dilated downsampling/upsampling convolutional layer and a residual block that also contains a dilated convolutional block and a skip-connection. Formally, the temporal dimensionality of the input signal x satisfy

$$\dim_T(x) = \dim_T(\mathcal{A}_D(\mathcal{A}_E(x))) \approx \frac{SR}{20} \dim_T(\mathcal{A}_E(x)), \tag{2}$$

where SR is the sampling rate, which is set to 22050Hz throughout this paper. As discussed in the MelGAN paper, we also use kernel

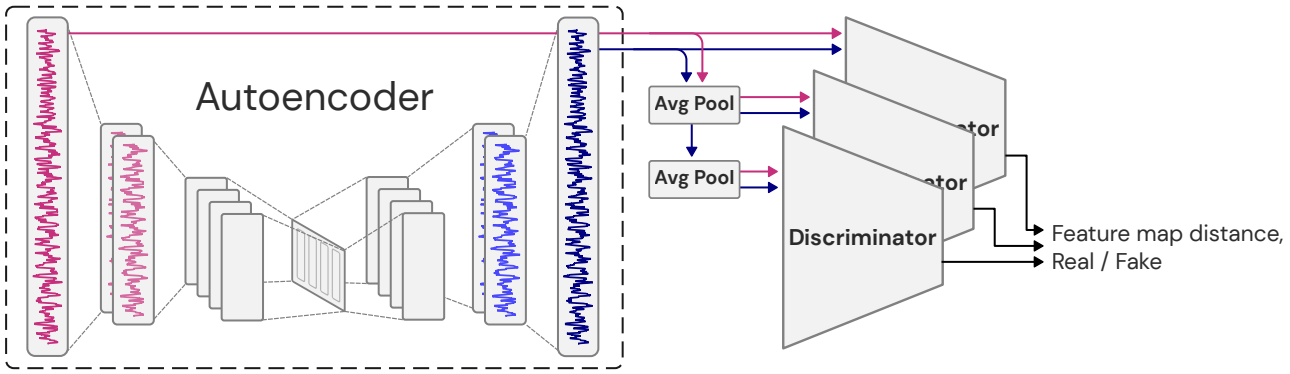

Figure 4: Overall architecture for the proposed training strategy. The number of channels increases with each layer of convolutional neural networks in the encoder. Both of the input and output of the autoencoder are one-dimensional audio signals. They are consumed by multi-scale discriminators to produce binary predictions. The feature maps of the discriminators are also used during the training to calculate the distance between input and reconstructed audio signal in discriminators' latent space.

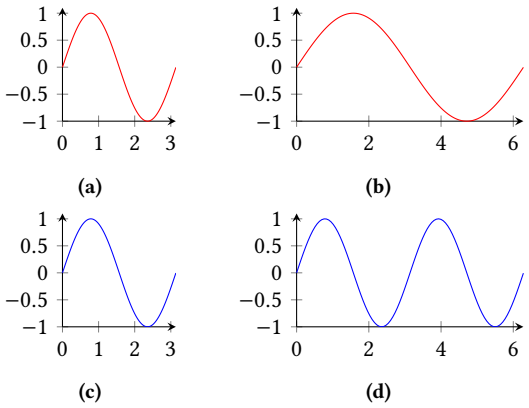

Figure 5: An illustration for the desire TSM. While doubling the audio duration, the frequency should be kept intact. The original signal contains the sinusoid for a single frequency 5a, 5c.The erroneous signal 5b is the result of directly scaling on the raw waveform, which changes the wavelength of the sinusoid and produces pitch shifting. The desired behavior 5d can be achieved by scaling on the Neuralgram, which compresses the entire sinusoid into one vector. We then repeat the vector and produce two waves through the decoder.

size as a multiple of stride to avoid checkerboard artifacts [23], and weight normalization [26] is also used after each layer to improve the sample quality.

3.2.2 Adversarial losses. In order to improve the high-frequency fidelity, we employ the same discriminators as the ones in the MelGAN. Three discriminators ($\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3$) operate at different scales simultaneously. With the exception of $\mathcal{D}_1$, which operates at the original scale, downsampling is performed beforehand using stridden average pooling with a kernel size of 4. This arrangement allows each discriminator to more effectively learn features for different frequency ranges of the audio. Next, we formulate the adversarial losses for training the autoencoder $\mathcal{A}$, following the MelGAN approach and using the hinge loss version of the GAN objective [17] to penalize only the unstable data distribution.

In order to provide additional information to the autoencoder for utilizing the input condition and prevent mode collapse, we also incorporate the feature-matching loss $\mathcal{L}_{FM}$. This loss minimizes the L1 norm between the discriminator's feature maps of the input and the reconstructed audio. Through empirical observation, we found that minimizing the distance between raw waveforms produces audible artifacts. During the training of the autoencoder, we accumulate the feature matching losses at each intermediate layer of all discriminators, such as

$$\mathcal{L}_{FM}(\mathcal{A}, \mathcal{D}_k) = \mathbb{E}_x \left[\sum_{i=1}^{T} \frac{1}{N_i} \| \mathcal{D}_k^{(i)}(x) - \mathcal{D}_k^{(i)}(\mathcal{A}(x)) \|_1 \right], \quad (3)$$

where $\mathcal{D}_k^{(i)}$ represents the ith layer's feature map output of the kth discriminator. N_i is a normalization factor and denotes the number of units in each layer. T represents the number of layers in each discriminator. Our final training objective is given by

$$\min_{\mathcal{D}_k} \sum_{k=1}^{3} \mathbb{E}_x \left[\min(0, 1 - \mathcal{D}_k(x)) + \min(0, 1 + \mathcal{D}_k(\mathcal{A}(x))) \right], \quad (4)$$

$$\min_{\mathcal{A}} \left(\mathbb{E}_x \left[- \sum_{k=1}^{3} \mathcal{D}_k(\mathcal{A}(x)) \right] + \lambda \sum_{k=1}^{3} \mathcal{L}_{FM}(\mathcal{A}, \mathcal{D}_k) \right), \quad (5)$$

where λ controls the strength of $\mathcal{L}_{LM}$ and is set to 10 by default. According to [10, 18], the noise input is not necessary when the conditioning information is very strong in the generative model.

Table 1: Mean opinion score on three datasets.

Method	FMA	Musicnet	VCTK	Average
WSOLA [30]	2.8	3.31	3.12	3.08
PV-TSM [19]	2.89	3.24	3.09	3.07
TSM-Net (ours)	2.87	3.20	3.12	3.06

4 EXPERIMENT

We extensively test our model on pop music, classical music, and speech data. We present comprehensive statistics of the feedback collected from our user study, showing that despite its simplicity, the proposed method demonstrates comparable or superior performance on various kinds of audio data.

4.1 Dataset

Since our framework does not require any human-annotated data, our model can be trained on any audio dataset. We consider three datasets with various audio contents in our experiment. FMA [2] contains a total length of 343 days of Creative Commons-licensed audio, arranged in a hierarchical taxonomy of 161 genres of pop music. Musicnet [28] is a collection of 330 freely-licensed classical music recordings. CSTR VCTK Corpus [31] includes speech data uttered by 110 English speakers with various accents. All of the audio is resampled to 22050Hz.

4.2 Implementation details

We train the model on a single Nvidia Tesla P100 for a week with each dataset. When training on a more complex dataset, such as the FMA dataset, which contains a large amount of Pop music, the wide frequency range in Pop music makes training the GAN from scratch more difficult. We pre-train the autoencoder on the classical music dataset, Musicnet, to stabilize the training progress. In contrast, a pre-trained discriminator cannot effectively guide the autoencoder in performing a proper reconstruction. We attach the source code[2] accompanying this paper to encourage reproducibility and enhancements.

4.3 Time-scaled modification

4.3.1 Setup. To effectively evaluate the quality of the stretched audio, we employ a Monte Carlo approach to collect user feedback on the test samples. We randomly select twenty 10-second (at most) samples from the FMA, Musicnet, and VCTK datasets for the listening test. In each round of the test, 10 out of the total of 60 test samples are drawn to be presented to the participants. We varied the speed of the audio using a factor r, randomly chosen from intervals $[0.5, 0.95]$ with an interval of 0.05 and $[1.1, 2.0]$ with an interval of 0.1. To evaluate the performance of the generated audio, we utilized the mean opinion score (MOS) and recruited 68 participants with diverse backgrounds, who contributed ratings for a total of 580 audio samples. For each audio sample, participants were presented with the original audio and five audio samples stretched to the same speed using various methods. They were asked to rate the generated audio on a scale of 1 (poor) to 5 (excellent) in terms of pitch correctness and overall audio quality. The listening test can be experienced on our website[3]

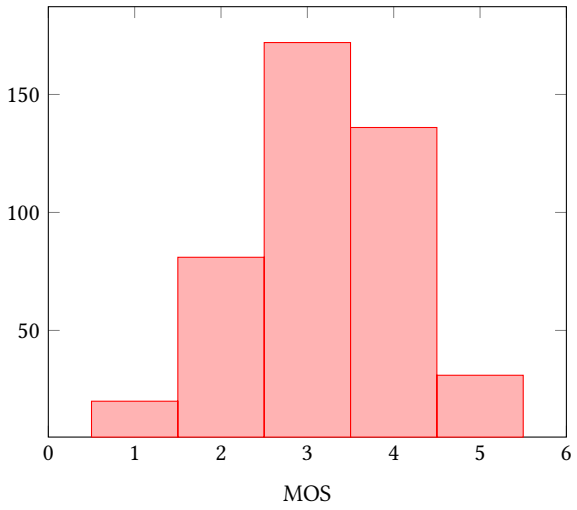

Figure 6: The MOS histogram for TSM-Net.

4.3.2 Results. Table 1 indicates that despite the simplicity of our method, the participants consider the samples generated by our method comparable to the state-of-the-art approach on both musical datasets and speech datasets. Furthermore, Figure 6 shows the histogram of MOS ratings for the audio samples stretched by the proposed method. The collected data roughly follow a normal distribution, indicating the reliability of our subjective test.

4.4 Study on different compression ratios

As mentioned in Section 3.2, a compression ratio (CR) of 1024× is large enough for the lowest perceivable frequency. We would like to investigate whether a lower CR results in pitch-shifting in TSM. We examine different models with CRs of 256×, 512×, and the original 1024×. Intuitively, a lower CR should result in smaller reconstruction errors due to reduced information loss. However, the pitch-shifting effect occurs across a wider range of frequency bands when the CR is small. The pitch-shifting effect gradually emerges at lower frequencies and seldom occurs in the high frequencies. We recommend listening to the audio samples on our website[4] to understand this interesting phenomenon.

In addition to the qualitative evaluation, we also conduct a listening test as set up in Section 4.3, in which we compare the audio samples stretched by the three variants of TSM-Net. The results in Table 2 show that sufficiently high compression ratios are crucial to prevent pitch-shifting and ensure acceptable audio quality. Figure 7 also indicates that the 1024× CR setting makes TSM-Net stretch audio to various speeds with better quality preservation. The figure also shows that our models perform worse when stretching the audio toward more extreme speeds. We hypothesize that the problem mainly arises from an overly naive interpolation algorithm. We leave the investigation of this issue for future work.

[2]https://github.com/tsmnet-mmasia23/tsmnet.

[3]https://tsmnet-mmasia23.web.app.

[4]https://tsmnet-mmasia23.github.io.

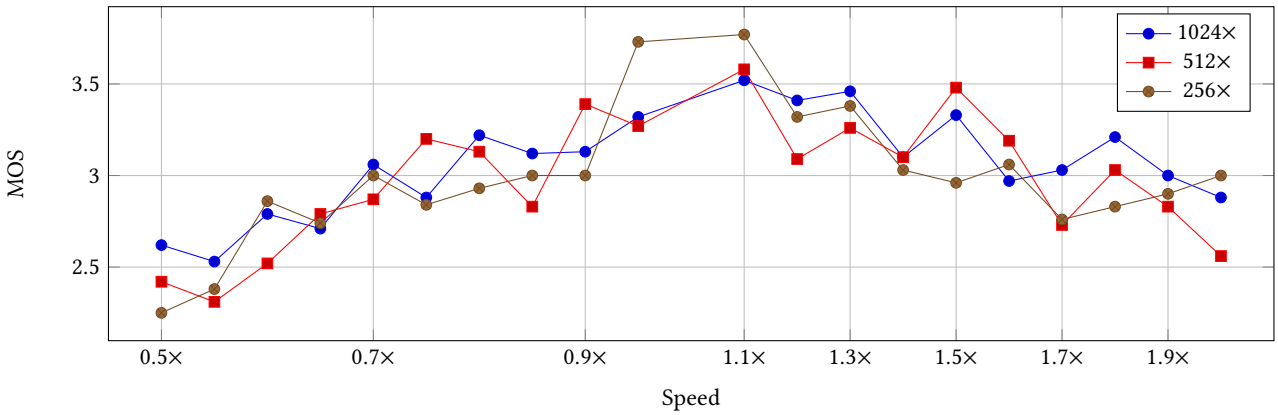

Figure 7: Average MOS on each speed. 256× and 512× indicate an architecture with a compressing ratio of 256 and 512, respectively. The default compressing ratio of the proposed method is 1024×.

Table 2: Mean opinion score on three datasets. 256× and 512× indicate an architecture with a compressing ratio of 256 and 512, respectively. The default compressing ratio of the proposed method is 1024×.

Compression ratio	FMA	Musicnet	VCTK	Average
256×	2.78	3.19	2.98	2.98
512×	2.78	3.14	3.01	2.97
1024× (ours)	2.87	3.20	3.12	3.06

Table 3: Average inference time per 1-sec audio generation.

Method	Average inference time
WSOLA [30]	16.4807 ms/sec
PV-TSM [19]	15.4474 ms/sec
TSM-Net 1024×	1.8954 ms/sec
TSM-Net 512×	1.7550 ms/sec
TSM-Net 256×	1.5599 ms/sec

4.5 Inference time

As a neural network-based approach, our method can easily leverage the highly parallel GPU computation unit. To study the processing speed improvement, we record the inference time for generating 1200 audio samples used in the listening test. We report the numbers in milliseconds per second of audio signal for the baseline methods and the variants of TSM-Net. As shown in Table 3, our method reduces the computation time by a large margin compared to the baseline methods, as it does not rely on time-consuming phase alignment or phase propagation, and it can leverage the power of GPU.

4.6 Additional training techniques

We also use two additional metrics to monitor our training but they are not included in the training losses.

(1) Audio Reconstruction (AR). AR is the L1 norm between the real and reconstructed raw audio waveform.
(2) Neuralgram Reconstruction (NR). NR is the L1 norm between the Neuralgrams encoded from the real and reconstructed raw audio waveform, i.e., the reconstructed Neuralgram has 1.5 passes through the autoencoder.

We note that the model is not guaranteed to converge in each run. The ideal loss for the discriminator tends to fluctuate around 6. If the discriminator's loss decreases drastically, both the autoencoder's loss and the feature matching loss fail to return to a healthy value, resulting in rapid divergence and poor quality. We can verify this phenomenon by examining AR and NR. Both reconstruction losses increase after the discriminator gains dominance in the training process. Notably, we can perceive background noises in the reconstructed audio samples.

5 CONCLUSION AND LIMITATION

In this paper, we introduce a custom neural network model and a novel audio representation for the time-scale modification (TSM). The proposed method demonstrates a simple yet efficient approach to manipulating audio contents temporally using the power of the neural compressor. Our method mitigates or solves the issues found in the traditional TSM, such as the harmonic alignment problem, the background sound loss, and the phasiness. However, our method sometimes produces other artifacts, which make the human evaluators consider the stretched audio less preferred. We attribute this issue to the insufficient model capacity and the naive choice of interpolation, which are practical limitations under our theoretical proposition. To improve audio quality, our method can be further incorporated with advancements in other domains, such as neural interpolation.

While the research of TSM had been silent for a long time, this ubiquitous technology is now used in our everyday life. We believe our work opens new possibilities for state-of-the-art TSM algorithms, allowing for further advancements and applications in this field.

ACKNOWLEDGMENTS

This work is supported by National Science and Technology Council (formerly Ministry of Science and Technology), Taiwan (R.O.C), under Grants no. 110-2813-C-110-050-E.

REFERENCES

[1] Wenbo Bao, Wei-Sheng Lai, Xiaoyun Zhang, Zhiyong Gao, and Ming-Hsuan Yang. 2021. MEMC-Net: Motion Estimation and Motion Compensation Driven Neural Network for Video Interpolation and Enhancement. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (2021).

[2] Kirell Benzi, Michaël Defferrard, Pierre Vandergheynst, and Xavier Bresson. 2016. FMA: A Dataset For Music Analysis. *arXiv preprint arXiv:1612.01840* (2016).

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

[4] Jonathan Driedger and Meinard Müller. 2016. A Review of Time-Scale Modification of Music Signals. *Applied Sciences* (2016).

[5] O.M. Essenwanger. 1986. *Elements of Statistical Analysis*.

[6] J. L. Flanagan and R. M. Golden. 1966. Phase Vocoder. *Bell System Technical Journal* (1966).

[7] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[8] D. Griffin and Jae Lim. 1984. Signal estimation from modified short-time Fourier transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing* (1984).

[9] Don Hejna and Bruce R Musicus. 1991. The SOLAFS time-scale modification algorithm. *Bolt, Beranek and Newman (BBN) Technical Report* (1991).

[10] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. 2017. Image-To-Image Translation With Conditional Adversarial Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[11] Sungwon Kim, Sang-Gil Lee, Jongyoon Song, Jaehyeon Kim, and Sungroh Yoon. 2019. FloWaveNet : A Generative Flow for Raw Audio. In *Proceedings of Machine Learning Research (PMLR)*.

[12] Sebastian Kraft, Martin Holters, Adrian von dem Knesebeck, and Udo Zölzer. 2012. Improved PVSOLA time-stretching and pitch-shifting for polyphonic audio. In *Proceedings of the International Conference on Digital Audio Effects (DAFx)*.

[13] Mark A. Kramer. 1991. Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal* (1991).

[14] Kundan Kumar, Rithesh Kumar, Thibault de Boissiere, Lucas Gestin, Wei Zhen Teoh, Jose Sotelo, Alexandre de Brébisson, Yoshua Bengio, and Aaron C Courville. 2019. MelGAN: Generative Adversarial Networks for Conditional Waveform Synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[15] J. Laroche and M. Dolson. 1999. Improved phase vocoder time-scale modification of audio. *IEEE Transactions on Speech and Audio Processing* (1999).

[16] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. 2017. Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[17] Jae Hyun Lim and Jong Chul Ye. 2017. Geometric GAN. *arXiv preprint arXiv:1705.02894* (2017).

[18] Michael Mathieu, Camille Couprie, and Yann LeCun. 2016. Deep multi-scale video prediction beyond mean square error. *arXiv preprint arXiv:1511.05440* (2016).

[19] Alexis Moinet and Thierry Dutoit. 2011. PVSOLA: A phase vocoder with synchronized overlap-add. In *Proceedings of the International Conference on Digital Audio Effects (DAFx)*.

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

[21] Eric Moulines and Francis Charpentier. 1990. Pitch-synchronous waveform processing techniques for text-to-speech synthesis using diphones. *Speech Communication* (1990).

[22] Frederik Nagel and Andreas Walther. 2009. A Novel Transient Handling Scheme for Time Stretching Algorithms. In *Audio Engineering Society Convention*.

[23] Augustus Odena, Vincent Dumoulin, and Chris Olah. 2016. Deconvolution and Checkerboard Artifacts. *Distill* (2016).

[24] Ryan Prenger, Rafael Valle, and Bryan Catanzaro. 2019. Waveglow: A Flow-based Generative Network for Speech Synthesis. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

[25] Alec Radford, Luke Metz, and Soumith Chintala. 2016. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *arXiv preprint arXiv:1511.06434* (2016).

[26] Tim Salimans and Durk P Kingma. 2016. Weight Normalization: A Simple Reparameterization to Accelerate Training of Deep Neural Networks. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[27] Xu Tan, Tao Qin, Frank Soong, and Tie-Yan Liu. 2021. A Survey on Neural Speech Synthesis. *arXiv preprint arXiv:2106.15561* (2021).

[28] John Thickstun, Zaid Harchaoui, Dean P. Foster, and Sham M. Kakade. 2018. Invariances and Data Augmentation for Supervised Music Transcription. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

[29] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. 2016. WaveNet: A Generative Model for Raw Audio. *arXiv preprint arXiv:1609.03499* (2016).

[30] W. Verhelst and M. Roelands. 1993. An overlap-add technique based on waveform similarity (WSOLA) for high quality time-scale modification of speech. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*.

[31] Junichi Yamagishi, Christophe Veaux, and Kirsten MacDonald. 2019. CSTR VCTK Corpus: English Multi-speaker Corpus for CSTR Voice Cloning Toolkit (version 0.92).

[32] Jinhyeok Yang, Junmo Lee, Youngik Kim, Hoonyoung Cho, and Injung Kim. 2020. VocGAN: A High-Fidelity Real-time Vocoder with a Hierarchically-nested Adversarial Network. *Interspeech* (2020).