# APPLADE: ADJUSTABLE PLUG-AND-PLAY AUDIO DECLIPPER COMBINING DNN WITH SPARSE OPTIMIZATION

*Tomoro Tanaka[†], Kohei Yatabe[†], Masahiro Yasuda[‡], Yasuhiro Oikawa[†]*

[†] Department of Intermedia Art and Science, Waseda University, Tokyo, Japan
[‡] NTT Corporation, Tokyo, Japan

## ABSTRACT

In this paper, we propose an audio declipping method that takes advantages of both sparse optimization and deep learning. Since sparsity-based audio declipping methods have been developed upon constrained optimization, they are adjustable and well-studied in theory. However, they always uniformly promote sparsity and ignore the individual properties of a signal. Deep neural network (DNN)–based methods can learn the properties of target signals and use them for audio declipping. Still, they cannot perform well if the training data have mismatches and/or constraints in the time domain are not imposed. In the proposed method, we use a DNN in an optimization algorithm. It is inspired by an idea called plug-and-play (PnP) and enables us to promote sparsity based on the learned information of data, considering constraints in the time domain. Our experiments confirmed that the proposed method is stable and robust to mismatches between training and test data.

***Index Terms***— Sparse optimization, audio declipping, proximity operator, soft-thresholding, ADMM algorithm.

## 1. INTRODUCTION

The aim of audio declipping is to restore clipped signals. Clipping is a common nonlinear distortion that occurs due to restriction of recording system and/or audio processing. As illustrated in Fig. 1, the samples of the true signal exceeding a dynamic range $[-\tau, \tau]$ with $\tau > 0$ are truncated (so-called *hard clipping*). Since clipping degrades audio quality [1, 2] and has bad effects on subsequent processing [3, 4], audio declipping is desired.

Audio declipping has been realized by various methods [5–13]. Among them, sparsity-based methods [5–10] have been studied for a long time. They introduce an optimization problem consisting of a data-fidelity term and a sparsity-promoting term. The data-fidelity term makes solutions feasible, and *clipping consistency* [10] was considered in recent studies [7–9]. The sparsity-promoting term makes solutions sparse in the time-frequency (T-F) domain. This is motivated by the fact that clipping produces extra harmonic components. Sparsity-based methods are well studied in theory and have some adjustable parameters for variation in the properties of data.

Deep neural network (DNN)–based audio declipping methods have also been studied lately [11–13]. A DNN can learn the properties of data and perform audio declipping based on them. In [12], deep filtering was utilized, and complex-valued filters were estimated by a DNN composed of the bidirectional long short-term memory layers [14]. In [13], the waveforms or magnitude spectrograms of non-clipped signals were directly estimated by a DNN which has the U-Net architecture [15]. However, these methods cannot perform well when the properties of input data (such as scale and clipping threshold $\tau$) are greatly different from those of the training data. Furthermore, the clipping consistency has not been explicitly considered during processing.
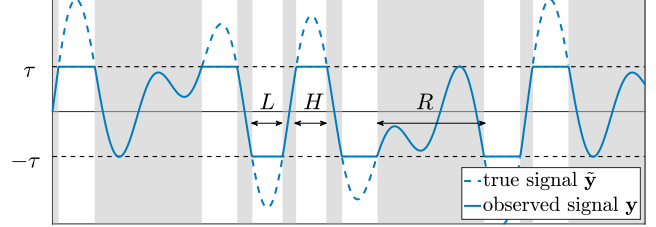


**Fig. 1**. The true signal $\tilde{\mathbf{y}}$ and an observed clipped signal $\mathbf{y}$ with the clipping threshold $\tau$. Gray-filled areas indicate that clipping has no effect. $R$ is the set of the non-clipped indices, and $H$ and $L$ are the sets of the clipped indices above and below $\pm\tau$, respectively.

In this paper, we propose an audio declipping method that combines a DNN with sparse optimization and named it APPLADE (**a**djustable **p**lug-and-**pl**ay **a**udio **de**clipper). We use a DNN in an optimization algorithm to promote sparsity based on the learned properties of data, considering the clipping consistency. Since the proposed method inherits the adjustability of the sparsity-based methods, it can handle a wide variety of data whose properties are greatly different from the training data. The DNN and its usage are designed not to bother stability of an optimization algorithm and not to increase execution time so much. Our experiments confirmed that the proposed method is stable and robust to differences between training and test data.

## 2. PRELIMINARIES

In this paper, we consider hard clipping, in which signals' samples that exceed a certain dynamic range $[-\tau, \tau]$ are truncated. Let $\tilde{\mathbf{y}} \in \mathbb{R}^T$ be the true signal, and $\mathbf{y} \in \mathbb{R}^T$ an observed clipped signal,

$$y[t] = \begin{cases} \tau & (\tilde{y}[t] \geq \tau) \\ \tilde{y}[t] & (-\tau < \tilde{y}[t] < \tau), \\ -\tau & (\tilde{y}[t] \leq -\tau) \end{cases} \tag{1}$$

where $T$ is the length of the signal. Our aim is to estimate the true signal $\tilde{\mathbf{y}}$ only from the clipped signal $\mathbf{y}$. The situation is briefly shown in Fig. 1. As illustrated, their indices can be divided into three disjoint sets, $H = \{t \in [1, T] \mid y[t] \geq \tau\}$, $R = \{t \in [1, T] \mid |y[t]| < \tau\}$, and $L = \{t \in [1, T] \mid y[t] \leq -\tau\}$.

### 2.1. Sparsity-based audio declipping methods

Sparse optimization has been applied to audio declipping [5–10]. It is realized by solving the following optimization problem:

$$\text{Find } \mathbf{x}^\star \in \arg\min_{\mathbf{x}} \mathcal{S}(\mathcal{G}\mathbf{x}) \text{ subject to } \mathbf{x} \in \Gamma, \tag{2}$$

where the set of feasible signals $\Gamma$ is defined as

$$\Gamma = \left\{ \mathbf{x} \in \mathbb{R}^T \;\middle|\; \begin{array}{ll} x[t] \geq \tau & (t \in H) \\ x[t] = y[t] & (t \in R) \\ x[t] \leq -\tau & (t \in L) \end{array} \right\}, \qquad (3)$$

$\mathcal{S}$ is a sparsity-promoting function (including $\ell_1$-norm, weighted $\ell_1$-norm [8], $\ell_0$-norm [5,7]), $\mathcal{G}$ is the discrete Gabor transform (DGT),

$$(\mathcal{G}\mathbf{x})[m,n] = \sum_{t=0}^{T-1} x[t+an]\, g[t]\, e^{-2\pi \mathrm{i} mt/M}, \qquad (4)$$

$\mathbf{g} \in \mathbb{R}^T$ is a window, i is the imaginary unit, $a$ denotes the time shifting step, $n \in [0, N-1]$ and $m \in [0, M-1]$ are the time and frequency indices, respectively, satisfying $aN = T$, and the indices are to be understood modulo $T$. From Eq. (1), $x^\star[t]$ should be exactly same as $y[t]$ for all $t \in R$. Moreover, it must be greater than $\tau$ for all $t \in H$ and smaller than $-\tau$ for all $t \in L$. That is, every solution must coincide with $\mathbf{y}$ after applying the same clipping. This is called clipping consistency [10], and the constraint in Eq. (2) forces $\mathbf{x}^\star$ to satisfy this consistency. The sparsity-promoting term $\mathcal{S}(\mathcal{G}\mathbf{x})$ makes a solution sparse in the T-F domain. Since clipping produces extra harmonic components, promoting sparsity in the T-F domain will remove them and make a solution closer to non-clipped signals.

### 2.2. DNN-based methods

DNN-based audio declipping methods have begun to be studied lately [11–13]. A DNN approximates a mapping that estimates the true signal only from the features of a clipped signal. Let $\mathcal{F}^{\text{TFD}}$ and $\mathcal{F}^{\text{TD}}$ be mappings in the time domain and T-F domain, respectively. In [11], the target mapping was $\mathcal{F}^{\text{TFD}}(\text{mel}(\mathcal{G}\mathbf{y})) = \text{mel}(\mathcal{G}\tilde{\mathbf{y}})$, where $\text{mel}(\cdot)$ extracts the mel-frequency cepstrum coefficients features. In [12], a DNN was trained to estimate the complex-valued deep filters. In [13], both mapping $\mathcal{F}^{\text{TFD}}(\log(|\mathcal{G}\mathbf{y}|^2)) = \log(|\mathcal{G}\tilde{\mathbf{y}}|^2)$ and $\mathcal{F}^{\text{TD}}(\mathbf{y}) = \tilde{\mathbf{y}}$ were approximated, where $|\cdot|$ is the element-wise absolute value. DNN-based methods can restore non-clipped signals based on the learned properties of training data.

### 3. PROPOSED METHOD: APPLADE

In this paper, we propose an audio declipping method named APPLADE. It is inspired by the plug-and-play (PnP) method [16–21], which utilizes a DNN in an optimization algorithm.

### 3.1. Alternating Direction Method of Multiplier (ADMM)

In this paper, we use ADMM [22] to solve Eq. (2). By applying ADMM to Eq. (2), we obtain the following iterative procedure:

$$\mathbf{x}^{[k+1]} = \Pi_\Gamma(\mathcal{G}^*(\mathbf{v}^{[k]} - \mathbf{u}^{[k]})), \qquad (5)$$

$$\mathbf{v}^{[k+1]} = \text{prox}_{(1/\rho)\mathcal{S}}(\mathcal{G}\mathbf{x}^{[k+1]} + \mathbf{u}^{[k]}), \qquad (6)$$

$$\mathbf{u}^{[k+1]} = \mathbf{u}^{[k]} + \mathcal{G}\mathbf{x}^{[k+1]} - \mathbf{v}^{[k+1]}, \qquad (7)$$

where $\rho \in \mathbb{R}_+$, $\mathbf{v} \in \mathbb{C}^{M \times N}$ and $\mathbf{u} \in \mathbb{C}^{M \times N}$ are the auxiliary variables, $\mathcal{G}^*$ is the adjoint of $\mathcal{G}$ assuming the operator is a tight Parseval frame [7], $\Pi_\Gamma$ is the projection operator onto $\Gamma$ [8, 10],

$$(\Pi_\Gamma(\mathbf{x}))[t] = \left\{ \begin{array}{ll} \max(x[t], \tau) & (t \in H) \\ y[t] & (t \in R) \\ \min(x[t], -\tau) & (t \in L) \end{array} \right., \qquad (8)$$

and $\text{prox}_{(1/\rho)\mathcal{S}}$ is the proximity operator [22–24]. For example, the proximity operator of $\ell_1$-norm is the soft-thresholding operator,

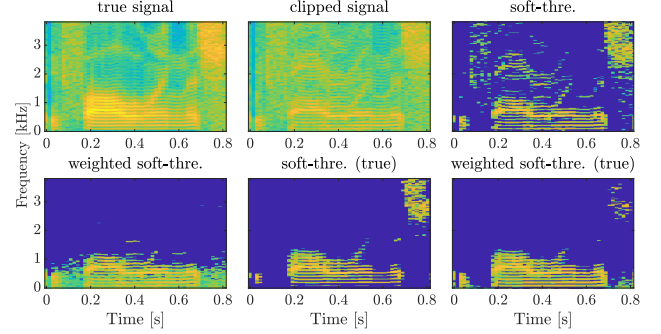$$\left(\mathcal{T}_{(1/\rho)\|\cdot\|_1}(\mathbf{z})\right)[m,n] = \left(1 - \frac{1/\rho}{|z[m,n]|}\right)_+ z[m,n], \qquad (9)$$



**Fig. 2**. Amplitude spectrograms with various processing. $\tau$ was fixed to 0.01, and $\rho$ was adjusted for ease of viewing. The spaectrograms with (true) were processed by the soft-thresholding operator whose denominator was the power spectrogram of the true signal.

where $\mathbf{z} \in \mathbb{C}^{M \times N}$, and $(\cdot)_+ = \max(\cdot, 0)$. In order to adjust bin-wise regularization strength, the weighted $\ell_1$-norm can be introduced. For example, the parabola weight $w[m,n] = (m+1)^2/M^2$ was introduced in [8], and its effectiveness was confirmed. Besides this, some generalized thresholding operators [6, 25–27] and heuristic thresholding [7] have been also utilized for audio declipping.

### 3.2. Inappropriateness of soft-thresholding

The conventional thresholding operators have some disadvantages due to their uniform and data-irrelevant nature. Let us explain it using Fig. 2. The upper left and center figures are the spectrograms of the true and a clipped signal, respectively. Clipping generated some extra harmonic components. The upper right figure is the spectrogram processed by Eq. (9). The extra components were not removed properly. The spectrogram processed by the parabola-weighted soft-thresholding operator [8] is given in the lower left. Although the weights protected the low-frequency part, the high-frequency components were severely removed. These disadvantages are due to the fact that these operators are unrelated to the properties of the data.

If the true signal is available, we can promote sparsity more reasonably. The lower center figure is the spectrogram of the clipped signal processed by the soft-thresholding operator in Eq. (9) whose denominator $|z[m,n]|$ was replaced by the power spectrogram of the true signal $|(\mathcal{G}\tilde{\mathbf{y}})[m,n]|^2$. The figure shows that extra harmonic components were properly removed. The lower right figure was processed by the true-signal-based soft thresholding operator with the parabola weight. The necessary parts remained even in high frequency. Therefore, if the true signals are available, we can properly remove the extra harmonic components by the thresholding operator.

### 3.3. APPLADE: Adjustable Plug-and-PLay Audio DEclipper

Here, we propose APPLADE to utilize the properties of data learned by a DNN. As we show in the previous subsection, the conventional thresholding uniformly promote sparsity, and it can lead to inappropriate results. As one solution to this, we introduced thresholding using the true signal, but of course it is not available. This unrealistic thresholding can be approximated using a DNN $\mathcal{F}_\theta$ that estimates the magnitude spectrogram of the true signal from that of a clipped signal. We use $\mathcal{F}_\theta$ in the weighted soft-thresholding operator as

$$(\mathcal{T}_\theta(\mathbf{z}))[m,n] = \left(1 - \frac{\lambda\, w[m,n]}{((\mathcal{F}_\theta(|\mathbf{z}|))[m,n] + \epsilon)^2}\right)_+ z[m,n], \quad (10)$$

where $\mathbf{w}$ is the parabola weight [8], $\lambda$ is a parameter to adjust the strength of thresholding, and $\epsilon$ is a small constant for numerical sta-

**Algorithm 1** APPLADE
___
1: **Input:** $\mathbf{x}^{[0]}, \mathbf{v}^{[0]}, \mathbf{u}^{[0]}$
2: **Output:** $\mathbf{x}^{[K]}$
3: **for** $k = 0, 1, \ldots, K - 1$ **do**
4:     $\mathbf{x}^{[k+1]} = \Pi_\Gamma(\mathcal{G}^*(\mathbf{v}^{[k]} - \mathbf{u}^{[k]}))$
5:     $\mathbf{v}^{[k+1]} = \mathcal{T}_\theta(\mathcal{G}\mathbf{x}^{[k+1]} + \mathbf{u}^{[k]})$
6:     $\mathbf{u}^{[k+1]} = \mathbf{u}^{[k]} + \mathcal{G}\mathbf{x}^{[k+1]} - \mathbf{v}^{[k+1]}$
7: **end for**
___

bility. We use this operator in place of $\mathrm{prox}_{(1/\rho)\mathcal{S}}$ of the ADMM in Eq. (6), and call this new algorithm APPLADE. The entire algorithm is shown in Alg. 1[1], where $\mathcal{F}_\theta$ is applied repeatedly in the 5th line. In general, careless use of a DNN in an iterative algorithm can be very unstable (as will be demonstrated by experiments). Thus, we propose the operator in Eq. (10) to reduce such undesirable effect.

The proposed method has five advantages. First, $\mathcal{T}_\theta$ can promote sparsity based on the features of data. This leads to more appropriate thresholding for audio declipping than the conventional ones. Second, any DNN can be used as $\mathcal{F}_\theta$. Third, training of the DNN is independent of the iterative algorithm. Fourth, the proposed method inherits adjustability from the sparsity-based methods. Finally, the clipping consistency is considered during processing, i.e., the proposed method does not alter the unclipped elements.

## 4. EXPERIMENTS AND RESULTS

### 4.1. Training

LIBRI speech corpus [28] was utilized for training. We used 5323 clean speech signals (5300 for training, 23 for validation), and 16384 samples of a voiced part were extracted from each signal (about 1 s at sampling frequency of 16 kHz). All data were peak-normalized. They were corrupted by clipping, where $\tau$ was set according to signal-to-distortion ratio (SDR), $\mathrm{SDR}(\mathbf{x}, \mathbf{y}) = 20 \log_{10} \|\mathbf{x}\|_2 / \|\mathbf{x} - \mathbf{y}\|_2$. The input SDR was randomly drawn from the uniform distribution in the interval $[1, 10]$.

The structure of the DNN used as $\mathcal{F}_\theta$ is shown in Fig. 3. The U-Net architecture was designed based on [15, 29]. Since we use $\mathcal{F}_\theta$ iteratively as in Alg. 1, using a too large network will lead to an increase in computation time. Thus, we modified its structure to avoid that. The number of parameters is about one-eleventh of that in TF-UNet [13] (856,033 vs. 9,711,361). The DNN was trained 200 epochs with the adam optimizer [30] with a batch size of 4, a learning rate of 0.0001, and decay rates of $\beta_1 = 0.9$ and $\beta_2 = 0.999$. 75 % overlapped 1024-point-long-Hann window was used for DGT. The input and output features were magnitude spectrograms sized $512 \times 64$ by removing the highest (Nyquist) frequency bin. When the output was used in Eq. (10), we padded zeros so that it has the original size $513 \times 64$. The scale of the leaky ReLU was 0.01. The loss function was the time-domain mean-squared-error (MSE) computed via inverse DGT. The loss was calculated only in clipped parts $H$ and $L$. After the training of DNN, $\lambda$ was determined using the validation data. By curve fitting, $\lambda = 30 p$ was obtained, where $p = (|H| + |L|)/T$ is the ratio of the number of clipped samples.

### 4.2. Testing

For testing, 200 speech signals [31] sampled at 16 kHz from TIMIT corpus were used. Clipping threshold $\tau$ was set according to SDR (1, 3, 5, 10, 15 dB). All data were cut out into 16384 samples and
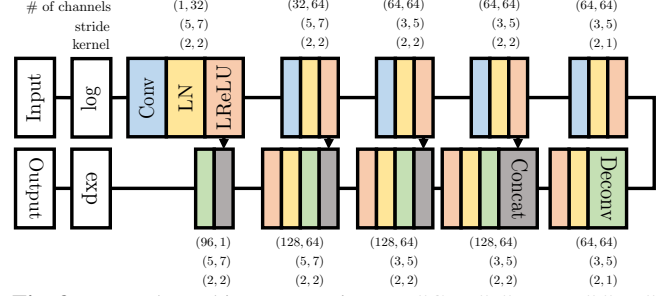
___
[1]Code publicly available at `https://doi.org/hgzf`.



**Fig. 3**. Network used in our experiments. "Conv", "Deconv" "LN", "LReLU", and "Concat" stand for convolution, deconvolution, layer normalization, leaky ReLU, and concatenation, respectively.
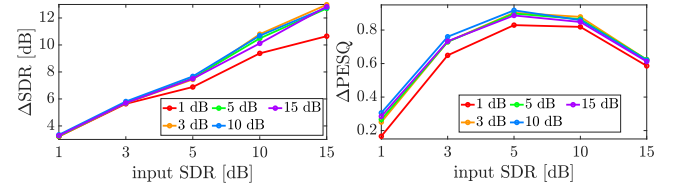


**Fig. 4**. Median of $\Delta$SDR and $\Delta$PESQ of APPLADE when there is the mismatch of clipping level (input SDR) between the training and test data. The legend indicates the input SDR of training data.

peak-normalized. The initial values were set as follows: $\mathbf{x}^{[0]}$ was set to an observed clipped signal, $\mathbf{v}^{[0]} = \mathcal{G}\mathbf{x}^{[0]}$, and $\mathbf{u}^{[0]} = \mathbf{0}$. $\epsilon$ was set to $10^{-6}$, and the number of iteration $K$ was set to 200. We used $\Delta\mathrm{SDR} = \mathrm{SDR}(\hat{\mathbf{y}}, \tilde{\mathbf{x}}) - \mathrm{SDR}(\hat{\mathbf{y}}, \mathbf{y})$ and $\Delta$PESQ, which is improvement of PESQ [32], for evaluation of quality of restored signals.

### 4.3. Experiment 1: Mismatch between training and test data

We conducted an experiment to see how the proposed method behaves for unseen data. During training, all training data were clipped by the same clipping level (input SDR was either 1, 3, 5, 10 or 15 dB). Then, the proposed method was tested for all clipping levels (i.e., 4 out of 5 conditions were unseen). The results are shown in Fig. 4. The mismatch between input SDR during training and testing did not result in a significant performance degradation. Therefore, even for unseen clipping levels, the proposed method will work properly with DNNs trained with an appropriate range of input SDR.

### 4.4. Experiment 2: Effect of the number of DNN parameters

Next, we conducted an experiment to see the effect of the number of DNN parameters on the performance of APPLADE. To naturally change the number of parameters, we used a special DNN only for this experiment. The DNN had the U-Net architecture, which has the same number of blocks for the encoder and decoder. The number of blocks were chosen from 3, 4, 5. Each block was composed of Conv/Deconv (kernel size: $(5, 7)$, stride size: $(2, 2)$), LN, and LReLU, but the last block of the decoder was Deconv only. The number of channels of Conv/Deconv were chosen from 32, 64, 128. Hence, 9 DNNs (determined by the combination of the numbers of blocks and channels) were trained as in Sec. 4.1. To see the effect of the iterative algorithm, the trained DNNs were used *with* and *without* the proposed algorithm.

Fig. 5 shows the results. When using the DNNs without the proposed algorithm (darker colors), the performance of all 9 DNNs was poor. In contrast, APPLADE using them as $\mathcal{F}_\theta$ in Eq. (10) (brighter colors) performed notably better. Interestingly, the performance of

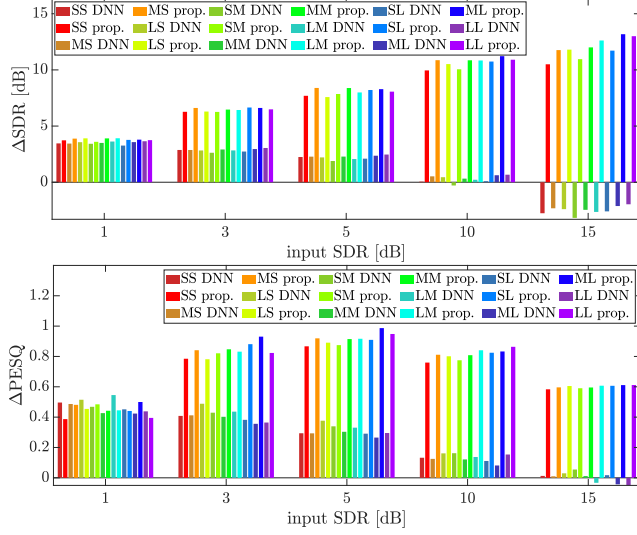**Fig. 5**. Median of $\Delta$SDR and $\Delta$PESQ when the number of DNN parameters was varied. The first letter indicates the number of blocks (S: 3, M: 4, L: 5). The second letter indicates the number of channels of Conv/Deconv (S: 32, M:64, L: 128). "prop." means that the same DNN was inserted and used in the proposed algorithm.

APPLADE was not greatly affected by the number of parameters. Also, the performance of APPLADE seems unrelated to the performance of the DNN itself. These results indicates that the proposed method can absorb the difference of the DNNs to some extent, which is a desirable property because choice of a DNN is not restricted.

### 4.5. Experiment 3: Comparison with other methods

The proposed method was compared with some other methods. As sparsity-based methods, consistent-IHT (IHT) [5], ASPADE [7], Social-Sparsity with Persistent Empirical Wiener (SS PEW) [6], and Parabola-Weighted $\ell_1$ minimization (PW$\ell_1$) [8] were performed. The parameters and other detailed settings were taken from the original paper or the summary paper [10] to suit for the speech signals sampled at 16 kHz. As a DNN-based method, T-UNet [13] $\mathcal{F}_\theta^{\text{T-UNet}}$ approximating $\mathcal{F}^{\text{TD}}(\mathbf{y}) = \tilde{\mathbf{y}}$ was applied. The number of parameters of T-UNet was 11,283,585 [13], and this was about 13 times larger than that of the DNN used in the proposed method (Fig. 3). In addition, the naive PnP method that uses a Gaussain denoising DNN[2] as $\text{prox}_{(1/\rho)S}$ (naive PnP) was also performed.

Fig. 6 shows the results. First of all, let us focus on APPLADE (blue) and PW$\ell_1$ (orange). The difference between them is whether the thresholding operator uses the DNN or not. By effectively using the DNN, $\Delta$SDR was improved by about 2 dB on average. Compared to other sparsity-based methods, the proposed method restored clipped signals best in terms of $\Delta$SDR and as well as ASPADE (red) and SS PEW (green) in terms of $\Delta$PESQ. T-UNet (gray) performed better in low input SDR (i.e., when the clipping was severe), but not in high input SDR. This should be because it cannot consider the clipping consistency, and hence T-UNet does not have information about the position of samples that must be restored. On the contrary, the sparsiy-based methods and APPLADE can take advantage of the clipping consistency by the projection in Eqs. (5) and (8).

To demonstrate the appropriateness of the proposed method, it

---

[2]We changed the output of Fig. 3 to the sigmoid function and trained it to remove 1 to 10 dB of time-domain white Gaussian noise by T-F masking.
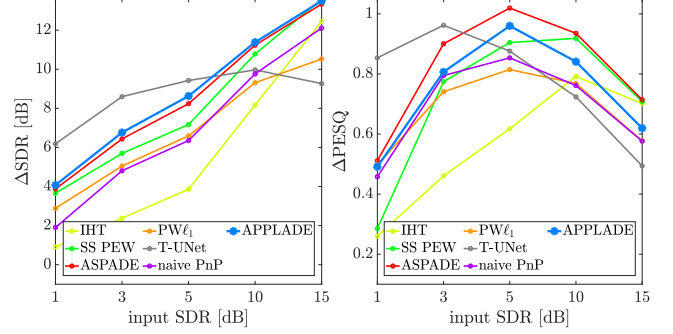


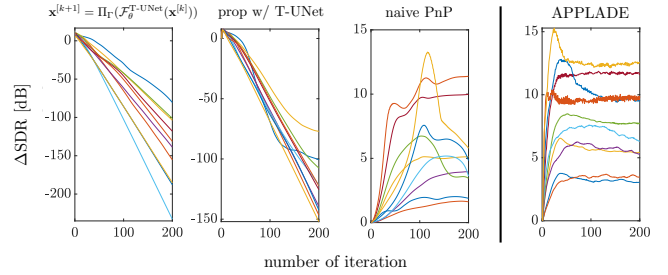**Fig. 6**. Median of $\Delta$SDR and $\Delta$PESQ for each input SDR.



**Fig. 7**. $\Delta$SDR for each iteration. Results for 10 signals are shown.

**Table 1**. Execution time per iteration $t_1$ and execution time $t_2$ necessary for reaching 95 % of $\Delta$SDR in Fig. 6.

|  | IHT* | SS PEW | ASPADE* | PW$\ell_1$ | APPLADE w/ CPU | APPLADE w/ GPU |
|---|---|---|---|---|---|---|
| $t_1$ [s] | 0.031 | 0.037 | 0.030 | 0.037 | 0.025 | 0.023 |
| $t_2$ [s] | 5.2 | 79 | 8.5 | 0.91 | 0.72 | 0.67 |

* Since ASPADE and IHT were performed window-wise, we multiplied $t_1$ and $t_2$ for one window by the number of windows.

was compared with some other combinations of the DNN and an iterative algorithm. Fig. 7 shows $\Delta$SDR of 10 signals for each iteration. Two methods were implemented in addition to naive PnP: a simple iteration of T-UNet and projection, and the ADMM algorithm using T-UNet without the proposed operator in Eq. (10). Although T-UNet itself performs well as in Fig. 6, the two simple methods using T-UNet on the left completely failed in audio declipping. The naive PnP obtained smooth curves, but its performance was unstable. Only APPLADE was able to achieve reasonable results for all signals. These results indicate that the proposed operator in Eq. (10) is essential for successfully perform audio declipping.

Table 1 shows execution time of each method computed by Intel Core i9-10900K (or RTX 2060 for APPLADE w/GPU). Since the proposed method can achieve a good performance with less iterations, the total time ($t_2$) was notably smaller than the other methods.

### 5. CONCLUSIONS

In this paper, we proposed the PnP audio declipping method named APPLADE. A DNN is embedded in the ADMM algorithm and helps in better thresholding for audio declipping. Our experiments showed that APPLADE was robust to variations in the number of DNN parameters and unseen data. Moreover, APPLADE was found to reach higher performance in less time than the other sparsity-based methods and to be more stable than the other possible methods combining a DNN and iteration. Future work will be on theoretical guarantees of convergence and applying to other clipping models.

## 6. REFERENCES

[1] C.-T. Tan and B.C.J. Moore, "Perception of nonlinear distortion by hearing-impaired people," *Int. J. Audiol.*, vol. 47, no. 5, pp. 246–256, 2008.

[2] K.H. Arehart, J.M. Kates, and M.C. Anderson, "Effects of noise, nonlinear processing, and linear filtering on perceived music quality," *Int. J. Audiol.*, vol. 50, no. 3, pp. 177–190, 2011.

[3] J. Malek, "Blind compensation of memoryless nonlinear distortions in sparse signals," in *21st Eur. Signal Process. Conf. (EUSIPCO)*, 2013, pp. 1–5.

[4] Y. Tachioka, T. Narita, and J. Ishii, "Speech recognition performance estimation for clipped speech based on objective measures," *Acoust. Sci. Technol.*, vol. 35, no. 6, pp. 324–326, 2014.

[5] S. Kitic, L. Jacques, N. Madhu, M.P. Hopwood, A. Spriet, and C. De Vleeschouwer, "Consistent iterative hard thresholding for signal declipping," in *IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, 2013, pp. 5939–5943.

[6] K. Siedenburg, M. Kowalski, and M. Dörfler, "Audio declipping with social sparsity," in *IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, 2014, pp. 1577–1581.

[7] P Záviška, P Rajmic, O. Mokrý, and Z. Průša, "A proper version of synthesis-based sparse audio declipper," in *IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, 2019, pp. 591–595.

[8] P. Záviška, P. Rajmic, and J. Schimmel, "Psychoacoustically motivated audio declipping based on weighted $\ell_1$ minimization," in *Int. Conf. Telecommun. Signal Process. (TSP)*, 2019, pp. 338–342.

[9] S. Emura and N. Harada, "An extension of sparse audio declipper to multiple measurement vectors," in *IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, 2021, pp. 686–690.

[10] C. Gaultier, S. Kitić, R. Gribonval, and N. Bertin, "Sparsity-based audio declipping methods: Selected overview, new algorithms, and large-scale evaluation," *IEEE/ACM Trans. Audio Speech Lang. Process.*, vol. 29, pp. 1174–1187, 2021.

[11] F. Bie, D. Wang, J. Wang, and T.F. Zheng, "Detection and reconstruction of clipped speech in speaker recognition," *Speech Commun.*, vol. 72, 07 2015.

[12] W. Mack and E.A.P. Habets, "Declipping speech using deep filtering," in *IEEE Workshop Appl. Signal Process. Audio Acoust. (WASPAA)*, 2019, pp. 200–204.

[13] A.A. Nair and K. Koishida, "Cascaded time + time-frequency Unet for speech enhancement: Jointly addressing clipping, codec distortions, and gaps," in *IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, 2021, pp. 7153–7157.

[14] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[15] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Med. Image Comput. Comput.-Assist. Interv. (MICCAI)*. 2015, pp. 234–241, Springer International Publishing.

[16] S.V. Venkatakrishnan, C.A. Bouman, and B. Wohlberg, "Plug-and-play priors for model based reconstruction," in *IEEE Glob. Conf. Signal Inf. Process.*, 2013, pp. 945–948.

[17] U.S. Kamilov, H. Mansour, and B. Wohlberg, "A plug-and-play priors approach for solving nonlinear imaging inverse problems," *IEEE Signal Process. Lett.*, vol. 24, no. 12, pp. 1872–1876, 2017.

[18] S.H. Chan, X. Wang, and O.A. Elgendy, "Plug-and-play ADMM for image restoration: Fixed-point convergence and applications," *IEEE Trans. Comput. Imaging*, vol. 3, no. 1, pp. 84–98, 2017.

[19] T. Meinhardt, M. Moeller, C. Hazirbas, and D. Cremers, "Learning proximal operators: Using denoising networks for regularizing inverse imaging problems," in *IEEE Int. Conf. Comput. Vis. (ICCV)*, 2017, pp. 1799–1808.

[20] M Terris, A Repetti, J.-C. Pesquet, and Y. Wiaux, "Building firmly nonexpansive convolutional neural networks," in *IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, 2020, pp. 8658–8662.

[21] J.-C. Pesquet, A. Repetti, M. Terris, and Y. Wiaux, "Learning maximally monotone operators for image recovery," *SIAM J. Imaging Sci.*, vol. 14, no. 3, pp. 1206–1237, 2021.

[22] N. Parikh and S. Boyd, *Proximal Algorithms*, Now Publishers Inc., 2014.

[23] N. Komodakis and J.-C. Pesquet, "Playing with duality: An overview of recent primal-dual approaches for solving large-scale optimization problems," *IEEE Signal Process. Mag.*, vol. 32, no. 6, pp. 31–54, 2015.

[24] E. Ryu and S. Boyd, "A primer on monotone operator methods survey," *Appl. comput. math.*, vol. 15, pp. 3–43, 01 2016.

[25] K. Yatabe and D. Kitamura, "Determined BSS based on time-frequency masking and its application to harmonic vector analysis," *IEEE/ACM Trans. Audio Speech Lang. Process.*, vol. 29, pp. 1609–1625, 2021.

[26] M. Kowalski, K. Siedenburg, and M. Dörfler, "Social sparsity! neighborhood systems enrich structured shrinkage operators," *IEEE Trans. Signal Process.*, vol. 61, no. 10, pp. 2498–2511, 2013.

[27] R. Chartrand, "Shrinkage mappings and their induced penalty functions," in *IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, 2014, pp. 1026–1029.

[28] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An ASR corpus based on public domain audio books," in *IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, 2015, pp. 5206–5210.

[29] D. Takeuchi, K. Yatabe, Y. Koizumi, Y. Oikawa, and N. Harada, "Effect of spectrogram resolution on deep-neural-network-based speech enhancement," *Acoust. Sci. Technol.*, vol. 41, no. 5, pp. 769–775, 2020.

[30] D.P. Kingma and J.L. Ba, "Adam: A method for stochastic optimization," in *Proc. IEEE Int. Conf. on Learn. Represent. (ICLR)*, 2015.

[31] P. Mowlaee, J. Kulmer, J. Stahl, and F. Mayer, *Single Channel Phase-Aware Signal Processing in Speech Communication: Theory and Practice*, Hoboken, NJ, USA: Wiley, 11 2016.

[32] A.W. Rix, J.G. Beerends, M.P. Hollier, and A.P. Hekstra, "Perceptual evaluation of speech quality (PESQ)-a new method for speech quality assessment of telephone networks and codecs," in *IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, 2001, vol. 2, pp. 749–752.