TSM-Net: 以對抗式時序壓縮自編碼器為基礎的音訊變速演算法

TSM-Net: Temporal Compressing Autoencoder with Adversarial Losses for Time-Scale Modification on Audio Signals

國立中山大學資訊工程學系
110 學年度大學部專題製作競賽

組員：B073040018 朱劭璿
B072010029 陳居廷

指導教師：陳嘉平教授

Abstract

We propose a novel machine learning algorithm for time-scale modification in the work to solve the current problem in the traditional time-scale modification (TSM). TSM is a historic research topic, in which we aim to modify the duration and speed of an audio without changing its pitch. Traditional TSM algorithms have utilized the modification on both time domain and spectral domain, making the audio preserve the original features and changing the overall duration at the same time. However, for example, on the well-known on-line video platform, YouTube, the TSM is not good enough to provide a modification such that the human ear cannot distinguish between the modified audio and the natural audio. The most common artifacts in the modified audio include phase jump artifacts, transient doubling/shuttering, secondary sources lost, etc. We' ve taken the notion from the well-known TSM algorithm to design our neural network model. We use recurrent neural networks and convolutional neural networks to generate the time-stretch audio. We also take advantage of autoencoder architecture and generative adversarial networks to improve the realness of both detailed and aggregated features in the audio. After we thoroughly evaluate the effectiveness of our algorithm, we plan to deploy it on a Web application to enable the reachability to more users and prove the feasibility of our algorithm and implementation.

Contents

1 Introduction **3**

2 Related work **3**

2.1 Time domain method . . . 4

2.1.1 OLA . . . 5

2.1.2 WSOLA . . . 6

2.2 Frequency domain method . . . 7

2.2.1 PV-TSM . . . 8

2.3 Hybrid method . . . 8

2.3.1 HPSS via median filtering . . . 10

2.3.2 HPSS via autoencoder . . . 10

3 Methodology **11**

3.1 Feature disentangle . . . 11

3.1.1 Instance normalization . . . 11

3.1.2 Adaptive instance normalization . . . 13

3.1.3 Evaluation of disentanglement . . . 13

3.2 Tempo estimation . . . 14

3.2.1 Architecture . . . 14

1 Introduction

Time-scale modification (TSM) is a field with a long history of research and applications. Dating back to the days of tape recording, attempts have been made to stretch the timeline of pre-recorded audio tracks. Modern applications such as YouTube’ s playback speed adjustment, commercial recording studio softwares’ speed adjustment, and speaking speed adjustment in language learning platforms. These applications aim to modify the playback speed without tuning the pitch of an audio. If we naively resample the audio to get the desired duration of an audio segment, the frequency of sound would change accordingly and consequently changing the pitch. The traditional TSM algorithms do change the playback speed without affecting the pitch, however, to this day, none of the TSM algorithms provides a modification such that the human ear cannot distinguish between the modified audio and the natural audio. The major artifacts in the modified audio include phase jump artifacts, transient doubling/shuttering, secondary sources lost.

The phase jump artifact is the discontinuity between the processing frames. The transient doubling is the duplicate sound of typically percussions produced by the adjacent frames which overlap on the same audio samples when slowing down the audio, and vice versa for transient shuttering. The secondary sources lost are most common on the polyphonic input signals such as recording of orchestral music. For some TSM algorithms, they can only preserve the most prominent periodic pattern in the input signal’ s waveform.

The TSM algorithms can be classified into three categories, time-domain based, spectral-domain based and hybrid approaches. Since there’ s currently no machine-learning approach in this field, we take the notion from previous work and design a neural-network-based algorithm aiming to produce more realistic audio.

2 Related work

When we try to change the length of a piece of audio, the most intuitive way is to resample it at a different sampling rate. Unfortunately, this method causes the frequency of the audio to change and the pitch to change as well. It’ s a tremendous research challenge to maintain the pitch while adjusting the length, vice versa. Once one problem is solved, another can be achieved by

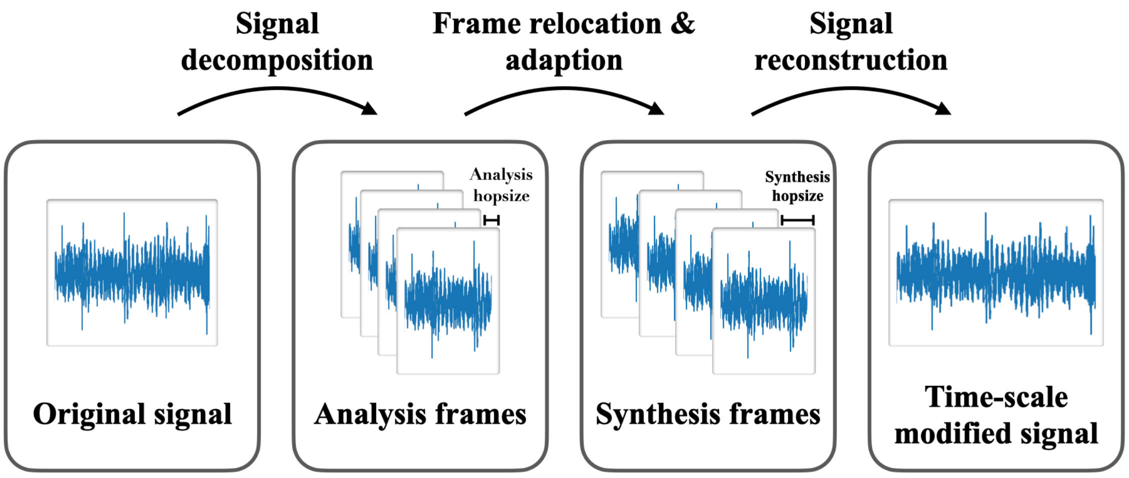

Figure 1: Generic processing pipeline of time-scale modification (TSM) procedures.

resampling. The second is that as audio consists of a complex combination of content, it can be broadly divided into harmonic, percussion and transient. The fiddle, for example, produces mainly harmonics, and the algorithm should focus on maintaining its pitch and tone. And the castanets are mainly percussive and do not have much tone character, so the focus should be on a short, crisp sound. The transient noise should maintain its short and simple state, however, few TSM algorithms can address all of these needs [3].

Most TSM algorithms cut the audio into small segments of fixed length, called frames, in order to minimise boundary breakage after processing, we overlap adjacent frames and these frames are rearranged and then synthesised into a new audio. As shown in Figure 1. We can divide the processing domain into time frequency domain method and hybrid method.

2.1 Time domain method

The time domain method takes frames of a fixed length on the time axis of the audio raw data and will overlap with adjacent frames to ensure continuity. These frames will be duplicated or discarded then repositioned. The common problem with this method is how to make the composite frames continuous, aligned and free from artificial noise. Several optimization algo-

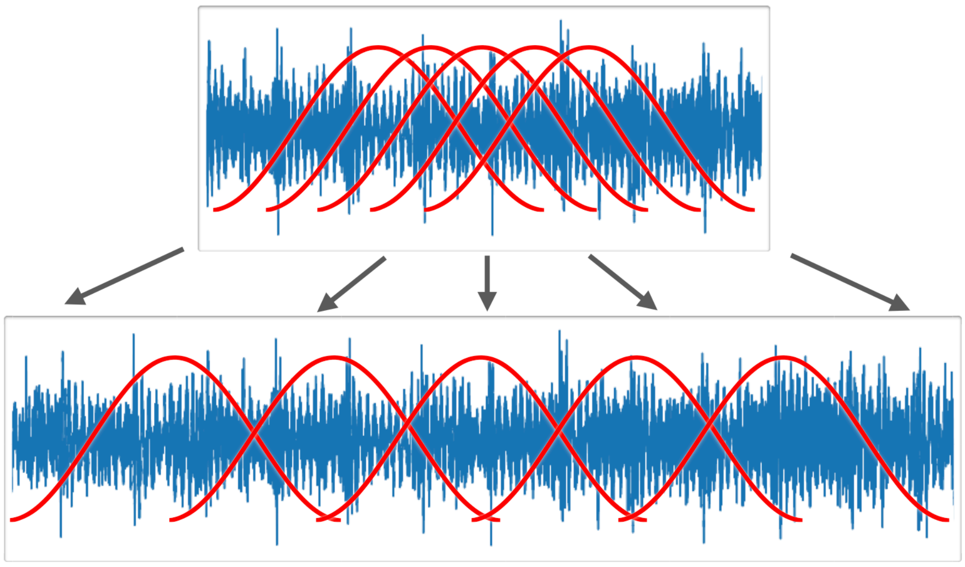

Figure 2: An illustration for overlap-add (OLA) algorithm.

rithms have been proposed, and this project reviews several important ones. The overlap-add (OLA) method was first proposed and is described in detail below, followed by synchronized overlap-add with fixed synthesis (SOLAFS) [8], time-domain pitch synchronized overlap-add (TD-PSOLA) [12] and waveform similarity overlap-add (WSOLA) [17], in order to improve the shortcomings of the previous algorithm.

2.1.1 OLA

The OLA algorithm can be thought of as mapping each frame from the original track to the synthesis track. The distance between frames in the original track is called the analysis hopsize and the synthesis hopsize in the synthesis track. When the synthesis hopsize is greater than the analysis hopsize, it is possible to achieve an elongated audio signal with no change in pitch, and vice versa. In order not to double the intensity of the overlapping parts, a window function is usually added to each frame. The Hann function is a common function used in practice, as shown in Figure 2. This is a good optimization for the percussion.

Although the audio can be maintained at roughly the same as its original strength. With significant periodicities, an unconstrained ratio of analysis to synthesis jump distance can cause

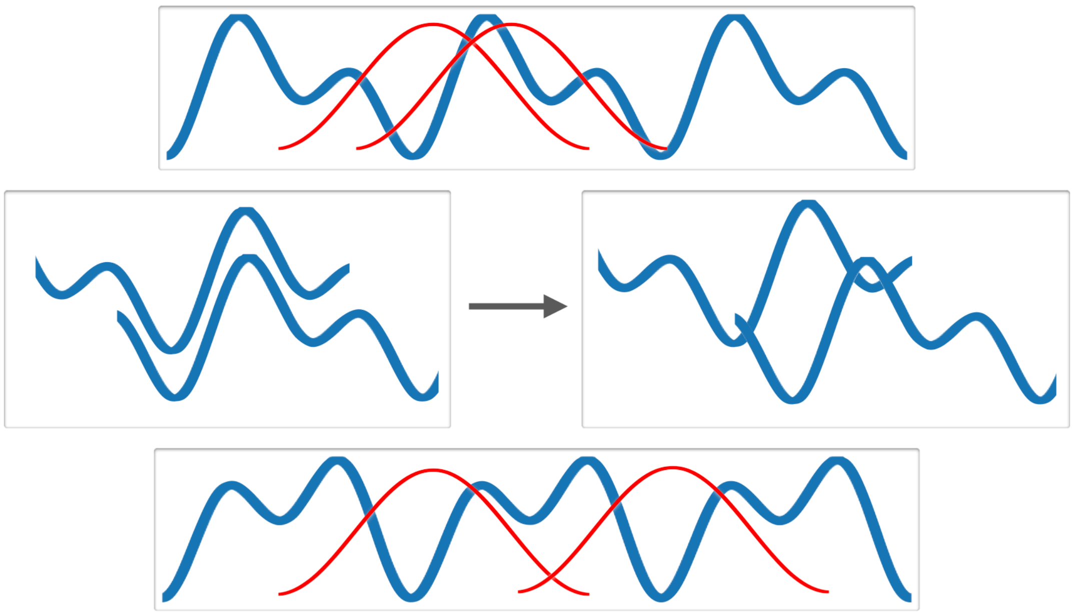

Figure 3: An illustration for harmonic alignment problem.

a discrepancy with the original waveform. This is known as the harmonic alignment problem, and is illustrated in Figure 3.

2.1.2 WSOLA

SOLA [13] adds alignment constraints so that the harmonics can maintain their character after processing. The harmonics can maintain their characteristics after processing. The proposed SOLAFS [8] allows us to define this ratio more freely. SOLAFS allows us to define this ratio more freely, but the accompanying large computational effort makes it difficult to WSOLA [17] significantly reduces the computational complexity by comparing the similarity of waveforms. The actual method is to find the frame that best matches the waveform of the previous aligned frame within a pre-defined error range. as shown in Figure 4. This frame is used as the reference point to iteratively find the next frame.

This approach significantly reduces the problem of OLA misalignment and makes harmonics sound more natural sounding. However, there are problems with percussive and transient noise that can be duplicated or disappear (transient doubling, transient skipping) . If the first and second frames of a sequence contain the same transient sound may cause problems with the

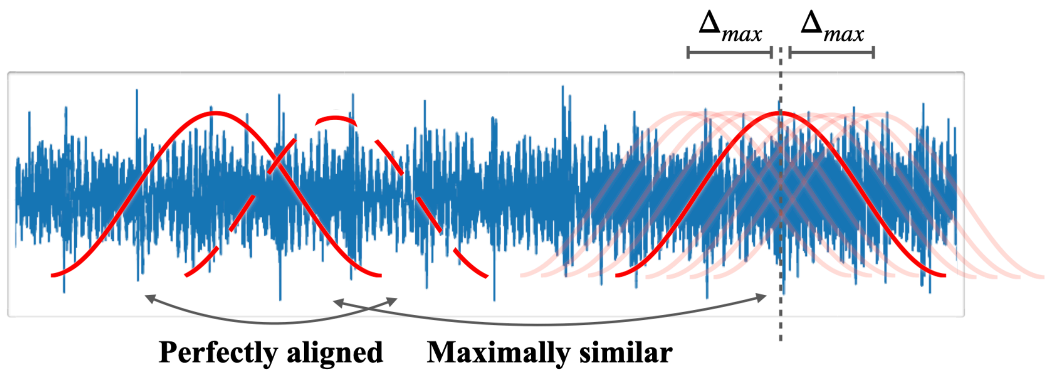

Figure 4: An illustration for waveform similarity overlap-add (WSOLA) algorithm.

alignment of the two frames, which often results in duplication in slowed-down processing and loss in fast-forward processing, as shown in Figure 5. In addition, this waveform is more specific to the most significant harmonics. In the case of orchestral music, only the most dominant instrument sounds may be retained.

2.2 Frequency domain method

In this category, the Fourier transform will be performed on each frame, i.e., short-time Fourier transform (STFT) [6] over the entire audio segment, to obtain a frequency spectrogram representing the frequency distribution at each time unit. The STFT also generates a phase spectrogram, which shows the phases of each frequency level at each time unit. We can reorganize the frame in the frequency spectrogram to get different playback speeds under the constraint of the same frequency distribution. However, merely reorganizing the frequency spectrogram leads to the discontinuity between each frame in the audio signal. We can use phase vocoder to solve this problem.The phase vocoder also gets a lot of attention these days, but it is beyond discussion in this work. A TSM based on the phase vocoder (PV-TSM) has been proposed by J. Laroche et all [9]

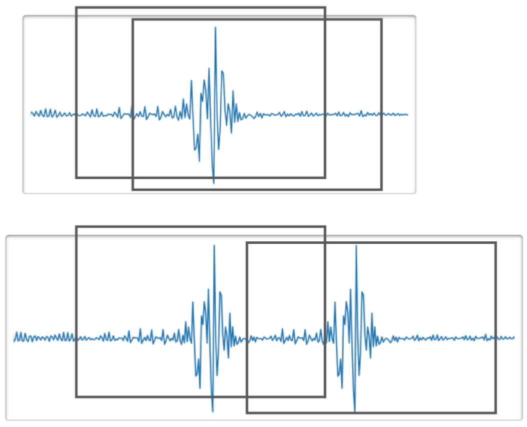

Figure 5: An illustration for transient doubling.

2.2.1 PV-TSM

When the distance between two frames gets modified, the gap of phases between two frames is introduced. This algorithm uses phase vocoder to correct the phases, mitigating the phase jump artifacts. Finally, the inverse STFT is applied on the modified frames to get the time-domain audio signals as shown in Figure 6.

This approach mitigates the loss of secondary sources since the sources are processed separately in the spectrogram. However, it also generates another kind of artifact, smeared transients. The phase relationships of sinusoidal components within one frame, is usually destroyed in the phase propagation process. The loss of vertical phase coherence affects the time localization of events such as transients. As a result, transients are often smeared in signals modified with PV-TSM as shown in Figure 7.

2.3 Hybrid method

From the algorithm above we can tell, different algorithms suit different kinds of audio signals. For example, despite the fact that the simple OLA has poor performance on harmonic signal, it

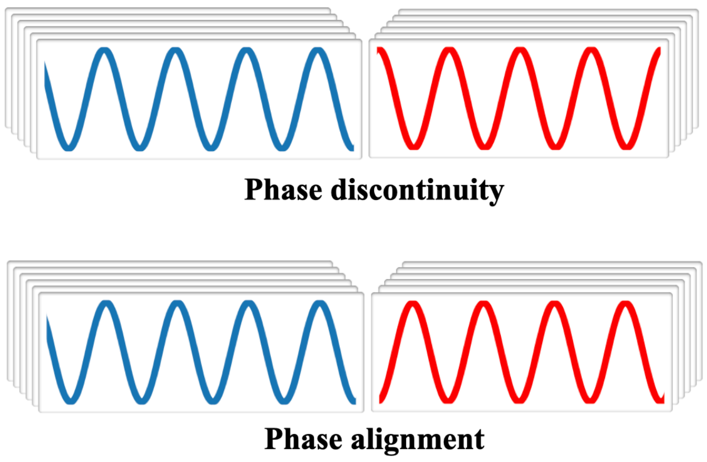

Figure 6: An illustration for TSM based on the phase vocoder (PV-TSM).

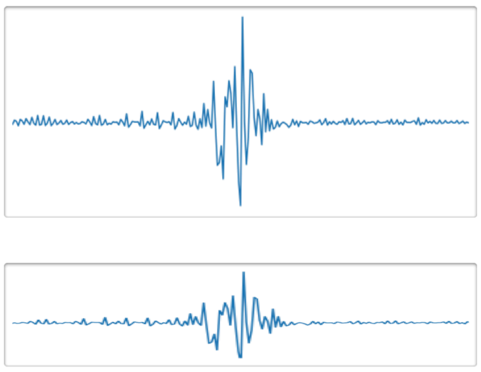

Figure 7: An illustration for smeared transient.

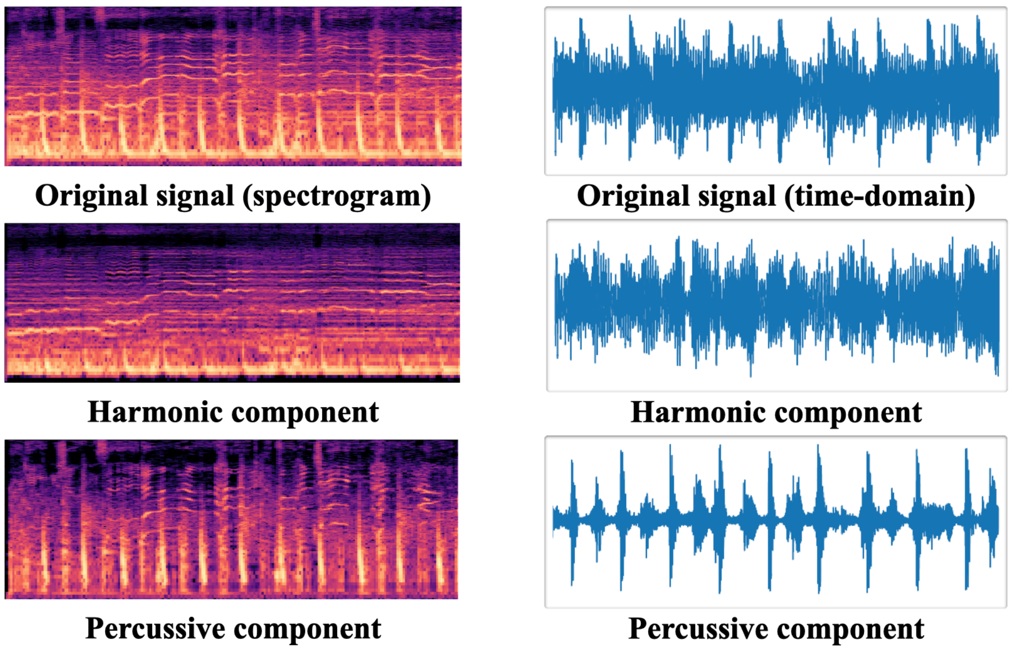

Figure 8: An illustration for harmonic/percussive source separation (HPSS).

has better performance on percussion than PV-TSM. As a result, the most intuitive improvement is to separate the different kinds of components in an audio signal and process the harmonics and percussion separately [4]. Harmonic/percussive source separation (HPSS) is a topic discussing how to effectively separate the different sources from an audio signal.

2.3.1 HPSS via median filtering

The harmonic signals typically contain periodical waveforms while percussive signals do not [5]. Interestingly, in the spectrogram, the horizontal frequency components represent harmonics, and the vertical components represent the percussion as shown in Figure 8. We can use this characteristic and median filter to separate these two groups of components in the spectrogram. Then we can apply the suitable TSM algorithms on these two to preserve both harmonic and percussive features.

2.3.2 HPSS via autoencoder

Learning-based HPSS methods have also been proposed [10]. Using the autoencoder neural network and the skip connections [7] via recurrent neural networks (RNN, GRU [2]), we can effectively encode an audio segment into a relatively low-dimensional latent vector. Then the model can learn the transformation of separating different components from the dataset in a supervised fashion.

3 Methodology

During this semester, we have been discussing and thinking about what models to use to train the data, and as this is an area of research that has not been done before, we have been carefully looking at the relevant data and figuring it out step by step. The instance normalization (IN) has been found effective on removing instance-specific-features while the adaptive instance normalization (AdaIN) can be used to inject the instance-specific-features into the final output. In addition, in order to improve the accuracy on the tempo estimation and enable end-to-end training, we decided to use neural networks to enhance the estimation of tempo, which is a one-step method to estimate tempo, unlike the conventional methods. Estimation methods often combine signal processing with heuristics. for example, used bandpass filters, followed by parallel comb filters, followed by peak picking.

We designed a model, as shown in Figure 9, to separate the content and the tempo component with CNN-based tempo estimation (TE) algorithms [14]. We expect the design can properly isolate the tempo information from others and thus achieve the time-scale modification (TSM) on the input audio. By which, we can now perform an end-to-end training and fine tune the TE model, producing more meaningful latent representation for the tempo information. See Figure 1 for the model architecture.The instance normalization (IN) has been found effective on removing instance-specific-features while the adaptive instance normalization (AdaIN) can be used to inject the instance-specific-features into the final output.

3.1 Feature disentangle

3.1.1 Instance normalization

The instance normalization (IN) [16] is said to be capable of removing instance-specific-features, e.g., speaker information, image style, and hopefully tempo information [1][15]. The IN without affine transformation is given by

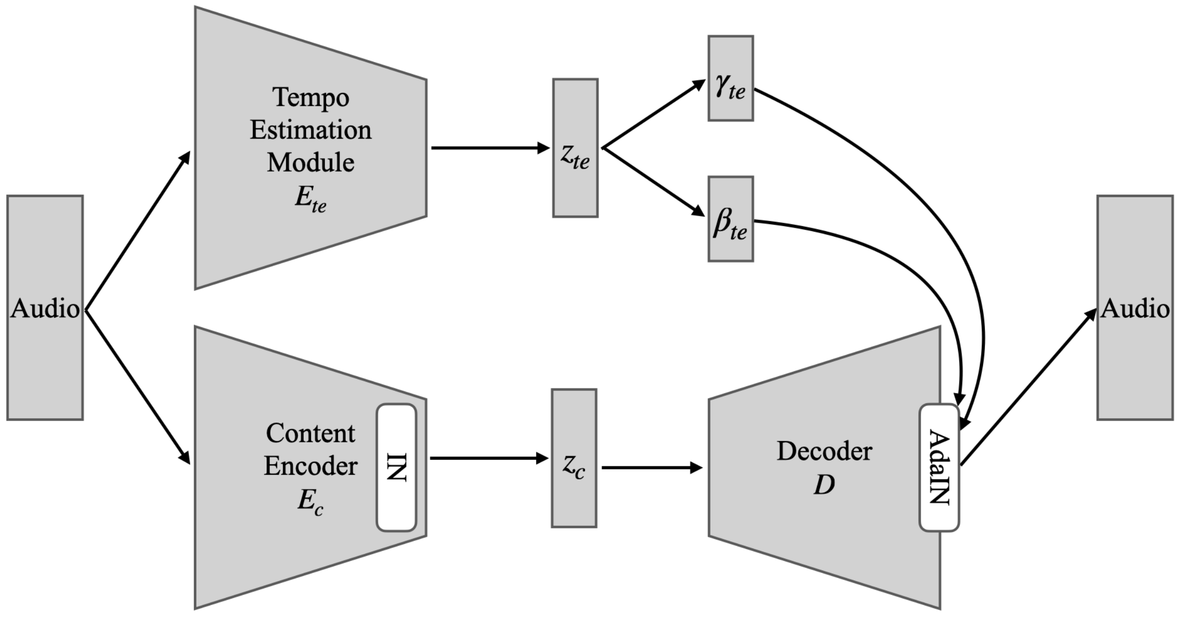

Figure 9: Model overview. E_c is the content encoder; E_{te} is the tempo estimation module and D is the decoder. IN is instance normalization layer without affine transformation. AdaIN represents adaptive instance normalization layer.

$$y = \frac{x - E[x]}{\sqrt{Var[x] + \epsilon}}$$

where the mean and standard-deviation are calculated per-channel separately for each sample in a mini-batch. That is to say, the mean and variance for each channel in each sample are equal to 0 and 1 respectively. From the previous CNN research, we know that each convolution kernel represents certain kind of feature. In image, it could be texture, edge, style, or semantic meanings in the deep layers. In audio, it may be tones, tempo, speaker characteristic, etc. If we make each channel have the same mean and variance, they will lose the characteristic and be composed of mainly general informations that shared across the samples. Which hopefully, in our case, is everything except the tempo information.

3.1.2 Adaptive instance normalization

To further enhance the influence of the output from TE network. We use the same techniques of the enhancement on the speaker information in a feature-disentangle voice conversion model [1]. Chou et al. provide the speaker information to the decoder by adaptive instance normalization (adaIN) layer. In adaIN layer, the output from the decoder first got normalized by IN, then a channel-wise linear transformation, whose coefficients comes from the output of the TE network, will apply follow through. The formula is given by

$$y = \gamma_{te} \frac{D(x) - E[D(x)]}{\sqrt{Var[D(x)] + \epsilon}} + \beta_{te}$$

where γ_{te} and β_{te} are propagated from the TE latent vector z_{te} and the mean and standard-deviation are calculated per-channel separately for each sample in a mini-batch.

3.1.3 Evaluation of disentanglement

Chou et al. [1] perform some evaluation on their content encoder to see if the instance-specific-features are actually ripped out from it. They trained another fully connected network to classify speaker identity given the latent representation encoded by the content encoder. Three different configurations were placed:

1. Content encoder with IN

E_c with IN	E_c without IN	E_c without IN and E_s with IN
0.375	0.658	0.746

Table 1: The accuracy for speaker identity prediction on content representation from Chou et al.

2. Content encoder without IN
3. Content encoder without IN while speaker encoder with IN

The results are shown in Table 1. The classification accuracy is apparently lower when IN is applied to the content encoder. However, the accuracy was not as high as expected for the content encoder without IN. This was probably because by the fact that the speaker encoder is able to control the channel statistics of decoder by adaIN, the whole model tends to learn speaker information from the speaker encoder rather than from the content encoder. The third setting is meant to verify this assumption, since we now apply IN on the speaker encoder, it can no longer hold as many speaker information as before and the highest accuracy on the evaluation network for this setting supports the notion.

3.2 Tempo estimation

In the paper, Schreiber et al. describes a CNN-based approach that estimates the local tempo of a short musical piece or excerpt based on mel-scaled spectrograms in a single step [14]. without explicitly creating mid-level features like a beat activation function that need to be processed further by another, separate system component. And we can combine multiple local tempo into a global tempo by using averaging. It' s a general system that can adapt to all kinds of data. In our training, we will use extended ballroom datasets, referred to as Eballs [11]. It contains 3,826 tracks with 30s length each.

3.2.1 Architecture

Although tempo estimation appears to be a regression problem, in the paper, the author approaches it as a classification problem. So instead of attempting to estimate a BPM value as decimal number, the model chooses one of 256 tempo classes, covering the integer tempo values from 30 to 285 BPM. First, process the input with three convolutional layers with 16 (1

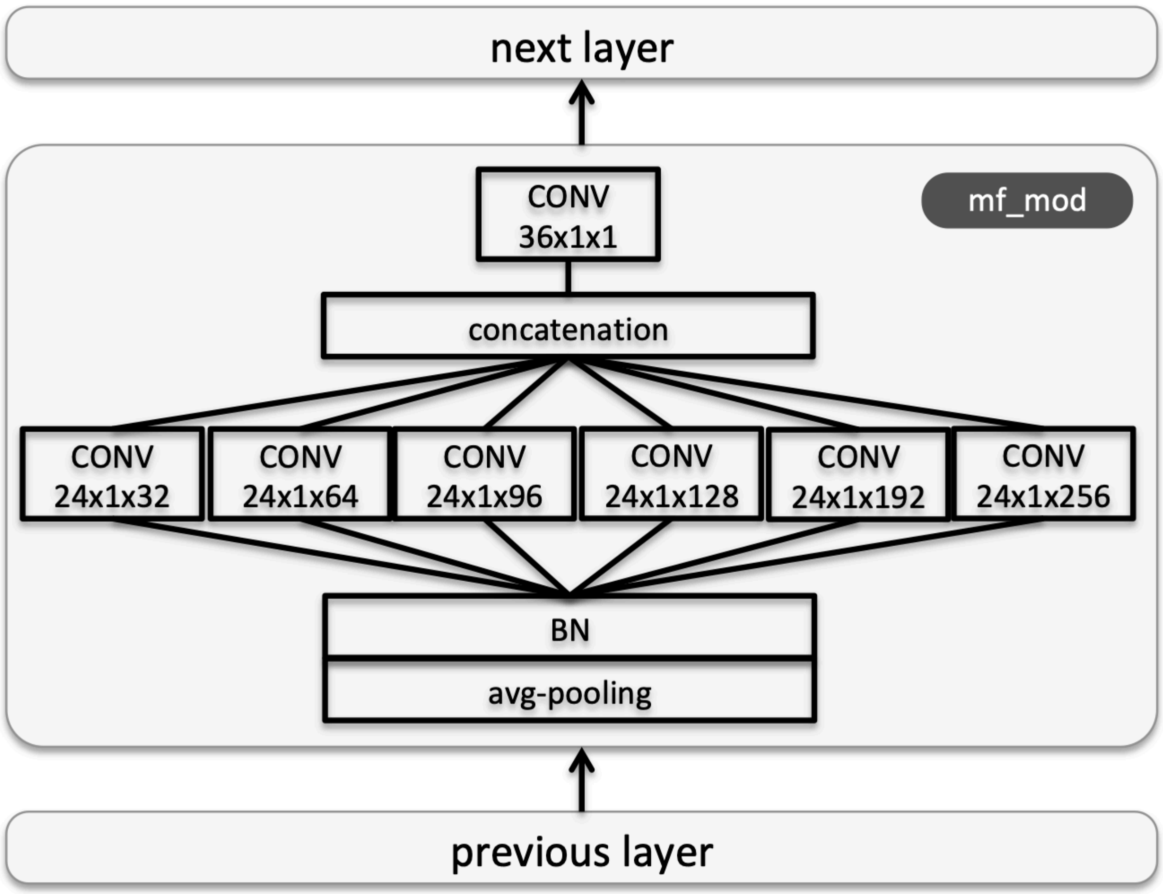

Figure 10: Each multi-filter module *mf_mod* consists of a pooling layer, batch normalization, six different convolutional layers, a concatenation layer and a bottleneck layer. The activation function for all convolutional layers is ELU.

× 5) filters for each. All filters are oriented along the time axis using padding and a stride of 1. These three layers are followed by four almost identical multi-filter modules (*mf_mod*, Figure 10, figure credit to Schreiber et al.), each consisting of an average pooling layer (m × 1), parallel convolutional layers with different filter lengths ranging from (1 × 32) to (1 × 256), a concatenation layer and a (1 × 1) bottleneck layer for dimensionality reduction. And add two fully connected layers (64 units each) followed by an output layer with 256 units to classify the features delivered by the convolutional layers. The first fully connected layer is additionally preceded by a dropout layer with p = 0.5 to counter overfitting. As loss function use categorical cross-entropy. Overall, the network has 2,921,042 trainable parameters.

References

[1] chieh Chou, J., and Lee, H.-Y. One-Shot Voice Conversion by Separating Speaker and Content Representations with Instance Normalization. In *Proc. Interspeech 2019* (2019), pp. 664–668.

[2] Cho, K., van Merrienboer, B., Gülçehre, Ç., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR abs/1406.1078* (2014).

[3] Driedger, J., and Müller, M. A review of time-scale modification of music signals. *Applied Sciences 6*, 2 (2016).

[4] Driedger, J., Müller, M., and Ewert, S. Improving time-scale modification of music signals using harmonic-percussive separation. *IEEE Signal Processing Letters 21*, 1 (2014), 105–109.

[5] Fitzgerald, D. Harmonic/percussive separation using median filtering. In *Proceedings of the International Conference on Digital Audio Effects (DAFx)* (2010), vol. 13.

[6] Gabor, D. Theory of communication. part 1: The analysis of information. *Journal of the Institution of Electrical Engineers - Part III: Radio and Communication Engineering 93* (November 1946), 429–441(12).

[7] He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2016).

[8] Hejna, D., and Musicus, B. R. The solafs time-scale modification algorithm. *Bolt, Beranek and Newman (BBN) Technical Report* (1991).

[9] Laroche, J., and Dolson, M. Improved phase vocoder time-scale modification of audio. *IEEE Transactions on Speech and Audio Processing 7*, 3 (1999), 323–332.

[10] Liu, J.-Y., and Yang, Y.-H. Denoising auto-encoder with recurrent skip connections and residual regression for music source separation. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)* (2018), pp. 773–778.

[11] Marchand, U., and Peeters, G. The Extended Ballroom Dataset, Aug. 2016. Late-Breaking Demo Session of the 17th International Society for Music Information Retrieval Conf.. 2016.

[12] Moulines, E., and Charpentier, F. Pitch-synchronous waveform processing techniques for text-to-speech synthesis using diphones. *Speech Communication 9*, 5 (1990), 453–467. Neuropeech ’89.

[13] Roucos, S., and Wilgus, A. High quality time-scale modification for speech. In *ICASSP ’85. IEEE International Conference on Acoustics, Speech, and Signal Processing* (1985), vol. 10, pp. 493–496.

[14] Schreiber, H., and Müller, M. A single-step approach to musical tempo estimation using a convolutional neural network.

[15] Ulyanov, D., Vedaldi, A., and Lempitsky, V. Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis, 2017.

[16] Ulyanov, D., Vedaldi, A., and Lempitsky, V. Instance normalization: The missing ingredient for fast stylization, 2017.

[17] Verhelst, W., and Roelands, M. An overlap-add technique based on waveform similarity (wsola) for high quality time-scale modification of speech. In *1993 IEEE International Conference on Acoustics, Speech, and Signal Processing* (1993), vol. 2, pp. 554–557 vol.2.