Enhancing Sound Source Localization via False Negative Elimination (Supplementary Material)

Zengjie Song, *Member, IEEE*, Jiangshe Zhang,
Yuxi Wang, Junsong Fan, and Zhaoxiang Zhang, *Senior Member, IEEE*

This supplementary material contains three sections:

- Section A presents full derivations to formulate the representation update rules of predictive coding module (PCM).
- Section B introduces more details about our implementation.
- Section C gives additional ablation results.

We also provide a video demo of sound localization in the supplement.

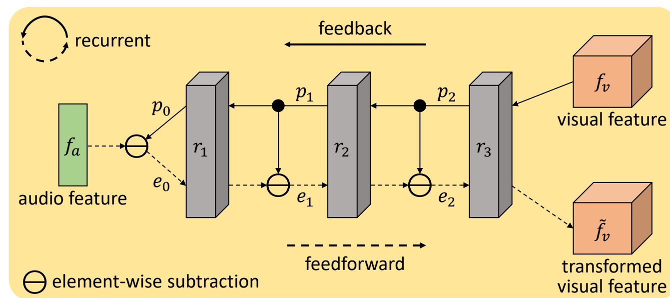

Fig. S1. Overview of predictive coding module (PCM). For simplicity we only show a 3-layer version.

A Full Formulation of PCM

The PCM, proposed for audio and visual feature alignment, plays an important role in improving sound localization performance of SSPL. As shown in Fig. S1, the key idea underlying PCM consists of three parts: (1) a feedback process (solid line) updates representations with the top-down predictions that originate from the visual feature; (2) a feedforward process (dashed line) also updates representations but with the bottom-up prediction errors that evolve from the audio feature; (3) a recursive modulation mechanism works to conduct the two processes alternatively. In the following, we first formulate the optimization objective of PCM, and then derive the representation update rules of the two processes, respectively, which are followed by a brief summary. *Note that for applications of PCM, we only need to explicitly update representations according to the rules given in Eqs. (S10)-(S13), without performing derivations again.*

Denote by f_a the audio feature, by f_v the visual feature, by $r_l(t), l \in \{1, \dots, L\}, t \in \{0, \dots, T\}$ the representation of the l-th layer of PCM network at time step t, and by $W_{l,l-1}$ the feedback connection weights from layer l to layer $l-1$ (and vice versa for $W_{l-1,l}$).

Optimization Objective. At layer l, PCM minimizes the following compound loss:

$$\mathcal{L}^l_{PCM} = \frac{\alpha_l}{2} \underbrace{||r_{l-1} - \mathcal{G}((W_{l,l-1})^T r_l)||^2_2}_{\mathcal{L}^l_1} + \frac{\beta_l}{2} \underbrace{||r_l - p_l||^2_2}_{\mathcal{L}^l_2}, \tag{S1}$$

where the function $\mathcal{G}$ corresponds to a generative process, α_l and β_l are scalars that control the weights of the two loss terms $\mathcal{L}^l_1$ and $\mathcal{L}^l_2$, and $p_l = \mathcal{G}((W_{l+1,l})^T r_{l+1})$ is the prediction of r_l.

Given the lower-level representation r_{l-1} and the top-down prediction p_l, our goal is to estimate r_l so as to decrease the loss in Eq. (S1). Minimizing $\mathcal{L}^l_1$ w.r.t. r_l leads to the representation that can be used to predict the *lower level* of representation r_{l-1}, while minimizing $\mathcal{L}^l_2$ w.r.t. r_l yields the representation that approximates the prediction signal p_l coming from a *higher level*. Therefore, the representation r_l associates lower- and higher-level information by reducing two prediction errors in $\mathcal{L}^l_1$ and $\mathcal{L}^l_2$. Minimizing losses at all layers can implicitly drive predictions at different levels to be mutually consistent [1].

Feedback Process. This process acts to update representations based on predictions from higher levels. Following [2], [3], we set $\mathcal{G}(x) = x$, and then employ gradient descent

- *Z. Song and J. Zhang are with the School of Mathematics and Statistics, Xi'an Jiaotong University, Xi'an 710049, China. Email: zjsong@hotmail.com, jszhang@mail.xjtu.edu.cn.*
- *Y. Wang and J. Fan are with the Center for Artificial Intelligence and Robotics, Hong Kong Institute of Science & Innovation, Chinese Academy of Sciences, Hong Kong, China. Email: yuxiwang93@gmail.com, junsong.fan@ia.ac.cn.*
- *Z. Zhang is with the New Laboratory of Pattern Recognition, State Key Laboratory of Multimodal Artificial Intelligence Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China, also with the University of Chinese Academy of Sciences, Beijing 100049, China, and also with the Center for Artificial Intelligence and Robotics, Hong Kong Institute of Science & Innovation, Chinese Academy of Sciences, Hong Kong, China. Email: zhaoxiang.zhang@ia.ac.cn.*

(Corresponding authors: Zhaoxiang Zhang and Jiangshe Zhang.)

to minimize $\mathcal{L}_2^l$ w.r.t. r_l, resulting in update rules:

$$p_l(t) = (W_{l+1,l})^T r_{l+1}(t), \tag{S2}$$

$$\frac{\partial \mathcal{L}_2^l}{\partial r_l(t)} = 2(r_l(t) - p_l(t)), \tag{S3}$$

$$\begin{aligned} r_l(t+1) &= r_l(t) - \eta_l \frac{\beta_l}{2} \frac{\partial \mathcal{L}_2^l}{\partial r_l(t)} \\ &= (1 - \eta_l \beta_l) r_l(t) + \eta_l \beta_l p_l(t), \end{aligned} \tag{S4}$$

where η_l is a non-negative scalar governing learning. For simplicity, let $b_l = \eta_l \beta_l$, and then Eq. (S4) is rewritten as follows:

$$r_l(t+1) = (1 - b_l) r_l(t) + b_l p_l(t). \tag{S5}$$

PCM carries out the feedback updating from top layer L to bottom layer 1, where the prediction of $r_L(t)$ at top layer is set as the visual feature, i.e., $p_L(t) \equiv f_v$.

Feedforward Process. This process works to update representations by using prediction errors from lower levels. For layer l, the lower-level prediction error e_{l-1} is the difference between r_{l-1} and p_{l-1}. We use gradient decent to minimize $\mathcal{L}_1^l$ w.r.t. r_l, leading to the following update rules:

$$e_{l-1}(t) = r_{l-1}(t) - p_{l-1}(t), \tag{S6}$$

$$\frac{\partial \mathcal{L}_1^l}{\partial r_l(t)} = -2 W_{l,l-1} e_{l-1}(t), \tag{S7}$$

$$\begin{aligned} r_l(t+1) &= r_l(t) - \kappa_l \frac{\alpha_l}{2} \frac{\partial \mathcal{L}_1^l}{\partial r_l(t)} \\ &= r_l(t) + \kappa_l \alpha_l W_{l,l-1} e_{l-1}(t), \end{aligned} \tag{S8}$$

where κ_l is a non-negative scalar like η_l. We also set $a_l = \kappa_l \alpha_l$ for simplicity. Similar to [2], we replace the feedback connection weights $W_{l,l-1}$ in Eq. (S8) with the transposed feedforward connection weights $(W_{l-1,l})^T$, and thus can endow PCM with more degrees of freedom to learn. Consequently the update rule in Eq. (S8) can be rewritten as a feedforward operation:

$$r_l(t+1) = r_l(t) + a_l (W_{l-1,l})^T e_{l-1}(t). \tag{S9}$$

In this process, PCM updates representations from bottom layer 1 to top layer L, where we let $r_0(t) \equiv f_a$ and $p_0(t) = (W_{1,0})^T r_1(t)$.

Summary. So far we formulate PCM with the simple linear activation functions. To introduce non-linearity into PCM, a nonlinear activation function ϕ (e.g., ReLU [4] used in [2] or GELU [5] used in this work) is applied to the above update Eqs. (S5) and (S9). By taking the recursive computing into account, we summarize the two processes as follows.

Nonlinear feedback process ($l = L, L-1, \ldots, 1$):

$$p_l(t) = (W_{l+1,l})^T r_{l+1}(t), \tag{S10}$$

$$r_l(t) \leftarrow \phi((1 - b_l) r_l(t-1) + b_l p_l(t)). \tag{S11}$$

Nonlinear feedforward process ($l = 1, 2, \ldots, L$):

$$e_{l-1}(t) = r_{l-1}(t) - p_{l-1}(t), \tag{S12}$$

$$r_l(t) \leftarrow \phi(r_l(t) + a_l (W_{l-1,l})^T e_{l-1}(t)). \tag{S13}$$

The two processes are conducted alternatively such that all representations in PCM are refined progressively. Finally, we transform the top layer representation at last time step, $r_L(T)$, to a new visual feature, $\tilde{f}_v$, with dimension the same as f_v by a 1×1 convolution. The representation learning of SSPL can proceed based on this $\tilde{f}_v$, instead of f_v as used in the vanilla SSPL.

TABLE S1
Learning Rate Settings in SSPL

Training set	SSPL (w/o PCM)		SSPL (w/ PCM)	
	lr_1	lr_2	lr_1	lr_2
Flickr10k	$2 \cdot 10^{-3}$	$5 \cdot 10^{-4}$	$5 \cdot 10^{-5}$	$2 \cdot 10^{-5}$
VGG-Sound10k	$1 \cdot 10^{-2}$	$5 \cdot 10^{-3}$	$5 \cdot 10^{-5}$	$2 \cdot 10^{-5}$
Flickr144k	$5 \cdot 10^{-3}$	$1 \cdot 10^{-3}$	$5 \cdot 10^{-5}$	$2 \cdot 10^{-5}$
VGG-Sound144k	$5 \cdot 10^{-3}$	$1 \cdot 10^{-3}$	$1 \cdot 10^{-5}$	$1 \cdot 10^{-5}$

lr_1 denotes the learning rate for projection and prediction MLPs, and lr_2 for remaining model parts.

TABLE S2
Learning Rate Settings in SACL

Training set	Visual network	Audio network
Flickr10k	$2 \cdot 10^{-4}$	$2 \cdot 10^{-4}$
VGG-Sound10k	$2 \cdot 10^{-4}$	$2 \cdot 10^{-4}$
Flickr144k	$1 \cdot 10^{-4}$	$1 \cdot 10^{-4}$
VGG-Sound144k	$1 \cdot 10^{-4}$	$1 \cdot 10^{-4}$

B Implementation Details

B.1 Architecture of PCM

For the feedback process of PCM, we use convolution layers (`kernel_size` $= 3$, `stride` $= 1$, `padding` $= 1$) followed by max pooling operation to reduce the spatial dimensionality of feature maps, while using 1×1 convolutions to decrease the number of channels. As for the feedforward process, the transposed convolutions (a.k.a. deconvolutions) are utilized and feature maps are upsampled by the "bilinear" upsampling algorithm, provided in PyTorch. Besides, the number of convolution layers is $L = 3$. From top layer L to bottom layer 1, the number of filters within each layer is 512, 512, and 128, respectively. The transposed convolution layers have the same setting. Moreover, we use GELU [5] as the nonlinear activation function for both processes. To stabilize and accelerate training, we adopt the batch normalization [6] before every non-linearity at each layer and at each time step, except the prediction of audio feature at bottom layer.

B.2 Training Details

The AdamW [7] optimizer is employed to train our model, where we set $(\beta_1, \beta_2) = (0.9, 0.999)$ and set weight decay to 10^{-4}. In practice, we find that learning rates have vital influence on SSPL's performance, hence we give detailed learning rate settings in Table S1. As for SACL, both visual and audio feature extractors keep the same learning rate, as shown in Table S2. During training, there are 256 image-audio pairs in each mini-batch, which are distributed in parallel on 2 or 4 NVIDIA GeForce GTX 1080 Ti GPUs.

TABLE S3
Parameters Used to Augment Images

Augmentation	Parameter
Crop	$p = 1$ output size of `Resize` = `int`(224×1.1) interpolation method of `Resize` = BICUBIC crop size = 224
Horizontal flip	$p = 0.5$
Vertical flip	$p = 0.5$
Translation	$p = 1.0$ maximum absolute fraction = $(0.2, 0.2)$
Rotation	$p = 1.0$ angle $\in \{0, 90, 180, 270\}$
Grayscale	$p = 0.2$
Color jittering	$p = 0.8$ maximum brightness adjustment = 0.4 maximum contrast adjustment = 0.4 maximum saturation adjustment = 0.4 maximum hue adjustment = 0.1
Gaussian blur	$p = 0.5$ $\sigma \in [0.1, 2.0)$

p denotes the probability that the corresponding operation will be performed.

TABLE S4
Ablation on Image Augmentations of SSPL

Augmentation	SoundNet-Flickr		VGG-SS	
	cIoU ↑	AUC ↑	cIoU ↑	AUC ↑
Crop (baseline)	0.514	0.499	<u>0.233</u>	0.324
+ Horizontal flip	<u>0.671</u>	<u>0.556</u>	**0.253**	**0.335**
+ Vertical flip	0.667	0.551	0.213	0.317
+ Translation	0.643	0.541	0.216	0.313
+ Rotation	0.639	0.543	0.227	<u>0.331</u>
+ Grayscale	0.610	0.535	0.226	0.318
+ Color jittering	**0.679**	**0.560**	0.232	0.328
+ Gaussian blur	0.619	0.533	0.204	0.299

For simplicity, we assess all cases based on SSPL (w/o PCM). All models are trained with 10k image-audio pairs and tested on the corresponding benchmarks. ***Bold*** *indicates the best and* <u>*underline*</u> *the runner-up.*

B.3 Image Augmentations

As shown in Table S3, a total of 8 image augmentations are considered in our SSPL. We follow HardWay [8] to select and set the first two augmentations: cropping with 224×224 resizing and horizontal flip. Then, we verify the effectiveness of other three spatial augmentations that are widely used in self-supervised visual representation learning [9], [10], i.e., vertical flip, translation, and rotation. Additionally, since our work draws inspiration from SimSiam [11], we also take into account its augmentation strategies: grayscale, color jittering, and Gaussian blur, while keeping their settings the same as SimSiam. Our SACL augments images in the way like SimSiam.

C Additional Ablations

C.1 Image Augmentation for Training SSPL

We investigate the influence of various image augmentations on SSPL's localization performance. As shown in Table S4, with the random crop baseline, our method can already achieve reasonable performance, indicating that object scales really matter in SSPL. However, except for horizontal flip (over 30% and 8% improvements on two datasets, respectively), randomly combining other augmentations with crop cannot obtain consistent gains. This is because compared with other combinations, the spatial augmentations (random crop + horizontal flip) are more suitable for the pre-trained and frozen VGG [12] to extract semantic visual features. Since our work is inspired by SimSiam [11], we also adopt its data augmentation strategies in SSPL, but find no benefits in this setting. Therefore, in all experiments of SSPL we take the spatial augmentations by default.

TABLE S5
Ablation on Feature Fusion Methods

Fusion method	Cat	$\otimes$	$\oplus$	AM (ours)
cIoU ↑	0.285	0.538	0.647	0.671
AUC ↑	0.414	0.512	0.540	0.556

We use different fusion methods in SSPL (w/o PCM), and train models on SoundNet-Flickr10k while evaluating on the standard benchmark.

TABLE S6
Influence of Recursive Cycles T in PCM

T	1	3	5	6	7	8
cIoU ↑	0.655	0.719	0.743	0.743	0.759	0.747
AUC ↑	0.562	0.584	0.587	0.595	0.595	0.590
GFLOPs ↓	38.3	43.0	47.6	49.9	52.2	54.5

All models are trained on SoundNet-Flickr10k and evaluated on the standard benchmark.

C.2 Ablation on Feature Fusion Methods

In SSPL, visual and audio features are fused by the attention mechanism to compute audio-visual representation. Here we compare other three feature fusion methods, i.e., concatenation (Cat), multiplication ($\otimes$), and addition ($\oplus$), with our attention module (AM). We can see from Table S5 that our AM outperforms others by a large margin. This verifies efficacy of the attention-based feature interaction.

C.3 Balance between Performance and Complexity of PCM

In Table S6, we quantitatively compare performance and time complexity of SSPL with varying recursive cycles T. We find that more recursive cycles cannot always bring gains as performance tends to be saturated when $T > 5$. Additionally, compared with SSPL (w/o PCM) that occupies 35.9 GFLOPs, SSPL (w/ PCM) conducts more operations with increasing T. As shown in Table S6, PCM takes, on average, 2.3 GFLOPs to complete one iteration. To balance between performance and time complexity, we set $T = 5$ during training.

References

[1] M. W. Spratling, "A review of predictive coding algorithms," *Brain and Cognition*, vol. 112, pp. 92–97, 2017. 1

[2] H. Wen, K. Han, J. Shi, Y. Zhang, E. Culurciello, and Z. Liu, "Deep predictive coding network for object recognition," in *ICML*, 2018, pp. 5266–5275. 1, 2

[3] Z. Song and Z. Zhang, "Visually guided sound source separation with audio-visual predictive coding," *IEEE Transactions on Neural Networks and Learning Systems (Early Access)*, 2023. 1

[4] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *ICML*, 2010, pp. 807–814. 2

[5] D. Hendrycks and K. Gimpel, "Gaussian error linear units (GELUs)," *arXiv:1606.08415*, 2016. 2

[6] C. Laurent, G. Pereyra, P. Brakel, Y. Zhang, and Y. Bengio, "Batch normalized recurrent neural networks," in *ICASSP*, 2016, pp. 2657–2661. 2

[7] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *ICLR*, 2019. 2

[8] H. Chen, W. Xie, T. Afouras, A. Nagrani, A. Vedaldi, and A. Zisserman, "Localizing visual sounds the hard way," in *CVPR*, 2021, pp. 16 867–16 876. 3

[9] S. Gidaris, P. Singh, and N. Komodakis, "Unsupervised representation learning by predicting image rotations," in *ICLR*, 2018. 3

[10] M. Ki, Y. Uh, J. Choe, and H. Byun, "Contrastive attention maps for self-supervised co-localization," in *ICCV*, 2021, pp. 2803–2812. 3

[11] X. Chen and K. He, "Exploring simple Siamese representation learning," in *CVPR*, 2021, pp. 15 750–15 758. 3

[12] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *ICLR*, 2015. 3