LEARNING THE SPECTROGRAM TEMPORAL RESOLUTION FOR AUDIO CLASSIFICATION

Anonymous authors
Paper under double-blind review

ABSTRACT

The audio spectrogram is a time-frequency representation that has been widely used for audio classification. The temporal resolution of a spectrogram is controlled by a parameter called the hop size. Previous works generally assume the hop size should be a constant value of around ten milliseconds. However, a fixed hop size (i.e., resolution) is not always optimal for different types of sound. This paper proposes a novel method, DiffRes, that is the first to enable differentiable temporal resolution learning to improve the performance of audio classification models. Given a spectrogram calculated with a fixed hop size, DiffRes merges non-essential time frames while preserving important frames. DiffRes acts as a "drop-in" module between regular spectrogram calculation and a classifier, and can be end-to-end optimized. We evaluate DiffRes on the mel-spectrogram, followed by state-of-the-art classifier backbones, and apply it to six different subtasks. By learning the optimal temporal resolution, the DiffRes-based method can achieve the same or better classification accuracy with at least 25% fewer temporal dimensions on the feature level, which alleviates the computational cost at the same time. Starting from a high-temporal-resolution spectrogram (e.g., one-millisecond hop size), we show that DiffRes can improve classification accuracy with the same computational complexity. Our code is publicly available.

1 INTRODUCTION

Audio classification refers to a series of tasks that assign labels to an audio clip. Those tasks include audio tagging (Kong et al., 2020), speech keyword classfication (Kim et al., 2021), and music genres classification (Castellon et al., 2021). The input to an audio classification system is usually a one-dimensional audio waveform, which can be represented by discrete samples. Although there are methods for directly modeling waveforms (Kong et al., 2020; Luo & Mesgarani, 2018; Lee et al., 2017), the majority of studies on audio classification convert the waveform into a spectrogram as the input feature (Gong et al., 2021b;a). The basis of the spectrogram is the Fourier transform (Champeney & Champeney, 1987), which is applied in short waveform trunks divided by a windowing function, resulting in a two-dimensional time-freqency (TF) representation. According to Gabor's uncertainty principle (Gabor, 1946), there is always a trade-off between time and frequency resolutions. To achieve desired resolution on the temporal dimension, a common practice (Kong et al., 2021; Liu et al., 2022) is applying a fixed hop size between windows to capture the dynamic between adjacent frames. In this case, the temporal dimension size is inversely proportional to the hop size, and the spectrogram has a fixed temporal resolution.

Using a fixed temporal resolution (TR) is not necessarily optimal for an audio classification model. Intuitively, TR should depend on the temporal pattern: fast-changing signals suppose to have high TRs, and relatively steady signals or blank signals may not need the same high TR for the best accuracy. For example, as we increase the TR (see Figure 1), more details appear in the spectrogram of *Alarm Clock* while the pattern of *Siren* stays mostly the same, which indicates the finer details in high-TR *Siren* may not essentially contribute to the accuracy. With similar motivations, (Ferraro et al., 2021) perform experiments on music tagging with spectrograms with reduced TR, and observes a similar performance can be maintained. In addition, our pilot study (see Figure 8a) shows the optimal TR is not the same for a different type of sound. The pilot study shows that improvement on different types of sound is not consistent when increasing TR, in which some of them even get worse with a higher temporal resolution. This evidence indicates the redundancy in the fix-resolution

audio spectrogram and the potential of learning suitable temporal resolutions. There are plenty of studies on learning a suitable frequency resolution with a similar spirit (Stevens et al., 1937; Sainath et al., 2013; Ravanelli & Bengio, 2018b; Zeghidour et al., 2021), but the work on learning TR is still rare.

Besides, the potential of high temporal resolution spectrogram, e.g., one milliseconds (ms) hop size, is still unclear. Previous studies (Kong et al., 2020) show classification performance can be steadily improved with the growth of temporal resolution. Some popular choices of hop size including 10 (Böck et al., 2012; Kong et al., 2020; Gong et al., 2021a) and 12.5 (Shen et al., 2018; Rybakov et al., 2022) milliseconds. But one remaining question is: Can even finer temporal resolution improve the performance? We conduct a pilot study for this question on the speech keyword classification task with hop sizes smaller than 10 ms (see Figure 8b). We noticed accuracy can still be improved with smaller hop size, at a cost of computational complexity. This indicates there is still useful information in the higher temporal resolution that has not yet received enough attention.

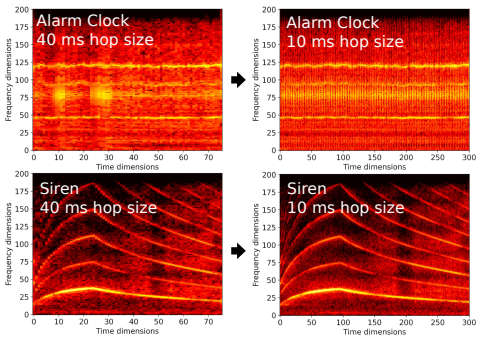


Figure 1: The spectrogram of *Alarm Clock* and *Siren* sound with 40 ms and 10 ms hop sizes. All with a 25 ms window size. The pattern of *Siren*, which is relatively stable, does not change significantly using a smaller hop size (i.e., larger temporal resolution), while *Alarm Clock* is the opposite.

In this work, we are the first to demonstrate learning TR (or temporal resolution) on the audio spectrogram and show efficiency and accuracy improvements over the fix-resolution spectrogram. We propose a lightweight algorithm, DiffRes, that makes spectrogram TR differentiable during model optimization. The optimal TR is learned at a frame level on the audio data. DiffRes can be used as a "drop-in" module after spectrogram calculation and optimized end-to-end with the downstream task. The output of DiffRes is a time-frequency representation with varying TR, which is achieved by adaptively merging the time steps of a fix-resolution spectrogram. We perform experiments on six different subtasks, including the largest and most challenging benchmark AudioSet (Gemmeke et al., 2017). DiffRes shows clear improvements over the fix-resolution mel-spectrogram baseline and other learnable front-end (Zeghidour et al., 2021; Ravanelli & Bengio, 2018b; Zeghidour et al., 2018). Compared with fix-resolution spectrogram, we show that using DiffRes can achieve at least 25% temporal dimension reduction with the same or better audio classification accuracy. On high-resolution spectrogram, we also show that DiffRes can effectively improve classifier performance without increasing the feature temporal dimensions. Our code is publicly available[1].

2 Learning Temporal Resolution with DiffRes

We first provide an overview in Section 2.1. We focus on applying DiffRes on the mel-spectrogram, which is one of the most popular feature in state-of-the-art systems (Chen et al., 2022; Gong et al., 2022; Verbitskiy et al., 2022; Koutini et al., 2021). We introduce the detailed formulation of DiffRes and the end-to-end optimization in Section 2.2.1, 2.2.2, and 2.3.

2.1 Overview

Let $\boldsymbol{x} \in \mathbb{R}^L$ denote a one-dimensional audio waveform. An audio classification system can be divided into a feature extraction stage and a classification stage. In the feature extraction stage, the audio waveform will be processed by a function

$$\mathcal{S}(l, h) : \mathbb{R}^L \to \mathbb{R}^{f \times t}, \tag{1}$$

which maps the waveform into a two-dimensional time-frequency representation $\boldsymbol{X}$, e.g., mel-spectrogram. Here, t and f stand for the time and frequency dimensions of the extracted feature. We also refer to the feature along the temporal dimensions as frames. l and h denote window length and

[1] anonymouslink

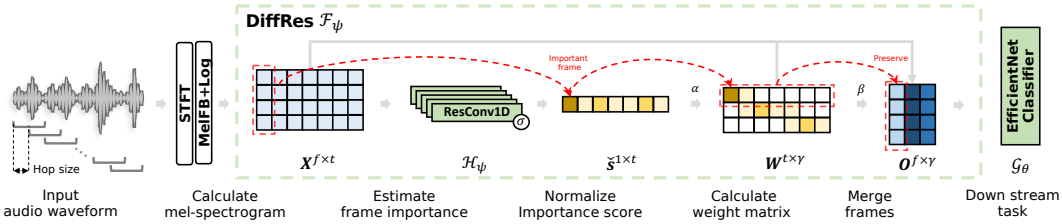


Figure 2: Breakdown of the DiffRes algorithm. Green blocks contain learnable parameters. DiffRes is a "drop-in" module between spectrogram calculation and the downstream task. DiffRes can be end-to-end optimized.

hop size, respectively. Usually $t \propto h^{-1}$. We define the temporal resolution by a value called frame per second (FPS), which denotes the number of frames in one second. In the classification stage, $\boldsymbol{X}$ will be processed by a model $\mathcal{G}_\theta$ parameterized by θ. The output of $\mathcal{G}_\theta$ is the label predictions. Given the paired training data $(\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{D}$, the optimization of the classification system can be formulated as

$$\arg\min_{\theta} \mathbf{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\mathbb{D}}\, \mathcal{L}(\mathcal{G}_\theta(\mathcal{S}(\boldsymbol{x})), \boldsymbol{y}), \tag{2}$$

where $\mathcal{L}$ is the loss function such as cross entropy (De Boer et al., 2005). Figure 2 and Algorithm 2 show an overview of performing classification with DiffRes. DiffRes is a "drop-in" module between F and $\mathcal{G}_\theta$ focusing on learning the optimal temporal resolution with a learnable function $\mathcal{F}_\psi$:

$$\mathcal{F}_\psi : \mathbb{R}^{f\times t} \rightarrow \mathbb{R}^{f\times\gamma}, \tag{3}$$

where γ is a constant number, denoting the output time dimensions of DiffRes. DiffRes formulates Equation 3 with two steps: i) estimating the importance of each time frame with a learnable model $\mathcal{H}_\psi$: $\boldsymbol{X} \rightarrow \check{\boldsymbol{s}}$, where $\check{\boldsymbol{s}}$ is a $1 \times t$ shape row vector; and ii) merging frames based on a proposed monotonic frame combination algorithm, in which monotonic (Kim et al., 2020) means the combination is performed along a single direction on the temporal dimension. We introduce the details of these two steps in Section 2.2.1 and Section 2.2.2. We define the compression rate of DiffRes (δ) by $\delta = (t - \gamma)/t$. Usually $\delta \leq 1$ and $\gamma \leq t$ because the temporal resolution of the $\mathcal{F}_\psi$ output is either smaller or equal than $\boldsymbol{X}$. Given the same t, a larger δ means fewer temporal dimensions in the output of DiffRes, and less computation needed for $\mathcal{G}_\theta$ at the same time. Similar to Equation 2, $\mathcal{F}_\psi$ can be end-to-end optimized by

$$\arg\min_{\theta,\psi} \mathbf{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\mathbb{D}}\, \mathcal{L}(\mathcal{G}_\theta(\mathcal{F}_\psi(\mathcal{S}(\boldsymbol{x}))), \boldsymbol{y}). \tag{4}$$

2.2 Differentiable Temporal Resolution Modeling

Figure 2 and Algorithm 3 illustrate how DiffRes performs adaptive temporal resolution learning. We introduce frame importance estimation in Section 2.2.1. The detailed design of weight matrix construction function α and frame merging function β are introduced in Section 2.2.2.

2.2.1 Frame Importance Estimation

We design the frame importance estimation module $\mathcal{H}_\psi$ for deciding how much each frame needs to be kept in the output. $\mathcal{H}_\psi$ will output a row vector with shape $t \times 1$, indicating the importance score of each time frame. This process can be denoted as

$$\boldsymbol{s} = \sigma(\mathcal{H}_\psi(\boldsymbol{X})), \tag{5}$$

where $\boldsymbol{s}$ is the importance score, and σ is the sigmoid function (Han & Moraga, 1995). A higher value in $\boldsymbol{s}$ indicates the corresponding frame is important for classification. We implement $\mathcal{H}_\psi$ with a stack of one-dimensional convolutional neural networks (CNN) (Fukushima & Miyake, 1982; LeCun et al., 1989). Specifically, the network is a stack of five one-dimensional convolutional blocks (ResConv1D) with exponentially decreasing channel numbers (see Table 6). As shown in Figure 3, each ResConv1D contains

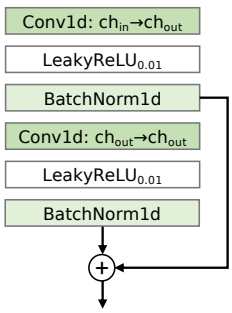


Figure 3: ResConv1D

two sets of one-dimensional CNN, batch normalization (Ioffe & Szegedy, 2015), and Leaky rectified linear unit (Xu et al., 2015) activation function with a residual connection (He et al., 2016). Each layer has padding to ensure the temporal dimension does not change. In the next frame merging step (Section 2.2.2), elements in $\boldsymbol{s}$ will represent how much each input frame contributing to an output frame. Therefore, we perform normalization on $\boldsymbol{s}$ to ensure $\check{\boldsymbol{s}} \in [0,1]^{t\times 1}$ and $\sum_{k=1}^{t} \check{\boldsymbol{s}}_k \leq \gamma$. The normalization process can be denoted as

$$\check{\boldsymbol{s}} = \frac{\boldsymbol{s}'}{(1 + (\max(\boldsymbol{s}') - 1)^{+})}, \quad \boldsymbol{s}' = \frac{\boldsymbol{s}}{\sum_{i=1}^{t} \boldsymbol{s}_i}\gamma, \tag{6}$$

where s' is an intermediate variable that may contain elements greater than one, $+$ denotes selecting the positive elements. To quantify how active $\mathcal{H}_\psi$ is trying to distinguish between important and less important frames, we also design a measurement, DiffRes activeness ρ, which is calculated by the standard derivation of the non-empty frames. The DiffRes activeness is given by

$$\rho = \frac{1}{\delta}\sqrt{\frac{\sum_{i\in\mathbb{S}}(\boldsymbol{s}_i - \mu)^2}{|\mathbb{S}|}}, \quad \mu = \overline{\boldsymbol{s}_i}, \quad i \in \mathbb{S} = \{i \mid \overline{\exp(\boldsymbol{X}_{:,i})} > \epsilon\}, \tag{7}$$

where $\mathbb{S}$ is a set of non-empty frame indexes, μ is the average importance score of the non-empty frames, and δ is the compression rate. $|\mathbb{S}|$ denotes the size of set $\mathbb{S}$. We use δ to unify the ρ value with different γ in Equation 6 for easier comparisons. ρ can be used as an indicator of how DiffRes behaves during training. A higher ρ indicates the model is more active at learning the frame importance. If $\mathcal{H}_\psi$ learns nothing and always outputs constant values, ρ will be zero. We will discuss the learning process of DiffRes with ρ in Section 3.3.

2.2.2 Monotonic Frame Merging

We perform monotonic frame merging based on $\check{\boldsymbol{s}}$ and $\boldsymbol{X}$ to calculate a representation with adaptive temporal resolution. The word *monotonic* means the output feature of the merging algorithm should preserve the original temporal order. Here monotonicity is a strong inductive bias (Mitchell, 1980; Gordon & Desjardins, 1995) in DiffRes algorithm considering the audio signal does not transmit in reverse time. The monotonicity makes the DiffRes feature more interpretable and performs better on the audio signal. Generally, the monotonic frame merging algorithm can be denoted as

$$\boldsymbol{W} = \alpha(\check{\boldsymbol{s}}), \ \boldsymbol{O} = \beta(\boldsymbol{X}, \boldsymbol{W}) \tag{8}$$

where $\boldsymbol{W}$ is a weight matrix with shape $\gamma \times t$. $\boldsymbol{W}_{i,j}$ denote the contribution of the j-th input frame $\boldsymbol{X}_{:,j}$ to the i-th output frame $\boldsymbol{O}_{:,i}$. We will introduce the realization of weight matrix construction function $\alpha(\cdot)$ and merging function $\beta(\cdot)$ in the following sections.

Weight matrix construction We calculate the weight matrix $\boldsymbol{W}$ based on the importance estimation $\check{\boldsymbol{s}}$:

$$\boldsymbol{W}_{i,j} = \begin{cases} \check{\boldsymbol{s}}_j, & \text{if } i < \sum_{k=1}^{j} \check{\boldsymbol{s}}_k \leq i+1 \\ 0, & \text{otherwise} \end{cases}, \tag{9}$$

where we calculate the cumulative sum of $\check{\boldsymbol{s}}$ to decide which output frame each input frame will be merged into. The weight matrix $\boldsymbol{W}$ here can be used directly in Equation 8. However, we found the model training could be more stable if the weight matrix satisfies $\sum_{j=1}^{\gamma} \boldsymbol{W}_{j,:} = \boldsymbol{1}$, which means each output frame is assigned equal amount of weights on input frames. We further process $\boldsymbol{W}$ to meet this requirement with Algorithm 1. We also provide a vectorized version of Algorithm 1 that can be run efficiently on GPUs (see Appendix A.2).

Frame merging We perform frame merging based on the weight matrix $\boldsymbol{W}$. The i-th output frame is calculated with $\boldsymbol{X}$ and the i-th row of $\boldsymbol{W}$, given by

$$\boldsymbol{O}_{i,j} = \log(\mathcal{A}(\exp(\boldsymbol{X}_{j,:}) \odot (\boldsymbol{W}_{i,:}))), \tag{10}$$

where $\mathcal{A} : \mathbb{R}^{f\times t} \to \mathbb{R}^{1\times t}$ stands for the frame aggregation function such as averaging, $\boldsymbol{O}$ is the final output feature with shape $f \times \gamma$. Note that the value of the mel-spectrogram $\boldsymbol{X}$ is in the log scale. We convert $\boldsymbol{X}$ to the linear scale before merging because the weighted sum is more intuitive in the linear scale.

Algorithm 1: Inplace weight matrix update

Inputs : $\boldsymbol{W}$ in Equation 9. $\boldsymbol{W}$ satisfies $\sum_{i=1,j=1}^{\gamma,t} \boldsymbol{W}_{i,j} \leq \gamma$.

Output: The updated $\boldsymbol{W}$ that satisfies both $\sum_{i=1}^{\gamma} \boldsymbol{W}_{i,:} = \mathbf{1}$ and $\sum_{i=1,j=1}^{\gamma,t} \boldsymbol{W}_{i,j} \leq \gamma$.

$i \leftarrow 1;\ j \leftarrow 1;\ s \leftarrow 0$ ▷ i, j for indexing, s store the sum value in each row.
while $i < \gamma$ **and** $j < m$ **do**
  if $\boldsymbol{W}_{i,j} > 0$ **then**
    $s \leftarrow s + \boldsymbol{W}_{i,j}$; ▷ Add up the weights in row i.
    $j \leftarrow j + 1$; ▷ Move to the next column in row i.
  else
    $\boldsymbol{W}_{i,j} \leftarrow 1 - s$; ▷ Assign a weight to $\boldsymbol{W}_{i,j}$ to make $\sum \boldsymbol{W}_{i,:} = 1$
    $\boldsymbol{W}_{i+1,j} \leftarrow \boldsymbol{W}_{i+1,j} - \boldsymbol{W}_{i,j}$; ▷ The assigned weight is taken from $\boldsymbol{W}_{i+1,j}$.
    $i \leftarrow i + 1$; ▷ Move to row $i + 1$ in column j.
    $s \leftarrow 0$; ▷ Moved to the next row. Reset the row sum value.

Resolution encoding The final output $\boldsymbol{O}$ does not contain the resolution information at each time step, which is crucial information for the classifier. Since the temporal resolution can be represented with $\boldsymbol{W}$, we construct a resolution encoding with $\boldsymbol{W}$ in parallel with frame merging. Firstly, we construct a positional encoding matrix $\boldsymbol{E}$ with shape $f \times t$, using the similar method described in (Vaswani et al., 2017). Each column of $\boldsymbol{E}$ represents a positional encoding of a time step. Then we calculate the resolution encoding by $\mathcal{E} = \boldsymbol{W}\boldsymbol{E}$. We use $\mathcal{E}$ to inform the classifier backbone about the temporal resolution of $\boldsymbol{O}$.

2.3 Optimization

The loss function of the DiffRes-based audio classification system includes our proposed hint loss $\mathcal{L}_{hint}$ and the binary cross entropy (Shannon, 2001) loss $\mathcal{L}_{bce}$:

$$\mathcal{L} = \mathcal{L}_{hint} + \mathcal{L}_{bce}. \tag{11}$$

We propose to use hint loss to provide guidance for DiffRes on learning frame importance. Since we do not know the ground truth frame importance, we cannot directly optimize $\boldsymbol{s}$. And we introduce $\mathcal{L}_{hint}$ as an inductive bias (Mitchell, 1980) to the system based on the assumption that the empty frame should have low importance score. Specifically, we realize the hint loss by

$$\mathcal{L}_{hint} = -\frac{1}{\mid \mathbb{S} \mid} \sum_{i \in \mathbb{S}} (\frac{\boldsymbol{s}_i}{\delta} - \lambda)^{+}, \ \ \mathbb{S} = \{i \mid \overline{\exp(\boldsymbol{X}_{:,i})} < \epsilon\}, \tag{12}$$

where $\mathbb{S}$ is a set of time indexes that have low energy, and λ is a constant threshold. Given that the output of DiffRes has fewer dimensions, the DiffRes layer forms an information bottleneck (Tishby et al., 2000; Shwartz-Ziv & Tishby, 2017) that encourages DiffRes to assign a higher score to important frames. We analyze the information bottleneck of DiffRes in Section 3.3. The parameter λ is a threshold for the hint loss to take effect. This threshold can alleviate the modeling bias toward energy. For example, if $\lambda = 0$, the importance scores of empty frames are strongly regularized, the model will tend to predict low importance scores for lower energy frames, which may also contain useful information. $\mathcal{L}_{bce}$ is the standard loss function for classification, given by Equation 13, where $\hat{y}$ is the label prediction and N is the total number of classes.

$$\mathcal{L}_{bce} = \frac{1}{N} \sum_{i=1}^{N} (y_i \log(\hat{y}_i) + (1 - y_i \log(1 - \hat{y}_i))), \tag{13}$$

3 Experiments

We evaluate DiffRes on six different tasks and datasets (see details in Table 5), including audio tagging on AudioSet (Gemmeke et al., 2017) and FSD50K (Fonseca et al., 2021), environmental

Task	FPS	FixRes	AvgPool	NNPool	Proposed
AudioSet tagging 43.7 ± 0.1 (%)	25	38.6 ± 0.3	39.9 ± 0.2	40.1 ± 0.2	**41.7** ± 0.1
	50	41.8 ± 0.2	42.4 ± 0.1	42.7 ± 0.2	**43.6** ± 0.1
	75	42.7 ± 0.2	43.6 ± 0.0	43.5 ± 0.2	**44.2** ± 0.1†
FSD50K tagging 55.6 ± 0.3 (%)	25	48.9 ± 0.4	51.4 ± 0.3	49.2 ± 0.4	**56.9** ± 0.2†
	50	53.3 ± 0.4	54.5 ± 0.4	52.2 ± 0.8	**57.2** ± 0.2†
	75	54.8 ± 0.4	55.3 ± 0.3	54.4 ± 0.2	**57.1** ± 0.4†
Environmental sound 85.2 ± 0.5 (%)	25	74.6 ± 0.6	75.6 ± 0.3	72.4 ± 1.2	**82.9** ± 0.5
	50	82.4 ± 0.5	83.2 ± 0.3	77.3 ± 0.8	**85.5** ± 0.4†
	75	84.9 ± 0.3	85.2 ± 0.4†	81.8 ± 0.6	**86.8** ± 0.3†
Speech recognition 97.2 ± 0.1 (%)	25	93.5 ± 0.1	94.9 ± 0.4	**95.8** ± 0.3	95.0 ± 0.3
	50	96.1 ± 0.1	96.0 ± 0.2	96.0 ± 0.1	**96.7** ± 0.2
	75	96.8 ± 0.2	96.9 ± 0.1	97.0 ± 0.1	**97.2** ± 0.0†
Music instrument 79.9 ± 0.2 (%)	25	79.7 ± 0.2	78.3 ± 0.7	78.0 ± 0.5	**80.5** ± 0.2†
	50	79.9 ± 0.0†	79.5 ± 0.3	79.4 ± 0.3	**81.0** ± 0.5†
	75	79.8 ± 0.2	79.6 ± 0.3	79.7 ± 0.4	**80.8** ± 0.2†

Table 1: Comparison of different temporal dimension reduction methods. Number with † means better or comparable performance with the 100 FPS baseline.

sound classification on ESC50 (Piczak, 2015), limited-vocabulary speech recognition on Speech-Commands (Warden, 2018), and music instrument classification on NSynth (Engel et al., 2017). All the datasets are resampled at a sampling rate of 16 kHz. Following the evaluation protocol in previous works (Zeghidour et al., 2021; Riad et al., 2021; Kong et al., 2020; Gong et al., 2021b), we report the mean average precision (mAP) as the main evaluation metric on AudioSet and FSD50K, and report classification accuracy (ACC) on other datasets. In all experiments, we use the same architecture as in Gong et al. (2021b), which is an EfficientNet-B2 (Tan & Le, 2019) with four attention heads (13.6 M parameters). We reload the ImageNet pretrained weight for EfficientNet-B2 in a similar way to (Gong et al., 2021a;b). We apply random spec-augmentation (Park et al., 2019) with up to 37.5% and 50% masking on frequency and time dimension following (Gong et al., 2021b). And we apply mixup (Zhang et al., 2017) augmentation with 0.5 probability. All the experiments are repeated three times with different seeds to alleviate the randomness. And we report the standard derivation of the repeated trails along with the averaged result. We train the DiffRes layer with $\lambda = 0.5$ and $\epsilon = 1 \times 10^{-4}$. For the frame aggregation function $\mathcal{A}$ (see Equation 10), we use both the max and mean operations, whose outputs are concatenated with the resolution encoding $\mathcal{E}$ on the channel dimension, hence our classifier backbone has three input channels. The frame importance estimation module we used in this paper is a stack of five $\mathrm{ResConv1D}$ (see Table 6) with a total number of parameters of 82,371. We calculate the mel-spectrogram with a Hanning window, 25 ms window length, 10 ms hop size, and 128 mel-filterbanks by default. We list our detailed hyperparameters setting in Table 4. We also encourage readers to look through Figure 10 and Figure 11 for DiffRes examples on mel-spectrogram.

3.1 Adaptively compress the temporal dimension

The compression of mel-spectrogram temporal dimension can lead to a considerable speed up on training and inference (see Figure 4), thus it is a valuable topic for efficient classification (Huang & Leanos, 2018) and on-device application (Choi et al., 2022). This experiment aims to evaluate the effectiveness of DiffRes in compressing temporal dimensions and maintaining precision. We compare DiffRes with three temporal dimension reduction methods: i) FixRes reduces the temporal dimension by enlarging the hop size. So, FixRes has a fixed resolution and may lose information between output frames; ii) AvgPool is a method that performs pooling to reduce the temporal dimensions. AvgPool also has a fixed resolution, but it can aggregate information between output frames by pooling; iii) NNPool is the model that processes mel-spectrogram with a stack of $\mathrm{ResConv1D}$ (mentioned in Section 2.2.1) and performs average pooling to the target temporal dimension. NNPool has a total of 493,824 parameters and we provide the detailed structure in Table 6. Based on a trainable network, NNPool has the potential of learning better features and temporal resolution implicitly.

Baseline comparisons Table 1 shows our experimental result. The baseline of this experiment is performed on mel-spectrogram without compression (i.e., 100 FPS) and the baseline result is shown under each task name. When removing 25% of the temporal dimension (i.e., 75 FPS), the proposed method can even considerably improve the baseline performance on most datasets, except on speech recognition tasks where we maintain the same performance. We assume the improvement comes from the data augmention effect of DiffRes, which means divergent temporal compression on the same data at different training steps. With a 50 FPS, 80% datasets can maintain comparable performance. With only 25 FPS, the proposed method can still improve the FSD50K tagging and music instrument classification tasks, which indicates the high temporal redundancy in these datasets. Our proposed method also significantly outperforms other temporal dimension reduction baselines. With fixed resolution and fewer FPS, the performance of FixRes degrades more notably. AvgPool can outperform FixRes by aggregating more information between output frames. Although NNPool has an extra learnable neural network, it does not show significant improvements compared with AvgPool. NNPool even has an inferior performance on FSD50K and environmental sound classification tasks. This indicates employing a neural network for feature learning is not always optimal. Also, by comparison, the interpretability of our proposed method is much better than NNPool (see Figure 7).

On variable-length audio data Note that the proposed method improves the FSD50K tagging by the largest margin, even improving the performance by 1.3 with only 25% temporal dimensions. This is because FSD50K is a dataset with non-consistent audio length (see Table 5). In previous studies (Gong et al., 2021a;b; Kong et al., 2020), a common practice is padding audio data during batched training and inference, which introduces a considerable amount of temporal redundancy in the data with a significantly slower speed. By comparison, DiffRes can unify the audio feature shape regardless of their durations. Model optimization becomes more efficient with DiffRes. As a result, the proposed method can achieve comparable performance on FSD50K with only 15% temporal dimensions and 28% of the original training time with an mAP of 55.6 $\pm$ 0.2. This result shows that DiffRes provides a new mind map for future work on classifying variable-length audio files.

3.2 Learning with smaller hop size

Hop size	10 ms	6 ms	3 ms	1 ms
FPS (Compression rate)	100 (0%)	100 (40%)	100 (70%)	100 (90%)
AudioSet tagging	43.7 $\pm$ 0.1	**44.1** $\pm$ 0.1	43.8 $\pm$ 0.0	43.7 $\pm$ 0.1
Environmental Sound Classification	85.2 $\pm$ 0.4	87.2 $\pm$ 0.3	88.0 $\pm$ 0.6	**88.4** $\pm$ 0.5
Speech recognition	97.2 $\pm$ 0.1	97.6 $\pm$ 0.0	**97.9** $\pm$ 0.1	97.8 $\pm$ 0.1
Music instrument	79.9 $\pm$ 0.2	81.3 $\pm$ 0.3	**81.8** $\pm$ 0.2	80.6 $\pm$ 0.4
Average	76.5 $\pm$ 0.2	77.6 $\pm$ 0.2	**77.9** $\pm$ 0.1	77.5 $\pm$ 0.2

Table 2: Learning with high temporal resolution spectrograms. FPS is controlled at 100, so the computational complexity of the classifier is the same in all hop size settings. Result reported in percentage format.

Previous studies have observed higher resolution on spectrogram can improve audio classification accuracy (Kong et al., 2020; Ferraro et al., 2021). But the hop size smaller than 10 ms is not widely explored. This is mainly because the performance gain with a smaller hop size becomes less significant than the increased computational cost. For example, with 1 ms hop size (i.e., 1000 FPS), the time and space complexity for an EfficientNet classifier will be 10 times heavier than the 10 ms hop size. Since DiffRes can control the temporal dimension size, working on a small hop size spectrogram becomes computationally tractable. Table 2 shows model performance can be considerably improved with smaller hop sizes. AudioSet and environment sound dataset achieve the best performance on 6 ms and 1 ms hop size, and other tasks benefit most from 3 ms hop sizes. In later experiments, we will use these best hop size settings on each dataset.

Comparing with other learnable front-ends The DiffRes is learnable, so the Mel+DiffRes setting as a whole can be seen as a learnable front-end. Table 3 compares our proposed method with SOTA learnable front-ends, our best setting is denoted as Mel+DiffRes (Best), which achieves the best result on all datasets. For a fair comparison, we control the experiment setup to be consistent with Zeghidour et al. (2021) in another experiment Mel+DiffRes. Specifically, we change the backbone to EfficientNet-B0 (5.3 M parameters) without ImageNet pretraining. And we remove spec-augment

	Mel	TD-fbank	SincNet	LEAF	Mel+DiffRes	Mel+DiffRes (Best)
Parameters	0	51 k	256	448	82 k	82 k
AudioSet tagging	96.8 ± 0.1	96.5 ± 0.1	96.1 ± 0.0	96.8 ± 0.1	97.0 ± 0.0	**97.5** ± 0.0
Speech recognition	93.6 ± 0.3	89.5 ± 0.4	91.4 ± 0.4	93.6 ± 0.3	95.4 ± 0.2	**97.9** ± 0.1
Music instrument	70.7 ± 0.6	66.3 ± 0.6	67.4 ± 0.6	70.2 ± 0.6	78.5 ± 0.7	**81.8** ± 0.2
Average	87.0 ± 0.3	84.1 ± 0.4	85.0 ± 0.3	86.9 ± 0.3	90.3 ± 0.3	**92.4** ± 0.1

Table 3: Comparison with state-of-the-art learnable front-ends. All the methods use 100 FPS. Result reported in percentage format.

and mixup, except in AudioSet. We also change our mel bin from 128 to 40, except in the AudioSet experiment where we change to 64. The result shows Mel+DiffRes can outperform state-of-the-art learnable front-end (Zeghidour et al., 2021; Ravanelli & Bengio, 2018b; Zeghidour et al., 2018) by a large margin.

Computational cost We assess the one-second throughput of different front-ends on various FPS settings to compare their computational efficiency. We control the FPS of Mel and LEAF by average pooling. The computation time is measured between inputting waveform and outputting label prediction (with EfficientNet-B2). We use 128 filters in LEAF (Zeghidour et al., 2021) for a fair comparison with 128 mel-filterbanks in Mel and DiffRes. As shown in Figure 4, our proposed DiffRes only introduces marginal computational cost compared with Mel. The state-of-the-art learnable front-end, LEAF, is about four times slower than our proposed method.

3.3 Analysis for the learning of DiffRes

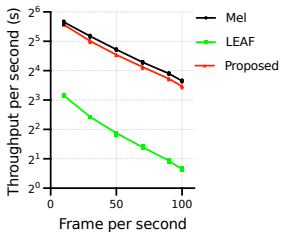


Figure 4: Audio throughput in one second. Evaluated on a 2.6 GHz Intel Core i7 CPU.

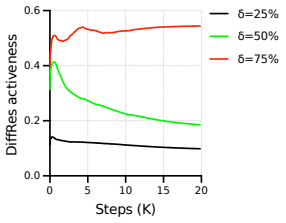


Figure 5: Trajectories of DiffRes learning activeness (ρ) on different training steps and compression rate (δ).

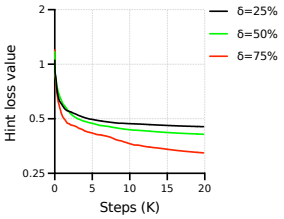


Figure 6: The training curve of hint loss ($\mathcal{L}_{hint}$) with different compression rate (δ).

Learning activeness DiffRes does not explicitly learn the optimal frame importance score because the ground truth is not available. Instead, DiffRes is optimized with the guidance of hint loss $\mathcal{L}_{hint}$ (Equation 12), which is a strong assumption we introduced to the model. Figure 5 shows the activeness (defined in Section 2.2.1) curve of DiffRes during the optimization with different compression rate δ settings on the speech recognition task. According to the final converged value, DiffRes with a higher δ tend to be more active at learning frame importance. This is intuitive since higher δ leads to more information bottleneck effects (Saxe et al., 2019) in DiffRes. With a 75% compression rate, the activeness even keeps increasing with more training steps, indicating the active learning of DiffRes. Figure 6 shows the hint loss curve during training with different δ settings. Intuitively, when the δ is high, a model needs to preserve more non-empty frames for better accuracy. This assumption is aligned with our experiment result, which shows the model tends to have a lower hint loss with higher δ.

Data augmentation and regularization effect As reflected in the curve of ρ and $\mathcal{L}_{hint}$ in Figure 5 and 6, DiffRes is optimized along with the classifier during training. Hence DiffRes may perform different temporal compression for the same training data at different epochs. This is equivalent to performing data augmentation on the audio data. We suppose this is the main reason for the improved performance shown in Table 1. Also, DiffRes reduces the sparsity of the audio feature by adaptive temporal compression. This is equivalent to performing an implicit regulariza-

tion (Neyshabur, 2017; Arora et al., 2019) on the feature level, which is beneficial for the system efficiency.

Weakness analysis Table 2 shows DiffRes does not improve the model performance on 1 ms setting on most datasets. We assume this is mostly owing to the insufficient receptive field of the convolutions in DiffRes, which is around 41 time steps. By comparison, the temporal dimension of $\boldsymbol{X}$ on AudioSet is $t = 3,333$ and $t = 10,000$ with 3 ms and 1 ms hop size, respectively. DiffRes may not effectively capture the useful information with only 41 temporal receptive field in this case. Future work will address this problem by designing the resolution prediction model with a large receptive field. Also, although we use the same λ setting in the hint loss (see Equation 12) in all the experiments, our experiment shows a proper tuning of λ can still affect the performance, which might require manual parameter tuning.

4 Related Works

Neural-network based methods have been successfully applied on audio classification and achieved state-of-the-art performance, such as PANNs (Kong et al., 2020), PSLA (Gong et al., 2021b), AST (Gong et al., 2021a). We will cover two related topics on audio classification in the following sections: i) learnable audio front-ends; and ii) learning feature resolution in neural networks.

4.1 Learnable audio front-ends

In recent years, learning acoustic features from waveform using trainable audio front-ends has attracted increasing interest from researchers. Sainath et al. (2013) is one of the earliest works that propose to jointly learn the parameter of filter banks with a speech recognition model. Later, SincNet (Ravanelli & Bengio, 2018b) proposes to learn a set of bandpass filters on the waveform and has shown success on speaker recognition (Ravanelli & Bengio, 2018b;a). Most recently, (Zeghidour et al., 2021) proposes to learn bandpass, and lowpass filtering as well as per-channel compression (Wang et al., 2017) simultaneously in the audio front-end and shows consistent improvement in audio classification. Different from existing work on learnable audio front-ends, which mostly focus on the frequency dimension, our objective is learning the optimal temporal resolution. We show that our method can outperform existing audio front-ends for audio classification on both accuracy and computation efficiency (see Table 3 and Figure 4). Note that our proposed method can also be applied after most learnable front-ends (Zeghidour et al., 2021), which will be our future direction.

4.2 Learning feature resolution with neural networks

One recent work on learning feature resolution for audio classification is DiffStride (Riad et al., 2021), which learns stride in convolutional neural network (CNN) in a differentiable way and outperforms previous methods using fixed stride settings. By comparison, DiffStride needs to be applied in each CNN layer and can only learn a single fixed stride setting, while DiffRes is a one-layer lightweight algorithm and can personalize the best temporal resolution for each audio during inference. Recently, (Gazneli et al., 2022) proposed to use a stack of one-dimension-CNN blocks to downsample the audio waveform before the audio classification backbone network (e.g., Transformer), which can learn temporal resolution implicitly for audio classification. In contrast, DiffRes can explicitly learn temporal resolution on the feature level with similar interpretability as mel-spectrogram.

5 Conclusions

In this paper, we introduce DiffRes, a "drop-in" differentiable temporal resolution learning module that can be applied between mel-spectrogram and downstream tasks. We demonstrate over a large range of tasks that DiffRes can improve or maintain similar performance with 25% to 75% reduction on temporal dimensions, and can also efficiently utilize the information in high-resolution spectrograms to improve accuracy. In future work, we will move forward to evaluate DiffRes on other time-frequency representations and evaluate with more sophisticated frame importance prediction models. Also, we believe the idea of DiffRes can work on other time series data as well for redundancy removals, such as video or seismic data.

REFERENCES

Sanjeev Arora, Nadav Cohen, Wei Hu, and Yuping Luo. Implicit regularization in deep matrix factorization. *Advances in Neural Information Processing Systems*, 32, 2019. 9

Sebastian Böck, Andreas Arzt, Florian Krebs, and Markus Schedl. Online real-time onset detection with recurrent neural networks. In *Proceedings of the 15th International Conference on Digital Audio Effects, York, UK*, 2012. 2

Rodrigo Castellon, Chris Donahue, and Percy Liang. Codified audio language modeling learns useful representations for music information retrieval. *arxiv:2107.05677*, 2021. 1

David C Champeney and DC Champeney. *A handbook of fourier theorems*. Cambridge University Press, 1987. 1

Ke Chen, Xingjian Du, Bilei Zhu, Zejun Ma, Taylor Berg-Kirkpatrick, and Shlomo Dubnov. Hts-at: A hierarchical token-semantic audio transformer for sound classification and detection. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 646–650, 2022. 2

Kwanghee Choi, Martin Kersner, Jacob Morton, and Buru Chang. Temporal knowledge distillation for on-device audio classification. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 486–490, 2022. 6

Pieter-Tjerk De Boer, Dirk P Kroese, Shie Mannor, and Reuven Y Rubinstein. A tutorial on the cross-entropy method. *Annals of operations research*, 134(1):19–67, 2005. 3

Jesse Engel, Cinjon Resnick, Adam Roberts, Sander Dieleman, Douglas Eck, Karen Simonyan, and Mohammad Norouzi. Neural audio synthesis of musical notes with wavenet autoencoders, 2017. 6

Andres Ferraro, Dmitry Bogdanov, Xavier Serra Jay, Ho Jeon, and Jason Yoon. How low can you go? reducing frequency and time resolution in current CNN architectures for music auto-tagging. In *IEEE European Signal Processing Conference*, pp. 131–135, 2021. 1, 7

Eduardo Fonseca, Xavier Favory, Jordi Pons, Frederic Font, and Xavier Serra. FSD50K: an open dataset of human-labeled sound events. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30:829–852, 2021. 5

Kunihiko Fukushima and Sei Miyake. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets*, pp. 267–285. Springer, 1982. 3

Dennis Gabor. Theory of communication. Part 1: The analysis of information. *Journal of the Institution of Electrical Engineers-part III: radio and communication engineering*, 93(26):429–441, 1946. 1

Avi Gazneli, Gadi Zimerman, Tal Ridnik, Gilad Sharir, and Asaf Noy. End-to-end audio strikes back: Boosting augmentations towards an efficient audio classification network. *arXiv:2204.11479*, 2022. 9

Jort F Gemmeke, Daniel PW Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R Channing Moore, Manoj Plakal, and Marvin Ritter. Audio set: An ontology and human-labeled dataset for audio events. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 776–780, 2017. 2, 5

Yuan Gong, YuAn Chung, and James Glass. AST: Audio spectrogram transformer. *arXiv:2104.01778*, 2021a. 1, 2, 6, 7, 9

Yuan Gong, YuAn Chung, and James Glass. PSLA: Improving audio tagging with pretraining, sampling, labeling, and aggregation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:3292–3306, 2021b. 1, 6, 7, 9

Yuan Gong, Cheng-I Lai, Yu-An Chung, and James Glass. SSAST: Self-supervised audio spectrogram transformer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 10699–10709, 2022. 2

Diana F Gordon and Marie Desjardins. Evaluation and selection of biases in machine learning. *Machine learning*, 20(1):5–22, 1995. 4

Jun Han and Claudio Moraga. The influence of the sigmoid function parameters on the speed of backpropagation learning. In *International workshop on artificial neural networks*, pp. 195–201. Springer, 1995. 3

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016. 4

Jonathan J Huang and Juan Jose Alvarado Leanos. AclNet: efficient end-to-end audio classification cnn. *arXiv:1811.06669*, 2018. 6

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456. PMLR, 2015. 4

Byeonggeun Kim, Simyung Chang, Jinkyu Lee, and Dooyong Sung. Broadcasted residual learning for efficient keyword spotting. *arxiv:2106.04140*, 2021. 1

Jaehyeon Kim, Sungwon Kim, Jungil Kong, and Sungroh Yoon. Glow-TTS: A generative flow for text-to-speech via monotonic alignment search. *Advances in Neural Information Processing Systems*, 33:8067–8077, 2020. 3

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014. 15

Qiuqiang Kong, Yin Cao, Turab Iqbal, Yuxuan Wang, Wenwu Wang, and Mark D Plumbley. PANNs: Large-scale pretrained audio neural networks for audio pattern recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:2880–2894, 2020. 1, 2, 6, 7, 9

Qiuqiang Kong, Yin Cao, Haohe Liu, Keunwoo Choi, and Yuxuan Wang. Decoupling magnitude and phase estimation with deep resunet for music source separation. *arXiv:2109.05418*, 2021. 1

Khaled Koutini, Jan Schlüter, Hamid Eghbal-zadeh, and Gerhard Widmer. Efficient training of audio transformers with patchout. *arXiv:2110.05069*, 2021. 2

Yann LeCun, Bernhard Boser, John Denker, Donnie Henderson, Richard Howard, Wayne Hubbard, and Lawrence Jackel. Handwritten digit recognition with a back-propagation network. *Advances in neural information processing systems*, 2, 1989. 3

Jongpil Lee, Taejun Kim, Jiyoung Park, and Juhan Nam. Raw waveform-based audio classification using sample-level cnn architectures. *arxiv:1712.00866*, 2017. 1

Haohe Liu, Xubo Liu, Xinhao Mei, Qiuqiang Kong, Wenwu Wang, and Mark D Plumbley. Segment-level metric learning for few-shot bioacoustic event detection. *arXiv:2207.07773*, 2022. 1

Yi Luo and Nima Mesgarani. Tasnet: time-domain audio separation network for real-time, single-channel speech separation. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 696–700, 2018. 1

Tom M Mitchell. *The need for biases in learning generalizations*. Department of Computer Science, Laboratory for Computer Science Research, 1980. 4, 5

Behnam Neyshabur. Implicit regularization in deep learning. *arXiv:1709.01953*, 2017. 9

Daniel S Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D Cubuk, and Quoc V Le. SpecAugment: A simple data augmentation method for automatic speech recognition. *arXiv:1904.08779*, 2019. 6

Karol J Piczak. ESC: Dataset for environmental sound classification. In *ACM international conference on Multimedia*, pp. 1015–1018, 2015. 6

Mirco Ravanelli and Yoshua Bengio. Interpretable convolutional filters with sincnet. *arXiv:1811.09725*, 2018a. 9

Mirco Ravanelli and Yoshua Bengio. Speaker recognition from raw waveform with SincNet. In *IEEE Spoken Language Technology Workshop*, pp. 1021–1028, 2018b. 2, 8, 9

Rachid Riad, Olivier Teboul, David Grangier, and Neil Zeghidour. Learning strides in convolutional neural networks. In *International Conference on Learning Representations*, 2021. 6, 9

Oleg Rybakov, Marco Tagliasacchi, Yunpeng Li, Liyang Jiang, Xia Zhang, and Fadi Biadsy. Real time spectrogram inversion on mobile phone. *arXiv:2203.00756*, 2022. 2

Tara N Sainath, Brian Kingsbury, Abdelrahman Mohamed, and Bhuvana Ramabhadran. Learning filter banks within a deep neural network framework. In *IEEE workshop on automatic speech recognition and understanding*, pp. 297–302, 2013. 2, 9

Andrew M Saxe, Yamini Bansal, Joel Dapello, Madhu Advani, Artemy Kolchinsky, Brendan D Tracey, and David D Cox. On the information bottleneck theory of deep learning. *Journal of Statistical Mechanics: Theory and Experiment*, 2019(12):124020, 2019. 8

Claude Elwood Shannon. A mathematical theory of communication. *ACM mobile computing and communications review*, 5(1):3–55, 2001. 5

Jonathan Shen, Ruoming Pang, Ron J Weiss, Mike Schuster, Navdeep Jaitly, Zongheng Yang, Zhifeng Chen, Yu Zhang, Yuxuan Wang, Rj Skerrv-Ryan, et al. Natural TTS synthesis by conditioning wavenet on mel spectrogram predictions. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 4779–4783, 2018. 2

Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information. *arXiv:1703.00810*, 2017. 5

Stanley Smith Stevens, John Volkmann, and Edwin Broomell Newman. A scale for the measurement of the psychological magnitude pitch. *The Journal of the Acoustical Society of America.*, 8(3): 185–190, 1937. 2

Mingxing Tan and Quoc Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pp. 6105–6114. PMLR, 2019. 6

Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. *arXiv physics/0004057*, 2000. 5

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017. 5

Sergey Verbitskiy, Vladimir Berikov, and Viacheslav Vyshegorodtsev. ERANNs: Efficient residual audio neural networks for audio pattern recognition. *Pattern Recognition Letters*, 161:38–44, 2022. 2

Yuxuan Wang, Pascal Getreuer, Thad Hughes, Richard F Lyon, and Rif A Saurous. Trainable frontend for robust and far-field keyword spotting. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 5670–5674, 2017. 9

Pete Warden. Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv:1804.03209*, 2018. 6

Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *arXiv:1505.00853*, 2015. 4

Neil Zeghidour, Nicolas Usunier, Iasonas Kokkinos, Thomas Schatz, Gabriel Synnaeve, and Emmanuel Dupoux. Learning filterbanks from raw speech for phone recognition. In *IEEE Acoustics, Speech and Signal Processing*, 2018. 2, 8

Neil Zeghidour, Olivier Teboul, Félix de Chaumont Quitry, and Marco Tagliasacchi. LEAF: A learnable frontend for audio classification. *arxiv:2101.08596*, 2021. 2, 6, 7, 8, 9

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. MixUp: Beyond empirical risk minimization. *arXiv:1710.09412*, 2017. 6

A APPENDIX

A.1 FIGURES

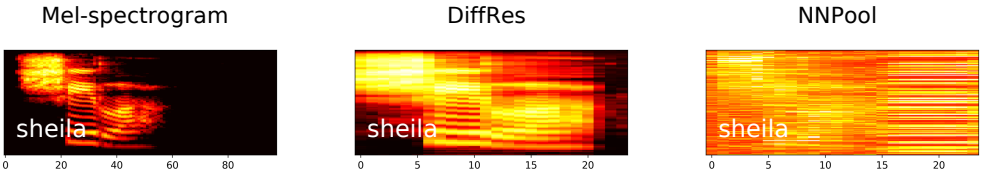
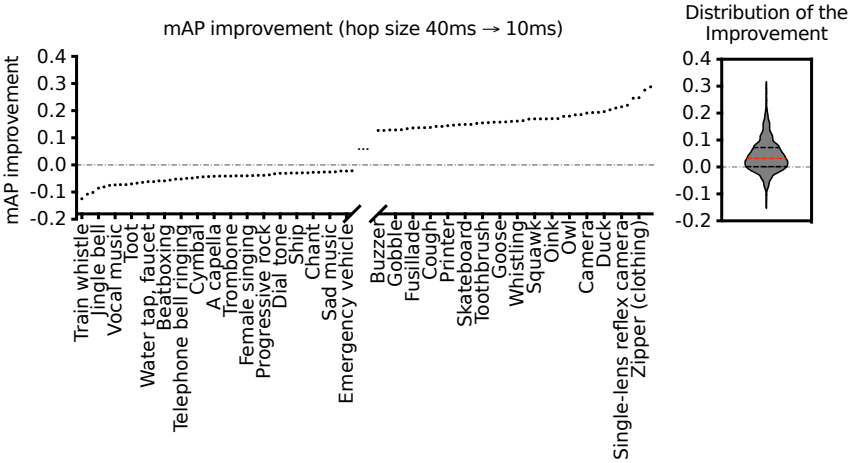
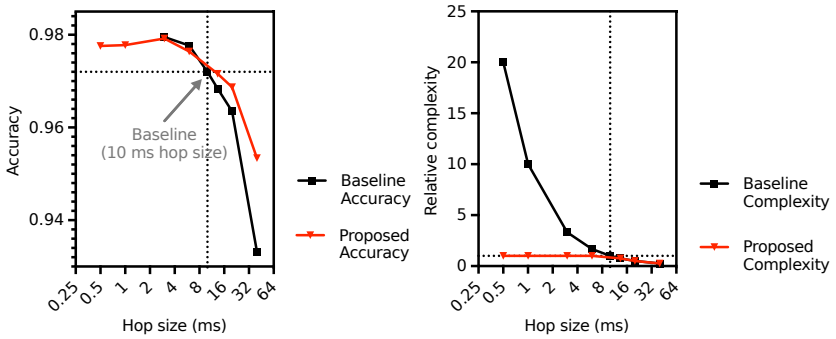


Figure 7: Comparison of mel-spectrogram, DiffRes feature, and NNPool learnt feature. DiffRes feature is more interpretable than the NNPool feature.



(a) Class-wise improvement after changing hop size from 40 ms to 10 ms.



(b) Accuracy and the classifier computational complexity with different hop size settings on speech recognition task.

Figure 8: Pilot studies. a) The mAP improvement for each class in the AudioSet after decreasing the hop size from 40ms to 10ms. The violin plot on the right side shows the improvement distribution, where the red dashed line is the median value. The inconsistency of improvement in different sound classes indicates they need different temporal resolutions to achieve optimal classification performance. a) The accuracy can be improved with a smaller hop size at the cost of computation. DiffRes can achieve similar improvements without increasing computational complexity.

A.2 FAST IMPLEMENTATION OF ALGORITHM 1

We provide a pure matrix operation based version of Algorithm 1 for efficient model optimization on GPUs. Given the output of Equation 9, $\boldsymbol{W}_0$, we will process it into a matrix $\boldsymbol{W}$ with the same shape so that $\sum_{i=1}^{\gamma} \boldsymbol{W}_{i,:} = \mathbf{1}$ and $\sum_{i=1,j=1}^{\gamma,t} \boldsymbol{W}_{i,j} \leq \gamma$. Figure 9 provides an example of the fast implementation discussed in this section.

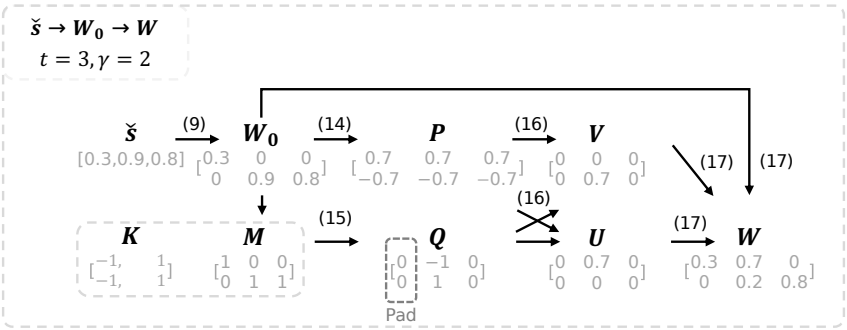


Figure 9: An example of the Algorithm 1 vectorized implementation. The number inside the parentheses means the equation number.

First, we calculate the cumulative sum of the distance between one and the total input weight assigned to each output frame, given by

$$\boldsymbol{P} = \mathrm{cumsum}(\mathbf{1}_\gamma - \sum^{t} \boldsymbol{W}_0)\mathbf{1}_t^T, \tag{14}$$

where $\mathbf{1}_t$ denotes the all-one column vector with shape t. For each row in $\boldsymbol{W}_0$, only the first non-negative element and the first zero after the last non-negative element need update (see Algorithm 1). And we locate those elements by performing the following convolution: convolving (no padding) $\boldsymbol{M}$ with a kernel $\boldsymbol{K}$ of shape $\gamma \times 2$, given by

$$\boldsymbol{Q} = \mathrm{Conv1D}(\boldsymbol{M}, \boldsymbol{K}), \quad \boldsymbol{K} = \begin{bmatrix} -1 & 1 \\ -1 & 1 \\ \dots & \dots \\ -1 & 1 \end{bmatrix}_{\gamma \times 2}, \quad \boldsymbol{M} = \mathrm{sgn}(\boldsymbol{W}_0), \tag{15}$$

where $\mathrm{sgn}(\cdot)$ stands for a sign function. We pad a column of zero on the first row of the output Q. The output $\boldsymbol{Q} \in \{0, 1, -1\}$ is a matrix with shape $\gamma \times t$. Then we calculate the value that needs updates with $\boldsymbol{P}$ and $\boldsymbol{Q}$, given by

$$\boldsymbol{U} = \boldsymbol{P} \odot (-\boldsymbol{Q})^+, \boldsymbol{V} = \begin{bmatrix} \mathbf{0}_t \\ \boldsymbol{P}_{1:\gamma-1} \odot \boldsymbol{Q}^+_{2:\gamma} \end{bmatrix} \tag{16}$$

where $\boldsymbol{P}_{1:\gamma-1} = (\boldsymbol{P}_{ij})_{i \in [1:\gamma-1], j \in [1,t]}$ denotes slicing of the matrix row, $\mathbf{0}_t$ is the all-zero row vector with length t, and $\odot$ denotes element-wise multiplication. Note that here the start index of a matrix is one. $\boldsymbol{U}$ and $\boldsymbol{V}$ store the values that need addition and subtraction for each element in $\boldsymbol{W}_0$. The final weight matrix is calculated by

$$\boldsymbol{W} = \boldsymbol{U} - \boldsymbol{V} + \boldsymbol{W}_0. \tag{17}$$

A.3 DATASET AND EXPERIMENT DETAILS

Dataset	Learning rate	Epoch	Batchsize	Learning rate scheduler (start epoch, gamma, every n epoch)	GPU(s)
Audioset	1.0×10^{-4}	30	22	$(11, 0.5, 5)$	4
FSD50K	5.0×10^{-4}	40	15	$(21, 0.5, 5)$	1
ESC50	2.5×10^{-4}	80	32	$(41, 0.95, 1)$	1
SpeechCommands	2.5×10^{-4}	60	128	$(25, 0.9, 1)$	1
NSynth	1.0×10^{-4}	30	48	$(11, 0.85, 1)$	1

Table 4: Hyper-parameter setting. We run all the experiments with an ADAM optimizer (Kingma & Ba, 2014) and GeForce RTX 2080 Ti GPU(s).

Task	Audio tagging	Audio tagging	Environmental sound	Speech recognition	Music instrument
Dataset	AudioSet	FSD50K	ESC50	SpeechCommands	NSynth
Classes	527	200	50	35	11
Train examples	$1,912,134$	$36,799$	$2,000$	$84,771$	$289,205$
Test examples	$18,887$	$10,231$	-	$10,700$	$12,678$
Duration (mean,std)	$9.91, 0.50$	$7.63, 7.82$	$5.00, 0.00$	$0.98, 0.07$	$4.00, 0.00$
Pad to length	1000	3000	500	98	400
Evaluation metric	mAP	mAP	Accuracy	Accuracy	Accuracy
5-fold cross-validation	-	-	✓	-	-
Class re-balancing	✓	✓	-	-	-

Table 5: Detailed information of the datasets we used in this paper. We perform padding to unify the data length. The last row shows the mel-spectrogram temporal dimension we used for batched training.

	Frame importance estimation module $\mathcal{H}_\psi$	NNPool Encoder
Parameters (k)	82.4	493.8
Kernel size	5	5
$\mathrm{ResConv1D}$ blocks (input chnanel, onput chnanel)	$(128, 64), (64, 32), (32, 16), (16, 8), (8, 1)$	$(128, 128), (128, 128), (128, 128)$

Table 6: The structure of the frame importance estimation module and the front-end structure of NNPool (baseline method in Table 1). The structure of $\mathrm{ResConv1D}$ is discussed in Section 2.2.1
.

A.4 ALGORITHM FORMULATION OF DIFFRES

Algorithm 2: Perform audio classification with the DiffRes layer

Inputs : $\boldsymbol{x} \in \mathbb{R}^L$: One-dimensional waveform. $\mathcal{F}_\psi$: DiffRes layer. $\mathcal{G}_\theta$: Classifier backbone.
Output: $\hat{y}$: Classification result.

$\boldsymbol{X}^{f \times t} \longleftarrow F(\boldsymbol{x})$; ▷ Project input to the Fourier domain with fixed temporal resolution.
$\boldsymbol{O}^{\gamma \times n}, \mathcal{E}^{\gamma \times n} \longleftarrow \mathcal{F}_\psi(\boldsymbol{X})$; ▷ DiffRes layer (see Algorithm 3). $\mathcal{E}^{\gamma \times n}$ is the resolution encoding.
$\hat{y} \longleftarrow \mathcal{G}_\theta(\boldsymbol{O}, \mathcal{E})$; ▷ Classifier forward propagation.
if *training* **then**
  $loss = \mathcal{L}_{bce} + \mathcal{L}_{hint}$; ▷ $\mathcal{L}_{hint}$ is the loss function for DiffRes (See Equation 12)
  backprop;

Algorithm 3: DiffRes layer $\mathcal{F}_\psi$

Inputs : $\boldsymbol{X} \in \mathbb{R}^{f \times t}$: Fix-time-resolution spectrogram. $\boldsymbol{E} \in \mathbb{R}^{f \times t}$: Positional encoding.
$\mathcal{H}_\psi$: Frame importance estimation network. $\gamma \in \mathbb{Z}$: Target temporal dimensions.
Output: $\boldsymbol{O} \in \mathbb{R}^{\gamma \times n}$: Two-dimensional representation with adaptive temporal resolution.
$\mathcal{E} \in \mathbb{R}^{f \times \gamma}$: Resolution encoding.

$\check{\boldsymbol{s}}^{t \times 1} \longleftarrow \mathrm{Norm}(\sigma(\mathcal{H}_\psi(\boldsymbol{X})))$; ▷ Calculate frame importance (see Equation 5, and 6).
$\boldsymbol{W}^{\gamma \times t} \longleftarrow \alpha(\check{\boldsymbol{s}})$; ▷ Construct a weight matrix (see Equation 9 and Algorithm 1).
$\boldsymbol{O}^{\gamma \times n} \longleftarrow \beta(\boldsymbol{X}, \boldsymbol{W})$; ▷ Merge frames with the weight matrix (see Equation 10).
$\mathcal{E}^{\gamma \times n} \longleftarrow \boldsymbol{W}\boldsymbol{E}$; ▷ Encode the information of $\boldsymbol{W}$ into $\mathcal{E}$

A.5 EXAMPLES

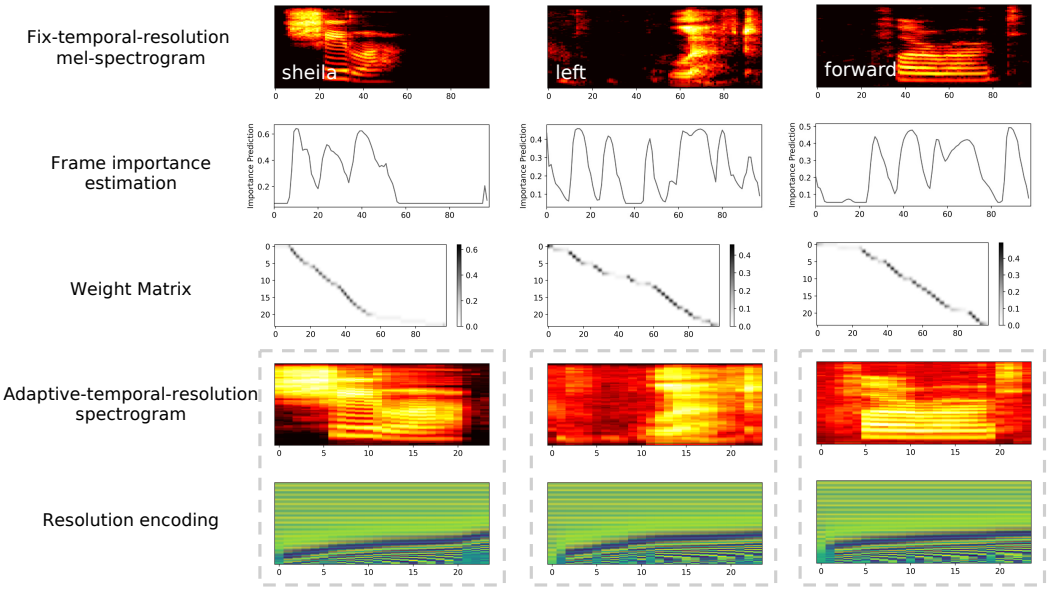


Figure 10: Examples of DiffRes adaptive-temporal-resolution spectrogram on the SpeechCommands dataset.

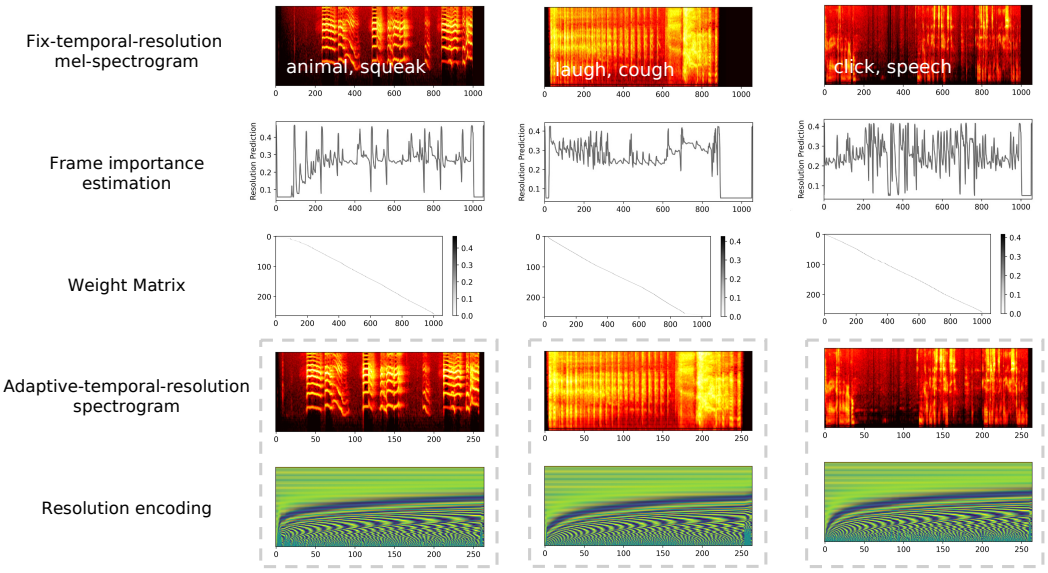


Figure 11: Examples of DiffRes adaptive-temporal-resolution spectrogram on the AudioSet dataset.

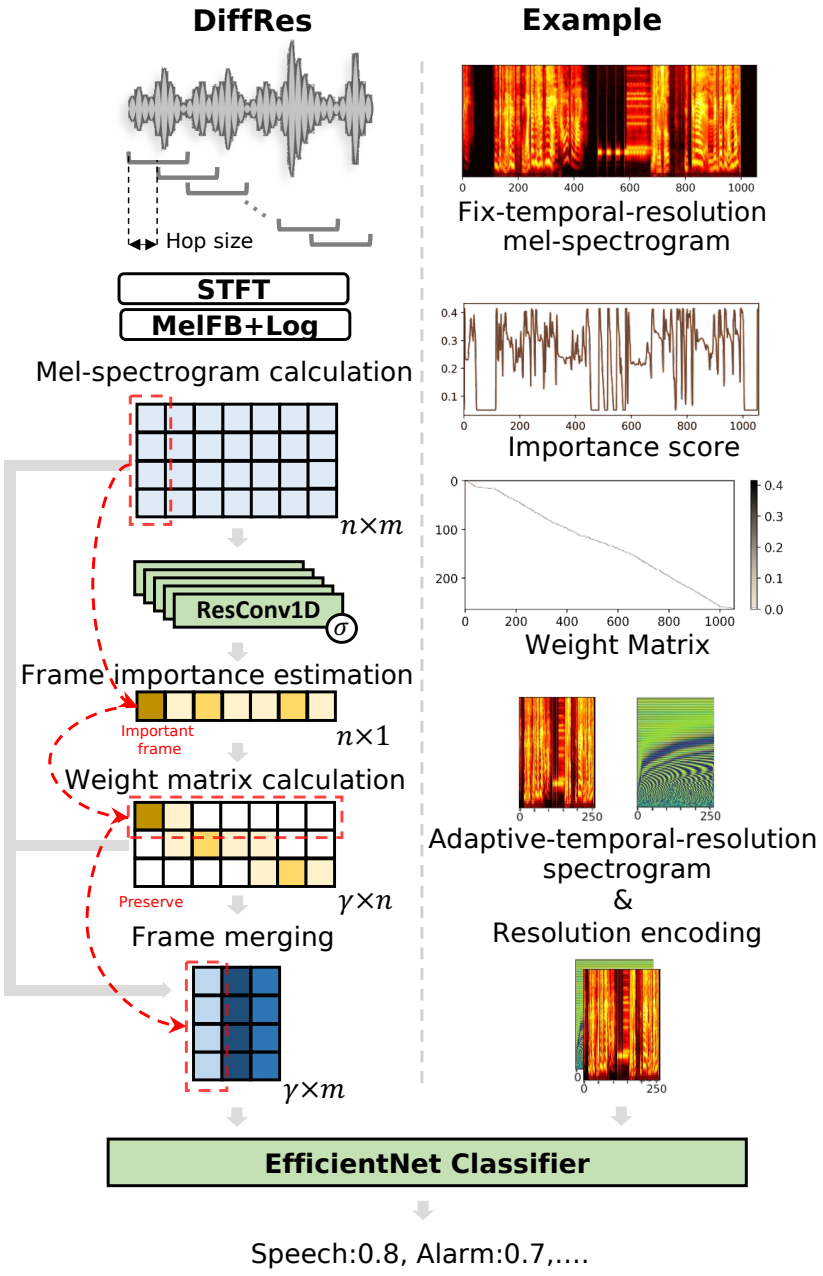


Figure 12: Overview of applying DiffRes on audio classification. The right side shows an example.