AUDIO SIGNAL ENHANCEMENT WITH LEARNING FROM POSITIVE AND UNLABELLED DATA

Nobutaka Ito and Masashi Sugiyama

The University of Tokyo, Japan

ABSTRACT

Supervised learning is a mainstream approach to audio signal enhancement (SE) and requires parallel training data consisting of both noisy signals and the corresponding clean signals. Such data can only be synthesised and are mismatched with real data, which can result in poor performance on real data. Moreover, clean signals may be inaccessible in certain scenarios, which renders this conventional approach infeasible. Here we explore SE using non-parallel training data consisting of noisy signals and noise, which can be easily recorded. We define the positive (P) and the negative (N) classes as signal inactivity and activity, respectively. We observe that the spectrogram patches of noise clips can be used as P data and those of noisy signal clips as unlabelled data. Thus, *learning from positive and unlabelled data* enables a convolutional neural network to learn to classify each spectrogram patch as P or N to enable SE.

Index Terms— Audio signal enhancement, convolutional neural networks, learning from positive and unlabelled data, weakly supervised learning.

1. INTRODUCTION

Audio signal enhancement. This paper deals with audio signal enhancement (SE),[1] the task of extracting a specific class of sounds (called a *signal*) while suppressing the other classes of sounds (called *noise*) from the mixture of them (called a *noisy signal*). Applications of SE include automatic speech recognition (ASR), music information retrieval, and sound event detection. In this paper, we focus on single-channel SE, which applies even in situations where only one microphone is available. See, e.g., [1] for a multichannel SE method.

Supervised learning has become a mainstream approach to SE, where an SE model such as a deep neural network (DNN) is trained to predict the clean (i.e., noise-free) signal or a mask for SE [2, 3, 4]. The approach requires parallel training data consisting of both noisy signals and the corresponding clean signals. As it is impossible to record such parallel data due to the crosstalk, a common practice is to synthesise them. Fig. 1(a) illustrates the data synthesis pipeline using speech enhancement as an example. First, clean signals (e.g., clean speech utterances) are recorded in a quiet environment such as a recording studio. Then, these signals are convolved with room impulse responses, which are simulated by an acoustic simulator such as the image method [5]. Finally, noisy signals are synthesised by numerically adding a clean signal and noise.

Issues. There are fundamental issues in the above data collection method for supervised learning. First, the synthetic training data are inevitably mismatched with the distribution of real data that we wish to apply SE to. This is because it is difficult to realistically simulate the characteristics of the sound source, the room sound field, and the recording device. This can result in poor SE performance on real data, if the mismatch is severe. Second, clean signals may be inaccessible in certain scenarios, which renders this conventional
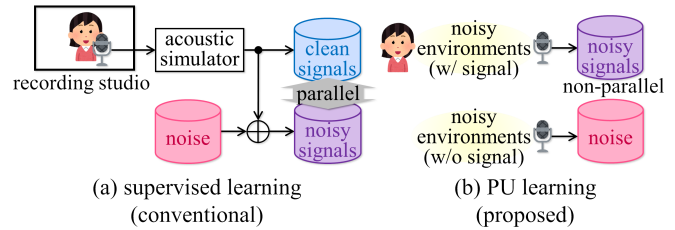

Fig. 1. Training data collection methods in conventional & proposed approaches to SE.

approach infeasible. For instance, clean signals may be inaccessible in the enhancement of music signals [6] or certain sound events like bird songs [7, 8].

Motivation. The above issues motivate us to explore SE using non-parallel (i.e., unpaired) training data consisting of noisy signals and noise (see Fig. 1(b)). Note that noisy signals and noise can be easily recorded in noisy environments, when the signal is active and inactive, respectively.

Learning from positive and unlabelled data for audio signal enhancement. Here we propose *PULSE (PU learning for SE)*, an SE method using non-parallel training data consisting of noisy signals and noise. PULSE is based on *learning from positive and unlabelled data (PU learning)* [9, 10, 11, 12], a methodology for weakly supervised learning from only positive (P) and unlabelled (U) data without negative (N) data. Let us define the P and the N classes as signal inactivity and activity, respectively.[2] The spectrogram patches of noise clips do not contain the signal by definition and thus can be used as P data. On the other hand, those of noisy signal clips can be either P or N, depending on whether the signal is active or not in those spectrogram patches. Thus, they are used as U data.[3] These P and U data enable a convolutional neural network (CNN) to learn to classify each spectrogram patch as P or N through PU learning. The trained CNN enables us to obtain a time–frequency (TF) dependent weight called a mask, which enables TF components dominated by noise to be suppressed for SE. We also propose a *weighted sigmoid loss*, a new loss appropriate for SE, which uses the magnitude spectrogram as a weight in the sigmoid loss [11]. The proposed loss turns out to be closely related to the signal approximation loss [13] and empirically gave a substantially better speech enhancement performance than the original unweighted sigmoid loss [11]. To the best of our knowledge, PULSE is the first application of PU learning to SE.

---

[1] We use the term "SE" in the sense of "noise suppression".

[2] We could define the P class as signal activity but we do it the other way around to be consistent with PU learning.

[3] For a noisy signal clip, we know that there exists at least one spectrogram patch containing the signal. Here we assume this constraint is satisfied.

2. RELATION TO PRIOR WORK

Parallel training data without clean signals. Like PULSE, *mixture invariant training (MixIT)* [14][4] uses noisy signals and noise for training. During training, the sum of a noisy signal and noise is given as an input and a DNN has to separate it into an enhanced signal (i.e., an estimate of the clean signal) and two noise estimates. To enable training without ground truth clean signals, MixIT assumes that the sum of the enhanced signal and one of the noise estimates approximates the original noisy signal and the other noise estimate approximates the original noise. This approximation error is used as the loss, which can be computed without clean signals. Note that there are two possibilities as to which of the noise estimates corresponds to the original noisy signal and it is unknown which is correct. To resolve this ambiguity, a mixture invariant loss similar to the permutation invariant loss [15] is used. Unlike PULSE, however, MixIT requires parallel training data, where each training example is a triplet consisting of a noisy signal, noise, and their sum. As such parallel data cannot be recorded but can only be synthesised (i.e., the noisy signal and the noise are numerically added), *MixIT also suffers from the above mismatch issue.* For example, there may be mismatches in terms of the signal-to-noise ratio (SNR), the number of noise sources, etc. A similar method was proposed in [16] and has the same mismatch issue.

Non-parallel training data with clean signals. There are some methods that do not require parallel training data unlike many SE methods. Such methods include generative modeling of clean signals [17, 18, 19] and adversarial training of an SE network [20]. Unlike PULSE, these methods require clean signals for training and thus also suffer from the above issue of clean signals being often inaccessible.

Unsupervised learning. Some SE methods require no supervision or even no training data at all based on unsupervised learning. However, these methods require strong inductive biases based on domain knowledge (e.g., stationary noise) [21, 22] and are inapplicable when the assumptions made are violated.

3. PRELIMINARIES: PU LEARNING

This section provides preliminaries on PU learning [9, 10, 11, 12]. Let $\mathcal{X}^{\mathrm{P}} := \{\mathbf{x}_i^{\mathrm{P}}\}_{i=1}^{n^{\mathrm{P}}}$ be P data and $\mathcal{X}^{\mathrm{U}} := \{\mathbf{x}_i^{\mathrm{U}}\}_{i=1}^{n^{\mathrm{U}}}$ be U data, where $n^{\mathrm{P}} := |\mathcal{X}^{\mathrm{P}}|$ and $n^{\mathrm{U}} := |\mathcal{X}^{\mathrm{U}}|$ denote the number of P and U training examples, respectively. From these P and U data, a classifier has to learn the relationship between an input pattern $\mathbf{x} \in \mathbb{R}^d$ and a class label $y \in \{+1, -1\}$ so that it can predict the correct class label for an unseen pattern. Here, d is the feature dimension. Let us assume that the P data $\mathcal{X}^{\mathrm{P}}$ are independent and identically distributed (i.i.d.) and follow an unknown class-conditional density $p(\mathbf{x} \,|\, y = +1)$. Let us also assume that the U data $\mathcal{X}^{\mathrm{U}}$ are i.i.d. and follow an unknown marginal density $p(\mathbf{x})$.

Let $f_{\boldsymbol{\theta}} : \mathbb{R}^d \to \mathbb{R}$ be a classifier parametrised by $\boldsymbol{\theta}$ and $\widehat{y} := \mathrm{sgn}(f_{\boldsymbol{\theta}}(\mathbf{x}))$ be the predicted label for $\mathbf{x}$, where sgn is the sign function

$$\mathrm{sgn}(t) := \begin{cases} +1 & (t \geq 0), \\ -1 & (t < 0). \end{cases} \tag{1}$$

Let us define a *loss* $\ell(\mathbf{x}, y, \boldsymbol{\theta})$, a non-negative function that measures the deviation of the classifier $f_{\boldsymbol{\theta}}$ from a data point $(\mathbf{x}, y)$. Note that $\ell(\mathbf{x}, y, \boldsymbol{\theta})$ can be any non-negative loss without any other constraints. An example is the sigmoid loss [11]

$$\ell(\mathbf{x}, y, \boldsymbol{\theta}) = \sigma(-y f_{\boldsymbol{\theta}}(\mathbf{x})), \tag{2}$$

---

where $\sigma(m) := (1 + e^{-m})^{-1}$ is the logistic sigmoid function. It is worth noting that (2) is bounded unlike many other common losses (e.g., the cross-entropy). Let us define the *risk*, also known as (a.k.a.) the *generalisation error*, as

$$R(\boldsymbol{\theta}) := \mathbb{E}_{p(\mathbf{x}, y)}[\ell(\mathbf{x}, y, \boldsymbol{\theta})], \tag{3}$$

which is the expectation of the loss with respect to (w.r.t.) an unknown joint data distribution $p(\mathbf{x}, y)$. As $p(\mathbf{x}, y)$ is unknown, the expectation in (3) cannot be computed in practice. Therefore, we compute an *empirical risk* (a.k.a. a *training error*) $\widehat{R}(\boldsymbol{\theta}) \approx R(\boldsymbol{\theta})$ by replacing expectation by empirical averaging over training data and obtain $\boldsymbol{\theta}$ by solving the minimisation of $\widehat{R}(\boldsymbol{\theta})$. This formulation is called *empirical risk minimisation*.

In supervised learning, an empirical risk is easily obtained as $R(\boldsymbol{\theta}) \approx \frac{1}{n} \sum_{i=1}^{n} \ell(\mathbf{x}_i, y_i, \boldsymbol{\theta})$, where $\{(\mathbf{x}_i, y_i)\}_{i=1}^{n} \sim p(\mathbf{x}, y)$ are labelled training data. In contrast, in PU learning, only P and U data are given as training data without N data. It may appear to be impossible to compute an empirical risk using only such data, but this turns out to be possible as follows.

Note first that

$$p(\mathbf{x}, y) = \begin{cases} \pi p(\mathbf{x} \,|\, y = +1) & (y = +1), \\ (1 - \pi) p(\mathbf{x} \,|\, y = -1) & (y = -1), \end{cases} \tag{4}$$

where $\pi := p(y = +1)$ is the class prior for the P class and assumed to be given in this paper. See [12, 23, 24, 25] for some methods for estimating π from only P and U data. Using (4), (3) is rewritten as

$$R(\boldsymbol{\theta}) = \pi \mathbb{E}_{p(\mathbf{x}|y=+1)}[\ell(\mathbf{x}, +1, \boldsymbol{\theta})] \tag{5}$$
$$+ (1 - \pi) \mathbb{E}_{p(\mathbf{x}|y=-1)}[\ell(\mathbf{x}, -1, \boldsymbol{\theta})].$$

Since $p(\mathbf{x}) = \pi p(\mathbf{x} \,|\, y = +1) + (1 - \pi) p(\mathbf{x} \,|\, y = -1)$, (5) is further rewritten as

$$R(\boldsymbol{\theta}) = \pi \mathbb{E}_{p(\mathbf{x}|y=+1)}[\ell(\mathbf{x}, +1, \boldsymbol{\theta})] \tag{6}$$
$$+ \mathbb{E}_{p(\mathbf{x})}[\ell(\mathbf{x}, -1, \boldsymbol{\theta})] - \pi \mathbb{E}_{p(\mathbf{x}|y=+1)}[\ell(\mathbf{x}, -1, \boldsymbol{\theta})].$$

Therefore, we obtain an empirical risk using only P and U data, $\widehat{R}_{\mathrm{PU}}(\boldsymbol{\theta}) \approx R(\boldsymbol{\theta})$, as follows [10]:

$$\widehat{R}_{\mathrm{PU}}(\boldsymbol{\theta}) := \frac{\pi}{|\mathcal{X}^{\mathrm{P}}|} \sum_{\mathbf{x} \in \mathcal{X}^{\mathrm{P}}} \ell(\mathbf{x}, +1, \boldsymbol{\theta}) \tag{7}$$
$$+ \frac{1}{|\mathcal{X}^{\mathrm{U}}|} \sum_{\mathbf{x} \in \mathcal{X}^{\mathrm{U}}} \ell(\mathbf{x}, -1, \boldsymbol{\theta}) - \frac{\pi}{|\mathcal{X}^{\mathrm{P}}|} \sum_{\mathbf{x} \in \mathcal{X}^{\mathrm{P}}} \ell(\mathbf{x}, -1, \boldsymbol{\theta}).$$

While $\mathbb{E}_{p(\mathbf{x})}[\ell(\mathbf{x}, -1, \boldsymbol{\theta})] - \pi \mathbb{E}_{p(\mathbf{x}|y=+1)}[\ell(\mathbf{x}, -1, \boldsymbol{\theta})]$ in (6) is non-negative for a non-negative loss ℓ, its empirical approximation in (7) can be negative. This is especially significant for an unbounded loss (e.g., the cross-entropy) or a flexible model (e.g., a DNN). To remedy this, the following non-negative empirical risk $\widehat{R}_{\mathrm{nnPU}}(\boldsymbol{\theta}) \approx R(\boldsymbol{\theta})$ was proposed [11]:

$$\widehat{R}_{\mathrm{nnPU}}(\boldsymbol{\theta}) := \frac{\pi}{|\mathcal{X}^{\mathrm{P}}|} \sum_{\mathbf{x} \in \mathcal{X}^{\mathrm{P}}} \ell(\mathbf{x}, +1, \boldsymbol{\theta}) + \left(\frac{1}{|\mathcal{X}^{\mathrm{U}}|} \sum_{\mathbf{x} \in \mathcal{X}^{\mathrm{U}}} \ell(\mathbf{x}, -1, \boldsymbol{\theta}) \right.$$
$$\left. - \frac{\pi}{|\mathcal{X}^{\mathrm{P}}|} \sum_{\mathbf{x} \in \mathcal{X}^{\mathrm{P}}} \ell(\mathbf{x}, -1, \boldsymbol{\theta}) \right)_{+}, \tag{8}$$

where 'nn' stands for 'non-negative' and

$$(x)_{+} := \begin{cases} x, & (x \geq 0), \\ 0, & (x < 0). \end{cases} \tag{9}$$
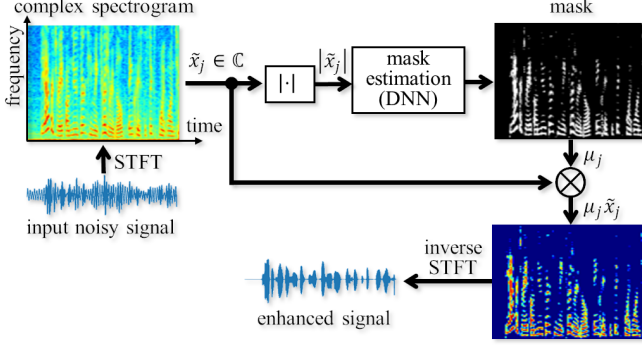
Fig. 2. Pipeline of masking-based SE.

4. PULSE: PU LEARNING FOR SE

Masking-based SE. In this paper, we employ a masking-based approach to SE, where we use a DNN to estimate a mask in the short-time Fourier transform (STFT) domain (Fig. 2). In this approach, an input noisy signal clip in the time domain is first transformed into the TF domain by the STFT. Let the resulting STFT-domain representation (i.e., the complex spectrogram) be $\widetilde{x}_j \in \mathbb{C}$, where j is the TF component index. Then, we compute a magnitude spectrogram $|\widetilde{x}_j|$ by taking the absolute value. A DNN is given the magnitude spectrogram and estimates a mask μ_j. An enhanced signal is obtained by elementwise multiplication (i.e., masking) $\mu_j \widetilde{x}_j$, which suppresses TF components dominated by noise. Finally, the enhanced signal is transformed back into the time domain by the inverse STFT.

Motivation. In this approach, it is crucial to train the DNN so that the mask can be estimated properly. If we were given parallel training data consisting of both noisy signals and the corresponding clean signals, we could do so by supervised learning in a straightforward way. However, as we already mentioned in Sec. 1, such a training methodology has fundamental issues, which motivated us to develop PULSE.

Pipeline of PULSE is illustrated in Fig. 3. The training data consist of noisy signal clips and noise clips. We first compute a magnitude spectrogram of each clip by applying the STFT and then taking the absolute value. Then, we crop a rectangular spectrogram patch centred at each TF point as the input feature. Let us define a TF component as P and N if the signal is inactive and active in the corresponding spectrogram patch, respectively. Each TF component of a noise clip is P. On the other hand, each TF component of a noisy signal clip can be either P or N and is thus treated as U. Thus, $\mathcal{X}^{\mathrm{P}}$ and $\mathcal{X}^{\mathrm{U}}$ consist of the spectrogram patches of noise clips and those of noisy signal clips, respectively. These P and U data are used to train a CNN to classify each TF component as P or N by PU learning described in Sec. 3. During testing, the mask μ_j is obtained by

$$\mu_j \leftarrow \begin{cases} 1 & (\widehat{y}_j = -1), \\ 0 & (\widehat{y}_j = +1). \end{cases} \tag{10}$$

Here, $\widehat{y}_j := \mathrm{sgn}(f_{\widehat{\theta}}(\mathbf{x}_j))$ is the predicted label of the jth TF component, where $\mathbf{x}_j$ is the corresponding spectrogram patch and $\widehat{\theta}$ is the trained parameters. This mask retains the TF components classified as N and removes those classified as P.

Architecture. The classifier f_{θ} is modelled by the following 11-layer CNN: Compress(1/15)-Conv2d(1, 8, 3)-Conv2d(8, 8, 3)-Conv2d(8, 16, 3)-Conv2d(16, 16, 3)-Conv2d(16, 32, 3)-Conv2d(32, 32, 3)-Conv2d(32, 64, 3)-Conv2d(64, 64, 3)-Conv2d(64, 128, 1)-Conv2d(128, 128, 1)-Conv2d(128, 1, 1). Here, Compress(α) is a power-law compression layer that applies the non-linear function
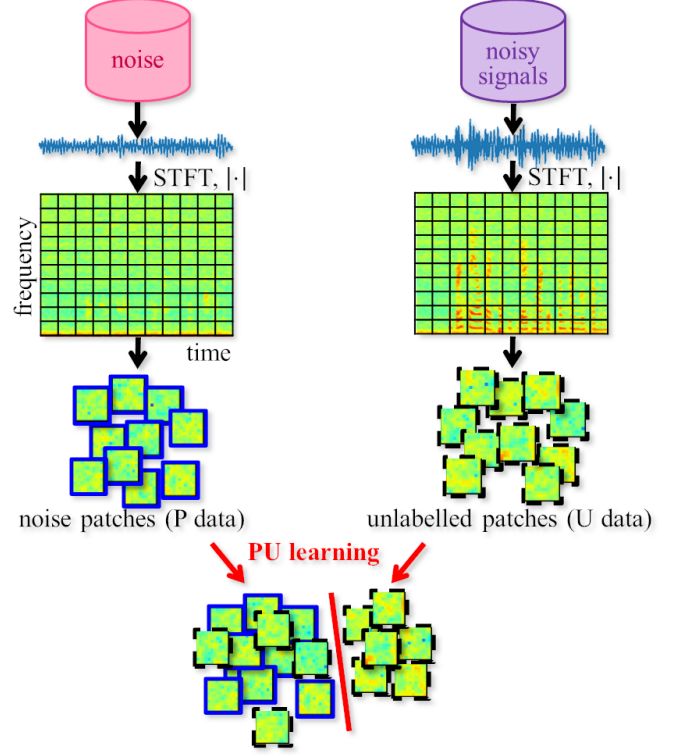

Fig. 3. Pipeline of PULSE.

$(\cdot)^{\alpha}$ elementwise. Conv2d($C_{\mathrm{in}}, C_{\mathrm{out}}, K$) is a two-dimensional convolutional layer with C_{in} input channels, C_{out} output channels, a kernel size of $K \times K$, a stride of $(1, 1)$, and no padding. All but the last convolutional layer are followed by a rectified linear unit (ReLU) and then a dropout layer with a dropout rate of 0.2. The size of the input spectrogram patch is set to that of the receptive field of the network (i.e., 17×17) so that the CNN output size is 1×1.

Loss. It is crucial to design the loss $\ell(\mathbf{x}, y, \theta)$ properly to obtain a good SE performance by PULSE. It measures the deviation of the classifier f_{θ} from $(\mathbf{x}, y)$, where $\mathbf{x}$ is a spectrogram patch and y is the corresponding label. Commonly used losses in PU learning, such as the sigmoid loss (2) or the cross-entropy, assign uniform weights to all TF components. As the classification accuracy of the TF component with a larger magnitude is more significant in SE, we introduce the magnitude spectrogram $w(\mathbf{x}) := |\widetilde{x}|$ as a weight in (2). Specifically, our loss is given by

$$\ell(\mathbf{x}, y, \theta) = w(\mathbf{x})\sigma(-yf_{\theta}(\mathbf{x})), \tag{11}$$

which we call a *weighted sigmoid loss*.[5]

---

[5]To give alternative interpretations of (11), note that it can be rewritten as $\ell(\mathbf{x}, y, \theta) = |\frac{1}{2}(y + 1)\widetilde{x} - (\sigma \circ f_{\theta})(\mathbf{x})\widetilde{x}|$, where $\circ$ is function composition. Here, $\frac{1}{2}(y + 1)$ can be interpreted as a ground truth binary mask for extracting noise as it retains the P TF components and removes the N TF components. On the other hand, $(\sigma \circ f_{\theta})(\mathbf{x})$ can be regarded as an estimated soft mask for extracting noise as it approaches 1 when $f_{\theta}(\mathbf{x})$ approaches $+\infty$ and approaches 0 when $f_{\theta}(\mathbf{x})$ approaches $-\infty$. Therefore, our loss is the absolute error between a noise estimate $\frac{1}{2}(y + 1)\widetilde{x}$ using the ground truth binary mask $\frac{1}{2}(y + 1)$ and a noise estimate $(\sigma \circ f_{\theta})(\mathbf{x})\widetilde{x}$ using the estimated soft mask $(\sigma \circ f_{\theta})(\mathbf{x})$. Our loss can also be seen as the signal approximation loss [13] with the following modifications. First, our loss is defined w.r.t. noise instead of the signal. Second, the target in our loss is the noise estimate using the ground truth mask, $\frac{1}{2}(y+1)\widetilde{x}$, instead of the ground truth noise itself. Third, our loss uses the 1-norm instead of the 2-norm.

Clip-wise processing. The above patch-wise processing is computationally inefficient as it repeats some operations common to neighbouring spectrogram patches. A more efficient clip-wise processing consists in simply applying the above CNN to entire magnitude spectrograms instead of spectrogram patches, similar to [26, 27]. In this case, the 'same' padding is applied to each convolutional layer to preserve the dimensions of the spectrogram.

5. SPEECH ENHANCEMENT EXPERIMENT

We conducted a preliminary experiment of speech enhancement using PyTorch as a proof of concept to confirm the feasibility of SE using non-parallel data through PULSE.[6]

Methods. We compared the following methods:

- **PULSE.** The 11-layer CNN (clip-wise processing) in Sec. 4 was trained by PU learning with the non-negative empirical risk (8) (specifically Algorithm 1 of [11]). The weighted sigmoid loss (11) was used. The class prior was set to $\pi = 0.7$, which was tuned on the validation set. During testing, a binary mask was obtained by (10).

- **Supervised learning.** The same CNN architecture was used except that the kernel size was 3×3 in all convolutional layers. The CNN was trained by supervised learning with the signal approximation loss in the STFT domain [13]. A soft mask was obtained by applying the sigmoid activation to the CNN output.

- **MixIT [14]:** The same architecture as in **supervised learning** was used except that there were three output channels corresponding to the enhanced signal and two noise estimates. The CNN was trained by solving the minimisation of a mixture invariant loss (see [14] for details) based on the signal approximation loss in the STFT domain [13]. Soft masks for the enhanced signal and two noise estimates were obtained by applying the sigmoid activation to the CNN output.

In all methods, the frame length and the hop for the STFT were set to 1024 samples (64 ms) and 256 samples (16 ms), respectively, and the hamming window was used.

Data. We focused on synthetic data. This is because most evaluation metrics for speech enhancement performance, including the scale-invariant SNR (SI-SNR) [28], require parallel data, which can only be synthesised. We will conduct an evaluation on real data w.r.t. the ASR accuracy or a non-intrusive metric, such as DNSMOS [29], in future work. We prepared a speech enhancement dataset using speech from TIMIT [30] and noise from DEMAND [31]. We used the training set of TIMIT to create our training and validation sets and the test set of TIMIT to create our test set. We used noise recordings from DEMAND in the following environments, which we found contained little speech: DKITCHEN, DLIVING, DWASHING, NFIELD, NRIVER, OHALLWAY, OOFFICE, STRAFFIC, and TCAR. Each noise recording was divided into halves, one for the training and the validation sets and the other for the test set. The training set for **PULSE** consisted of 4019 noisy speech clips and 4019 noise clips (3.49 h each). Throughout this experiment, all clips were 3.125 s long and sampled at 16 kHz. The training set for **supervised learning** consisted of 4019 noisy speech clips along with the corresponding clean speech clips. The training set for **MixIT** consisted of 4019 noisy speech clips, 4019 noise clips, and the corresponding 4019 mixture (i.e., noisy speech plus noise) clips. In all methods, the validation/test set consisted

of 601/1680 noisy speech clips (0.52 h/1.46 h) along with the corresponding clean speech clips. Each noise clip above was a random excerpt from DEMAND; Each noisy speech clip was generated by adding a TIMIT clip and a random excerpt from DEMAND at an SNR sampled uniformly from the interval $[-5, 10]$ dB.

Training. Data-parallel distributed training was performed on 16 NVIDIA A100 GPUs for 400 epochs with the Adam optimiser [32], a batch size per GPU of 16, and a learning rate of 0.0018/0.0032/0.00055 (**PULSE/supervised learning/MixIT**).

Metric. We used the SI-SNR [28] as the evaluation metric for speech enhancement performance. Let $\mathbf{s}$ be the clean speech and $\widehat{\mathbf{s}}$ be an estimate of it, both in the time domain. Note that $\widehat{\mathbf{s}}$ can be decomposed as $\widehat{\mathbf{s}} = \widehat{\mathbf{s}}_{\parallel} + \widehat{\mathbf{s}}_{\perp}$. Here, $\widehat{\mathbf{s}}_{\parallel} := \frac{\mathbf{s}^T \widehat{\mathbf{s}}}{\|\mathbf{s}\|^2} \mathbf{s}$ is the component parallel to $\mathbf{s}$ and $\widehat{\mathbf{s}}_{\perp} := \widehat{\mathbf{s}} - \widehat{\mathbf{s}}_{\parallel}$ is that perpendicular to it, where $(\cdot)^T$ is transposition and $\|\cdot\|$ is the 2-norm. The SI-SNR is defined using the ratio of the squared norms of these components as follows:

$$\text{SI-SNR}(\mathbf{s}, \widehat{\mathbf{s}}) := 10 \log_{10} \frac{\|\widehat{\mathbf{s}}_{\parallel}\|^2}{\|\widehat{\mathbf{s}}_{\perp}\|^2} = 10 \log_{10} \frac{\|a\mathbf{s}\|^2}{\|\widehat{\mathbf{s}} - a\mathbf{s}\|^2} \quad (12)$$

with $a := \frac{\mathbf{s}^T \widehat{\mathbf{s}}}{\|\mathbf{s}\|^2}$. It is convenient to define an SI-SNR improvement (SI-SNRi) as the difference of SI-SNR$(\mathbf{s}, \widehat{\mathbf{s}})$ with $\widehat{\mathbf{s}}$ being the enhanced speech and SI-SNR$(\mathbf{s}, \widehat{\mathbf{s}})$ with $\widehat{\mathbf{s}}$ being the observed noisy speech. Not only did we use the test SI-SNRi for performance evaluation, we also tuned the hyperparameters based on the validation SI-SNRi. A model checkpoint was saved at the end of each epoch and the one with the highest validation SI-SNRi was selected for evaluation on the test set.

Results. Table 1 shows the SI-SNRi on the test set. By using only non-parallel training data, which can be easily recorded, **PULSE** was able to give an SI-SNRi of as much as 14.62 dB. This score was superior to 12.19 dB for **MixIT** and close to 15.86 dB for **supervised learning**. For ablation, we also evaluated PULSE with the original sigmoid loss (2) instead of the weighted one (11) and PULSE with the original empirical risk (7) instead of the non-negative one (8). Without (11) and (8), the SI-SNRi dropped to 9.30 dB and 12.96 dB, respectively, which shows the significance of the loss weighting and the non-negative empirical risk.

Table 1. SI-SNRi on the test set. The mean over five trials is shown along with the standard deviation in the parentheses.

method	SI-SNRi (dB)
PULSE	14.62 (0.20)
supervised learning	15.86 (1.28)
MixIT [14]	12.19 (4.50)
PULSE w/o (11)	9.30 (0.70)
PULSE w/o (8)	12.96 (3.19)

6. CONCLUSIONS

We proposed PULSE, a PU-learning-based paradigm for training SE models on non-parallel data consisting of noisy signals and noise. The feasibility of PULSE was confirmed through a speech enhancement experiment. The future work shall include exploring more sophisticated architectures, conducting more extensive experiments, and applying PU learning to other audio tasks.

---

[6]The source code is available at `https://github.com/nobutaka-ito/pulse`.

7. REFERENCES

[1] Z.-Q. Wang, P. Wang, and D. L. Wang, "Complex spectral mapping for single- and multi-channel speech enhancement and robust ASR," *IEEE/ACM Trans. Audio, Speech, Language Process.*, vol. 28, pp. 1778–1787, 2020.

[2] H. Erdogan, J. R. Hershey, S. Watanabe, and J. Le Roux, "Phase-sensitive and recognition-boosted speech separation using deep recurrent neural networks," in *Proc. ICASSP*, 2015, pp. 708–712.

[3] K. Tan and D. L. Wang, "Learning complex spectral mapping with gated convolutional recurrent networks for monaural speech enhancement," *IEEE/ACM Trans. Audio, Speech, Language Process.*, vol. 28, pp. 380–390, 2020.

[4] Y. Hu, Y. Liu, S. Lv, M. Xing, S. Zhang, Y. Fu, J. Wu, B. Zhang, and L. Xie, "DCCRN: Deep complex convolution recurrent network for phase-aware speech enhancement," in *Proc. Interspeech*, 2020, pp. 2472–2476.

[5] J. B. Allen and D. A. Berkley, "Image method for efficiently simulating small-room acoustics," *J. Acoust. Soc. Amer.*, vol. 65, no. 4, pp. 943–950, 1979.

[6] D. Stoller, S. Ewert, and S. Dixon, "Adversarial semi-supervised audio source separation applied to singing voice extraction," in *Proc. ICASSP*, 2018, pp. 2391–2395.

[7] D. Stowell and R. E. Turner, "Denoising without access to clean data using a partitioned autoencoder," 2015, arXiv:1509.05982.

[8] T. Denton, S. Wisdom, and J. R. Hershey, "Improving bird classification with unsupervised sound separation," in *Proc. ICASSP*, 2022, pp. 636–640.

[9] M. C. du Plessis, G. Niu, and M. Sugiyama, "Analysis of learning from positive and unlabeled data," in *Proc. NIPS*, 2014, pp. 703–711.

[10] M. C. du Plessis, G. Niu, and M. Sugiyama, "Convex formulation for learning from positive and unlabeled data," in *Proc. ICML*, 2015, pp. 1386–1394.

[11] R. Kiryo, G. Niu, M. C. du Plessis, and M. Sugiyama, "Positive-unlabeled learning with non-negative risk estimator," in *Proc. NIPS*, 2017, pp. 1674–1684.

[12] M. Sugiyama, H. Bao, T. Ishida, N. Lu, T. Sakai, and G. Niu, *Machine Learning from Weak Supervision: An Empirical Risk Minimization Approach.* Cambridge, MA: MIT Press, 2022.

[13] D. L. Wang and J. Chen, "Supervised speech separation based on deep learning: An overview," *IEEE/ACM Trans. Audio, Speech, Language Process.*, vol. 26, no. 10, pp. 1702–1726, 2018.

[14] S. Wisdom, E. Tzinis, H. Erdogan, R. Weiss, K. Wilson, and J. Hershey, "Unsupervised sound separation using mixture invariant training," in *Proc. NeurIPS*, 2020, pp. 3846–3857.

[15] M. Kolbæk, D. Yu, Z.-H. Tan, and J. Jensen, "Multitalker speech separation with utterance-level permutation invariant training of deep recurrent neural networks," *IEEE/ACM Trans. Audio, Speech, Language Process.*, vol. 25, no. 10, pp. 1901–1913, 2017.

[16] T. Fujimura, Y. Koizumi, K. Yatabe, and R. Miyazaki, "Noisy-target training: A training strategy for DNN-based speech enhancement without clean speech," in *Proc. EUSIPCO*, 2021, pp. 436–440.

[17] P. Smaragdis, B. Raj, and M. Shashanka, "Supervised and semi-supervised separation of sounds from single-channel mixtures," in *Proc. ICA*, 2007, pp. 414–421.

[18] Y. Bando, M. Mimura, K. Itoyama, K. Yoshii, and T. Kawahara, "Statistical speech enhancement based on probabilistic integration of variational autoencoder and non-negative matrix factorization," in *Proc. ICASSP*, 2018, pp. 716–720.

[19] Y. C. Subakan and P. Smaragdis, "Generative adversarial source separation," in *Proc. ICASSP*, 2018, pp. 26–30.

[20] T. Higuchi, K. Kinoshita, M. Delcroix, and T. Nakatani, "Adversarial training for data-driven speech enhancement without parallel corpus," in *Proc. ASRU*, 2017, pp. 40–47.

[21] S. F. Boll, "Suppression of acoustic noise in speech using spectral subtraction," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-27, no. 2, pp. 113–120, 1979.

[22] N. Mohammadiha, P. Smaragdis, and A. Leijon, "Supervised and unsupervised speech enhancement using nonnegative matrix factorization," *IEEE Trans. Audio, Speech, Language Process.*, vol. 21, no. 10, pp. 2140–2151, 2013.

[23] S. Jain, M. White, and P. Radivojac, "Estimating the class prior and posterior from noisy positives and unlabeled data," in *Proc. NIPS*, 2016.

[24] M. C. du Plessis, G. Niu, and M. Sugiyama, "Class-prior estimation for learning from positive and unlabeled data," *Mach. Learn.*, vol. 106, no. 4, pp. 463–492, 2017.

[25] Y. Yao, T. Liu, B. Han, M. Gong, G. Niu, M. Sugiyama, and D. Tao, "Rethinking class-prior estimation for positive-unlabeled learning," in *Proc. ICLR*, 2022.

[26] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," in *Proc. ICLR*, 2014.

[27] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. CVPR*, 2016, pp. 779–788.

[28] J. Le Roux, S. Wisdom, H. Erdogan, and J. R. Hershey, "SDR—half-baked or well done?" in *Proc. ICASSP*, 2019, pp. 626–630.

[29] C. K. A. Reddy, V. Gopal, and R. Cutler, "DNSMOS: A non-intrusive perceptual objective speech quality metric to evaluate noise suppressors," in *Proc. ICASSP*, 2021, pp. 6493–6497.

[30] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, D. S. Pallett, N. L. Dahlgren, and V. Zue, *TIMIT Acoustic-Phonetic Continuous Speech Corpus LDC93S1. Web Download.* Philadelphia: Linguistic Data Consortium, 1993.

[31] J. Thiemann, N. Ito, and E. Vincent, "The diverse environments multi-channel acoustic noise database (DEMAND): A database of multichannel environmental noise recordings," in *Proc. Mtgs. Acoust.*, vol. 19, no. 1, 2013.

[32] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR*, 2015.