On Sorting and Padding Multiple Targets for Sound Event Localization and Detection with Permutation Invariant and Location-based Training

Robin Scheibler, Tatsuya Komatsu, Yusuke Fujita, and Michael Hentschel
LINE Corporation, Yotsuya, Shinjuku, 160-0004 Tokyo, Japan
E-mail: robin.scheibler@linecorp.com

***Abstract*—We explore the performance of permutation invariant and location-based training (PIT and LBT, respectively) for sound event localization and detection (SELD). Due to being intrinsically a multi-output multi-class and multi-task problem, the design space of loss functions for SELD is large, and, as of yet, rather unexplored. Our study revolves around the multiple activity coupled direction of arrival target format which cleverly combines direction and event probability into a single mean squared error loss. While PIT, and its variant auxiliary duplicating PIT (ADPIT), have been prominently featured in recent DCASE challenges, LBT has not yet been applied to SELD. In this work, we investigate some modifications to PIT and ADPIT, as well as the application of LBT to SELD. First, the PIT loss is changed to have a variable number of tracks per event class, providing extra flexibility. Second, we propose auxiliary duplicating or silence PIT (ADPIT-S), where unused tracks are indifferently filled with a duplicate event, or nothing. Finally, we propose to use LBT with ordering of the events by Cartesian or polar coordinates. We give two ways of padding the unused tracks, with zeros or by repeating the last event. We conduct experiments using the STARSS22 dataset from the DCASE Challenge 2022. We find that ordering by Cartesian coordinates with repeat padding is best for LBT. When comparing all loss functions, we suprisingly found that PIT performed the best. In addition, LBT turned out to be competitive with PIT and ADPIT. While ADPIT-S had slightly worse overall performance, it did better in terms of error rate and F-score metrics.**

I. Introduction

Sound event localization and detection (SELD) is the task of predicting the class and direction of arrival (DOA) of an acoustic event [1]. This task has been introduced in the DCASE[1] Challenge 2019. Taken separately, both tasks are well-understood and a large literature is available. DOA estimation in particular has a long history in the signal processing field [2]. Recently, several data-driven methods based on neural networks (NN) have been proposed, e.g., [3]. Sound event detection has taken off together with NN, as exemplified in the DCASE Challenge task 4, although it can be traced to computational auditory scene analysis [4]. The SELD task proposed in the DCASE Challenge makes available two types of recording formats, both derived from the 32 channel rigid spherical microphone array Eigenmike[2]. One is a tetrahydral subset of four of the channels of the eigenmike (MIC), the other is the first order ambisonics format (FOA) derived from all 32 channels. Both formats have four channels, for a total of eight, and can be used freely in the challenge.

The task being difficult to solve by conventional signal processing based methods, data-driven neural network based solutions have been widely adopted [1]. Various features have been proposed as inputs to the network. Compared to sound event detection (SED) only, in addition to spectral characteristics, the features need to also retain the spatial cues necessary for source localization. Generalized cross-correlation, inter-aural level and time differences, intensity vectors [5], per-channel energy normalization [6], SALSA-lite [7] have all been found effective. A comprehensive overview can be found in [8]. For the network architectures, a few different solutions have been introduced. Convolutional recurrent networks is popular and forms the backbone of the DCASE challenge baseline [8]. Recently, self-attention layers are replacing or complementing this architecture [9]. Another contender is the event-independent network architecture that uses two parallel branches for SED and DOA, with soft-stitching in between [10].

The SELD task is made difficult due to a few complications. First, this must be done for several event classes. Second, both the event presence *and* its DOA must be predicted. In addition, the DOA is meaningless when no event is present. Third, the number of targets is variable, from none to up to five concurrent targets can be present. This can change on a frame by frame basis. The multiple activity couple direction of arrival (multi-ACCDOA) target format consists of several tracks, each of which can detect one source of each class [11]. For each track and class, the network outputs a three-dimensional vector whose magnitude and direction represent event presence probability and direction, respectively. Then, a mean-squared error (MSE) PIT loss is applied between the network output and the target. While regular PIT places zero targets into empty tracks, auxiliary duplication PIT (ADPIT) repeats some of the events present in the frame to fill all the tracks [11]. Recently, location-based training (LBT) has been proposed as an alternative to PIT for source separation [12]. Instead of allowing the network to output any permutation of the sources, it is trained to output sources in a specific order, e.g., by azimuth or distance. LBT

[1] https://dcase.community
[2] https://mhacoustics.com/products

978-616-590-477-3 ©2022 APSIPA APSIPA ASC 2022

has not yet been applied to SELD.

The goal of this study is to compare the different loss functions for the multi-ACCDOA. In addition to the conventional PIT and ADDPIT, we propose three new loss functions. First, we propose to use PIT with a variable number of tracks per class. This allows to tune the number of tracks necessary for different types of events. Second, in ADPIT, there should be no preference for silence or a repeated event. Hence, we propose ADPIT-silence (ADPIT-S) where all possible permutations are taken from events present, a duplicated event, or silence. Finally, we adapt LBT for SELD. We try two ways to sort the order of events: by azimuth/elevation or Cartesian coordinates. Finally, we try both three and four tracks. When fewer events than tracks are present, we propose to pad with either zeros, or repeat the last event. We assess the performance on a system derived from our DCASE Challenge 2022 submission [13]. For training and validation, we use a synthetic dataset and the STARSS22 dataset [14].

Our main findings are summarized as follows. First, the performance difference between loss functions is not as large as expected. The best performing one is PIT with three tracks, closely followed by PIT with per-class number of tracks (max. three) and LBT using Cartesian coordinates and repeat padding. The ADPIT losses were found to be slightly inferior in our experiments, but had the lowest error rates (ER), in particular the newly proposed ADPIT-S.

The rest of this paper is organized as follows. In Section II, we review the background on ACCDOA, PIT and ADPIT. Then, we introduce the proposed loss functions in Section III. The experiments and theirs results are described in Section IV. Section V concludes.

II. Background

The goal of SELD is to detect sound events from C given classes, as well as their direction of arrival. Predictions must be made for each time frame of length $100\,\mathrm{ms}$. In addition, since multiple events of the same class may happen in the same frame, recent systems use several tracks. Let us index classes, frames, and tracks with indices c, f, and t, respectively. These indices run from 1 to C, F, and T, respectively. We will denote the presence and direction for an event of class c in frame f and track t by

$$p_{cft} \in \{0, 1\}, \quad \text{and} \quad \boldsymbol{d}_{cft} \in \mathbb{R}^3, \tag{1}$$

respectively. The direction $\boldsymbol{d}_{cft}$ is a unit vector, i.e., $\|\boldsymbol{d}_{cft}\| = 1$, and points from the center of the microphone array towards the event in three dimensional space. The notation $\|\,.\,\|$ denotes the Euclidean norm operator. Finally, we define $\tilde{T}_{cf} = \sum_{t=0}^{T} p_{cft}$, the actually number of events of class c occuring in frame f.

In the Multi-ACCDOA format, the network outputs vectors $\hat{\boldsymbol{q}}_{cft}$ for each class c, frame f, and track t. The targets for training the network are formed by defining the event presence probability as the norm of the direction vector such that the estimates of presence and direction are given by

$$\hat{p}_{cft} = \|\hat{\boldsymbol{q}}_{cft}\|, \quad \text{and} \quad \hat{\boldsymbol{d}}_{cft} = \frac{\hat{\boldsymbol{q}}_{cft}}{\|\hat{\boldsymbol{q}}_{cft}\|}. \tag{2}$$

Targets to train the network are formed simply as $\boldsymbol{d}_{cft} = p_{cft}\boldsymbol{d}_{cft}$. Then an MSE loss can be applied between $\boldsymbol{d}_{cft}$ and $\hat{\boldsymbol{d}}_{cft}$ to train the network. However, when $T > 1$, without further constraint, the order in which the network outputs the sources can be arbitrary. Thus, $\boldsymbol{d}_{cft}$ and $\hat{\boldsymbol{d}}_{cft}$ may correspond to different sources. It follows that a loss, such as the MSE, cannot be directly applied without knowing the correct order of the targets in the output.

A. Permutation Invariant Training (PIT)

Originally proposed for source separation [15], PIT goes around the order problem by finding and using the permutation with minimum loss value. It is defined as

$$\ell_{\mathrm{PIT}} = \sum_{c,f} \min_{\pi \in \mathcal{P}(T)} \sum_{t=1}^{T} \|\boldsymbol{d}_{cf\pi(t)} - \hat{\boldsymbol{d}}_{cft}\|^2. \tag{3}$$

The minimum in the inner sum is taken over all permutations $\mathcal{P}(T)$, the set of permutations of the elements of $\{1, ..., T\}$. We note that when the number of targets is less than T, then the invalid targets are the all zero vector. Note that the PIT loss of (3) is the class-wise PIT described in [11], which is different from the track-wise PIT of [16] that does not use the ACCDOA format.

B. Auxiliary Duplicating PIT (ADPIT)

In ADPIT [11], when at least one target exists, the invalid targets are not set to zero, as in PIT. Instead, they are set to a copy of some existing target. In other words, the network will output the same target in multiple tracks. When no events at all are present for a class, then the targets are zero vectors, as in PIT. This approach was initially proposed for source separation [17] where usual loss functions cannot be applied to zero outputs. While this original motivation does not apply in SELD, ADPIT was shown to be effective nonetheless [11]. The loss is similarly to (3), with the minimum taken over a different set of functions. Let us define the function $\rho : \{1, \ldots, T\} \to \{1, \ldots, \tilde{T}_{cf}\}$ where $\tilde{T}_{cf}$ is the number of events of class c occurring in frame f. The set $\mathcal{A}(\tilde{T}, T)$ is the set of distinct combinations of T elements of $\{1, \ldots, \tilde{T}\}$ with replacement and containing $\{1, \ldots, \tilde{T}\}$. We write the loss as

$$\ell_{\mathrm{ADPIT}} = \sum_{c,f} \min_{\rho \in \mathcal{A}(\tilde{T}_{cf}, T)} \sum_{t=1}^{T} \|\boldsymbol{d}_{cf\rho(t)} - \hat{\boldsymbol{d}}_{cft}\|^2. \tag{4}$$

Listing 1 describes how to generate the set $\mathcal{A}(\tilde{T}, T)$ for arbitrary values of $\tilde{T}$ and T in Python. While a more efficient version based on the Kuhn-Munkres algorithm [18] almost certainly exists, we only present here a naive implementation.

```python
from typing import List, Dict, Optional
from itertools import permutations,
    combinations_with_replacement

def get_adpit_permutations(
    n_events: int,
    n_tracks: int,
    use_adpit_s: Optional[bool] = False
) -> List[int]:
    """
    Generates the ADPIT permutations

    Parameters:
    n_events (int): num. of events (<= n_tracks)
    n_tracks (int): maximum number of tracks
    use_adpit_s (bool): Use ADPIT-S if True

    Returns:
    A dict of list of ints. The Tth element is
    the list of permutations for T targets
    """
    assert n_events <= n_tracks

    if use_adpit_s:
        base = tuple(range(1, n_events + 1))
        possible_events = range(n_events + 1)
    else:
        base = tuple(range(n_events))
        possible_events = range(n_events)

    combs = []
    for comb in combinations_with_replacement(
        possible_events, n_tracks - n_events
    ):
        combs.append(base + comb)

    perm_set = set()
    for c in combs:
        for cp in permutations(c):
            perm_set.add(cp)

    return list(perm_set)
```

Listing 1. Python code to generate permutations for ADPIT and ADPIT-S.

C. Location-based Training (LBT)

LBT has been proposed recently as an alternative to PIT for source separation [12]. It disambiguates the output order of sources by training the network to sort them according to their physical location. Unlike PIT and ADPIT, the loss only needs to be evaluated for a single permutation of the sources and is thus linear in complexity with respect to the number of events (rather than cubic or factorial, depending on the implementation for PIT and ADPIT). Furthermore, while it is unclear how PIT or ADPIT chose the permutation in the network, LBT is interpretable by construction. For source separation, the loss was shown to be superior to PIT on simulated data. It has not been applied yet to SELD. Furthermore, [12] does not consider the case of a variable number of outputs. In Section III-C, we extend LBT for SELD and propose some solutions to this shortcoming.

TABLE I
NUMBER OF TRACKS USED FOR DIFFERENT CLASSES IN PIT-C. THE PICTOGRAMS ARE DESCRIBED IN SECTION IV-A. THE MAXIMUM NUMBER OF TRACKS T IS 3 OR 4 THROUGHOUT THIS WORK.

												
T	T	T	1	T	2	2	1	1	T	1	1	1

III. PROPOSED LOSS FUNCTIONS

There is still a considerable part of the design space of SELD loss functions left unexplored. We describe now some variations of PIT and ADPIT, as well as the adaptation of LBT to SELD training.

A. PIT with per-class number of tracks (PIT-C)

The original class-wise PIT described in Section II-A uses the same number of tracks for all classes. Since not all event classes are likely to have the same number of simultaneous events, we propose to use a per-class number of tracks, T_c. For example, in typical scenes, it is likely that multi concurrent speakers are present. Thus, for *male speech* or *female speech* it makes sense to have a large number of tracks. On the contrary, events of *water tap, faucet* class are likely to come from a single source. The the number of tracks was chosen heuristically and is given in Table I.

B. Auxiliary duplicating or silence PIT (ADPIT-S)

As explained in Section II, PIT and ADPIT make distinct choices as to what to do with empty tracks when the number of events is less than T (but at least one). PIT pads the targets to length T by adding zero (or silence) events. On the contrary, ADPIT makes sure to use all tracks by duplicating some of the events. Unlike in source separation [17], there is no clear motivation to prefer either strategy in SELD. We thus propose ADPIT-S that evaluates a superset of the target of PIT and ADPIT. We introduce the mapping $\gamma : \{1, \dots, T\} \to \{0, \dots, \tilde{T}_{cf}\}$. Compared to the mapping ρ of ADPIT, the codomain of γ contains 0 which will correspond to no event (silence). We also define the set $\mathcal{S}(\tilde{T}, T)$ that contains all combinations of T elements of $\{0, \dots, \tilde{T}_{cf}\}$ with replacement and containing $\{1, \dots, \tilde{T}\}$. Then, the ADPIT-S loss is

$$\ell_{\text{ADPIT-S}} = \sum_{c,f} \min_{\gamma \in \mathcal{S}(\tilde{T}_{cf}, T)} \sum_{t=1}^{T} \|\boldsymbol{d}_{cf\gamma(t)} - \hat{\boldsymbol{d}}_{cft}\|^2, \quad (5)$$

where we define the silent target as $\boldsymbol{d}_{cf0} = \mathbf{0}$. See Listing 1 for Python code to generate $\mathcal{S}(\tilde{T}, T)$.

C. LBT for SELD

In [12], sources are ordered by azimuth or distances. Furthermore, sources are considered static. In SELD, sources are dynamic and the loss is computed frame-wise. The loss is very similar to other losses defined so far, except for the

absence of a minimum,

$$\ell_{\mathrm{LBT}} = \sum_{c,f} \sum_{t=1}^{T} \|\boldsymbol{d}_{cft} - \hat{\boldsymbol{d}}_{c,f,t}\|^2, \tag{6}$$

which is just the MSE, but where, crucially, the targets $\boldsymbol{d}_{cft}$ have been sorted in advance. There are two important choices to make for this function, 1) how to sort the direction vectors, and 2) how to pad the targets to the number of tracks. For 1), we define the comparison function $\sigma(\boldsymbol{d})$ and $\boldsymbol{d}_{cft}$ are sorted such that

$$\sigma(\boldsymbol{d}_{cfm}) \leq \sigma(\boldsymbol{d}_{cfn}), \quad \forall m \leq n,\ m, n \in \mathbb{N} \tag{7}$$

We let $\boldsymbol{d} = [x, y, z]^\top \in \mathbb{R}^3$ be a direction vector with azimuth θ_{az} and elevation θ_{el}, and define two such functions,

$$\sigma_{\text{pol}}(\boldsymbol{d}) = 10\frac{\theta_{\text{az}} - \pi}{2\pi} + \frac{\theta_{\text{el}} - \pi/2}{\pi}, \tag{8}$$

$$\sigma_{\text{cart}}(\boldsymbol{d}) = 100\frac{x+1}{2} + 10\frac{y+1}{2} + \frac{z+1}{2}, \tag{9}$$

i.e., sorting by polar and Cartesian coordinates, respectively, of the DOA. For 2), we define the two following padded target definitions,

$$\boldsymbol{d}_{cft}^{(\text{zero})} = \begin{cases} \boldsymbol{d}_{cft} & \text{if } t \leq \tilde{T}_{cf} \\ \boldsymbol{0} & \text{else} \end{cases}, \tag{10}$$

$$\boldsymbol{d}_{cft}^{(\text{repeat})} = \begin{cases} \boldsymbol{d}_{cft} & \text{if } t \leq \tilde{T}_{cf} \\ \boldsymbol{d}_{cf\tilde{T}_{cf}} & \text{else} \end{cases}. \tag{11}$$

In words, *zero* sets unoccupied tracks to no events (i.e., silence), and *repeat* to the last event.

IV. Experiments

We carry out experiments on a variation of our submission system for the DCASE Task 3 Challenge 2022 [13]. Our experiments have two main objectives. First, we want to identify the best combination of sorting order and padding for LBT. Second, we will compare the performance of all the loss functions discussed.

A. Dataset

The official challenge dataset STARSS22 [14] is used for the evaluation and training following the official split. This dataset contains the 13 following classes: *female speech* (), *male speech* (), *clapping* (), *telephone* (), *laughter* (), *domestic sounds* (), *walk* (), *door* (), *music* (), *music instruments* (), *water tap, faucet* (), *bell* (), *knock* (). Due to its high quality, this dataset only contains a total of $4.9\,\mathrm{h}$ which is not sufficient to train our system. Thus, we also use the baseline training synthetic dataset provided by the task organizers [19]. This dataset is created by remixing sound events from the FSD50K dataset [20], [21] with the measured RIR from the TAU-SRIR database [22], [23]. However, this synthetic dataset only contains up to two overlapping events, and no interfering events. To solve this issue, we complement it by a second synthetic dataset created using the same recipe [24] but with up to 4 overlapping events and interferring sound events. The total training and evaluation data used is $42.9\,\mathrm{h}$ and $2\,\mathrm{h}$, respectively.

B. System

1) Features: All the features are derived from the FOA signals. We use three different types of features, two of which are log-mel-spectrogram of the four FOA channels and intensity vectors [5]. The main novelty of our system described in [13] is robust features obtained from pre-trained models. These features are obtained by applying a fine-tuned self-supervised model on the output of a pre-trained multichannel separation model to obtain a preliminary classification of the events present. The separation model uses weighted prediction error (WPE) [25] for dereverberation and independent vector analysis with iterative source steering for the separation [26]. It is trained in advance end-to-end with a spatial loss [27] using the DOA groundtruth provided in the SELD training dataset. The self-supervised model is a self-supervised audio spectrogram transformer (SSAST) fine tuned for single channel event detection on the SELD dataset described in Section IV-A and stripped of DOA information. The SSAST is applied in parallel to the omni channel of the FOA and the four output channels of the separation, producing a size $5 \times 13 \times 497$ output tensor, where 13 and 497 are the numbers of classes and frames, respectively.

2) Network Architecture: The backbone of our network is a conventional CNN-Conformer network. The log-mel-spectrograms of the four FOA channels and the intensity vectors (total of 7 input channels) are first processed by a convolutional network with two layers. We use kernels of size 3×3 and stride 2 in frequency and time to reduce the size of the input signal. The number of channels at the output is 256. Group normalization with four groups and ReLU activations are used after each layer. After the two strided convolutions, the remaining 32 frequency dimensions are merged with the 256 channels and projected to dimension 256 by a linear layer before the output. The output of this stage is an embedding signal with 256 dimensions and a frame interval of $40\,\mathrm{ms}$. This output is combined with the predictions from the SSAST network using a linear projection and concatenation operation. The resulting signal is fed into a conformer-encoder [28] with eight layers and convolution kernel size 7. The output of the encoder is again combined with the predictions from the SSAST through a different linear projection. Finally, a multi-layer perceptron (MLP) output head with a GeLU non-linearity projects the embedding to the final output dimension. The size of this network is 15.3 millions parameters. The overall system architecture is illustrated in Fig. 1. Details can be found in [13].

3) Post-processing and Calibration: The post-processing works in two steps. First, events are detected if $p_{cft} \geq \tau_c$ at the output framerate of the network, where τ_c is the detection threshold. We run a de-duplication procedure to remove duplicate events produced by the Multi-ACCDOA

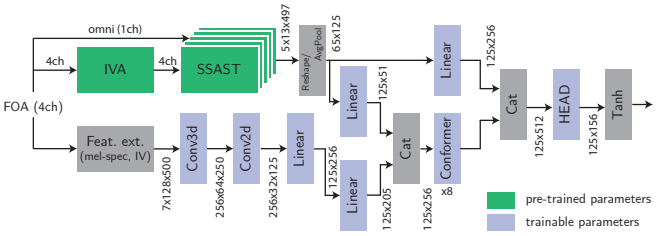

Fig. 1. Diagram of the system used in the experiments [29], [13].

format. Events from different tracks of the same class with directions closer than θ_c, a class specific threshold, are merged together. Second, all the events from the same output frame are aggregated together. Because the output frames of the network are $40\,\mathrm{ms}$ and the target frames are $100\,\mathrm{ms}$, there are 2 or 3 events per output frame, track, and class. For every output frame and class, we find the event with largest p_{cft} and count all events within θ_c. If the count is larger than η_c, we declare an event with direction given by the average of all aggregated events, weighted by their probability. By default, we use $\tau_c = 1/2$, $\theta_c = 15°$, and $\eta_c = 1$. To maximize performance, we use a post-processing calibration procedure where τ_c, θ_c, and η_c are chosen per class to minimize the SELD score on the validation dataset of STARSS22.

C. Performance Metrics

The SELD task is evaluated with a set of metrics designed to evaluate both the classification and DOA estimation in a joint manner [1]. An event is considered to be correctly detected when the class is correct and the estimated DOA is within $20°$ of the groundtruth. The *error rate* (ER) counts the number of errors, e.g., misdetection, false positive, etc, according to this definition. An *F-score* (F) gives insight into the precision and recall of the system. The *location error* (LE) is a pure location metric. It is the average error for correctly detected events. Finally, the *location recall* (LR) measures how well a type of event is detected. The ER metric is evaluated for all classes jointly, while F, LE, and LR are evaluated per-class and macro-averaged. The SELD score aggregates all metrics,

$$\mathrm{SELD} = \frac{1}{4}\left[\mathrm{ER} + (1-\mathrm{F}) + \frac{\mathrm{LE}}{180} + (1-\mathrm{LR})\right], \qquad (12)$$

but is not an official challenge metric.

D. Training

The optimizer is Adam [30] with learning rate 0.001. We do learning rate warm-up over the first 10000 steps. The network is trained for 500 epochs. We apply data augmentation during training, namely, SpecAugment [31] and random data rotations [32]. The progress of the optimization is monitored on the validation set of STARSS22 using the SELD score defined in (12). At the end of the training we average the models from the M epochs with the lowest validation SELD scores. In this work, we try $M = 1, 3, 10$ and pick the best model after the post-processing calibration.

TABLE II
PERFORMANCE OF LBT LOSS WITH $T = 4$ TRACKS FOR DIFFERENT CHOICES OF THE ORDER AND PADDING FUNCTIONS. THE NUMBER OF MODEL AVERAGES IS M.

Order (σ)	Pad	ER$^{\downarrow}$	F$^{\uparrow}$	LE$^{\downarrow}$	LR$^{\uparrow}$	SELD$^{\downarrow}$
cart	repeat	0.51	0.48	16.53	0.62	0.3749
cart	zero	0.51	0.48	17.31	0.58	0.3895
pol	repeat	0.52	0.46	18.13	0.57	0.3981
pol	zero	0.58	0.45	18.94	0.63	0.4007

TABLE III
THE NUMBER TRACKS AND MODEL AVERAGED ARE T AND M, RESPECTIVELY.

Loss	T	ER$^{\downarrow}$	F$^{\uparrow}$	LE$^{\downarrow}$	LR$^{\uparrow}$	SELD$^{\downarrow}$
PIT	3	0.50	0.49	17.79	0.61	0.3728
PIT-C	3	0.50	0.49	18.01	0.62	0.3739
LBT-cart-repeat	4	0.51	0.48	16.53	0.62	0.3749
PIT	4	0.50	0.45	17.21	0.64	0.3754
ADPIT	4	0.50	0.46	18.11	0.64	0.3758
ADPIT-S	3	0.50	0.47	16.16	0.60	0.3786
ADPIT	3	0.50	0.45	17.39	0.64	0.3787
ADPIT-S	4	0.48	0.49	17.63	0.57	0.3798
LBT-cart-repeat	3	0.50	0.45	17.09	0.62	0.3808
PIT-C	4	0.52	0.46	17.26	0.58	0.3959

E. Results

1) Evaluation of LBT losses: Table II shows the SELD metrics for LBT with all combinations of sorting order (Cartesian or polar) and padding (zero or repeat). These experiments were run for $T = 4$ channels. Ordering the target events by Cartesian coordinates (given by (8)) significantly outperformed the polar coordinates order (given by (9)). In particular ER is reduced and F is increased. For the padding, *repeat*, where the last event is repeated, resulted in a lower SELD score. We thus selected LBT with Cartesian order and repeat padding for the remaining experiments.

2) Comparison of All Loss Functions: We evaluated PIT, PIT-C, ADPIT, ADPIT-S, and LBT-cart-repeat with $T = 3$ and $T = 4$ tracks. Table III lists the performance of all systems ranked by SELD score from lowest (best) to highest (worst). Surprisingly, the best performing system was the regular PIT loss with only 3 tracks. It was closely followed by PIT-C with also 3 tracks. This shows that for the STARSS22 dataset, only 3 tracks was largely sufficient, due to more than 3 overlapping events *of the same class* being very rare. Reducing the track number for some classes in PIT-C did not lead to an advantage. Following, with very close metrics, were LBT, PIT, and ADPIT with 4 tracks each. This finding suggests that none of the loss functions is fundamentally superior to others and all can efficiently train a SELD network. Nevertheless, PIT seemed to outperform ADPIT consistently. ADPIT-S performed slightly worse than ADPIT for 4 tracks, but did better with 3 tracks. Interestingly,

ADPIT-S has better ER and F scores than ADPIT, but worse LR. Thus, it introduces a different bias into the training. Finally, LBT with 3 tracks and PIT-C with 4 tracks performed the worst in the experiments.

V. Conclusions

Loss functions are a key component in the training of neural networks. They can help bias the learning when data is scarce, or the evaluation metrics not differentiable, as in SELD. In this work, we have evaluated losses based on permutation invariant training (PIT), auxiliary duplicating permutation invariant training (ADPIT), and location-based training (LBT), with some variations, for a total of five loss functions. Contrary to previously published work [11], we found that the simple PIT outperformed the ADPIT loss. LBT was found to be a serious contender with results on par or better than PIT or ADPIT. Importantly, it requires the fewest computations and is straightforward to implement. Our proposed ADPIT-S performed on par with ADPIT but displayed better error rates and F-scores.

Since, different loss functions encode different biases, it would be important in future work to extend this evaluation to more models and datasets. In particular, the performance may change dramatically in the data-rich regime. While data for SELD is difficult to obtain, promising data augmentation and simulation schemes could be used instead, for example that of the winning team of the 2022 challenge [33].

References

[1] A. Politis, A. Mesaros, S. Adavanne, T. Heittola, and T. Virtanen, "Overview and evaluation of sound event localization and detection in dcase 2019," *IEEE/ACM Trans. Audio Speech Lang. Process.*, vol. 29, pp. 684–698, 2020.

[2] H. Krim and M. Viberg, "Two decades of array signal processing research: the parametric approach," *IEEE Signal Process. Mag.*, vol. 13, no. 4, pp. 67–94, Jan. 1996.

[3] S. Chakrabarty and E. A. P. Habets, "Multi-Speaker DOA Estimation Using Deep Convolutional Networks Trained With Noise Signals," *Selected Topics in Signal Processing, IEEE Journal of*, vol. 13, no. 1, pp. 8–21, Apr. 2019.

[4] G. J. Brown and M. Cooke, "Computational auditory scene analysis," *Comput. Speech Lang.*, vol. 8, no. 4, pp. 297–336, Oct. 1994.

[5] K. Lopatka, J. Kotus, and A. Czyzewski, "Detection, classification and localization of acoustic events in the presence of background noise for acoustic surveillance of hazardous situations," *Multimedia Tools and Applications*, vol. 75, no. 17, pp. 10 407–10 439, 2016.

[6] V. Lostanlen et al., "Per-Channel Energy Normalization: Why and How," *IEEE Signal Process. Lett.*, vol. 26, no. 1, pp. 39–43, Jan. 2019.

[7] T. N. Tho Nguyen, D. L. Jones, K. N. Watcharasupat, H. Phan, and W.-S. Gan, "SALSA-Lite: A fast and effective feature for polyphonic sound event localization and detection with microphone arrays," in *Proc. IEEE ICASSP*, Singapore, SG, May 2022.

[8] S. Adavanne, A. Politis, J. Nikunen, and T. Virtanen, "Sound event localization and detection of overlapping sources using convolutional recurrent neural networks," *IEEE J. Sel. Top. Signal Process.*, vol. 13, no. 1, pp. 34–48, Apr. 2019.

[9] K. Shimada et al., "Ensemble of ACCDOA- and EINV2-based Systems with D3Nets and Impulse Response Simulation for Sound Event Localization and Detection," DCASE Challenge, Tech. Rep., Jun. 2021.

[10] Y. Cao, T. Iqbal, Q. Kong, F. An, W. Wang, and M. D. Plumbley, "An Improved Event-Independent Network for Polyphonic Sound Event Localization and Detection," in *Proc. IEEE ICASSP*, Toronto, CA, Jun. 2021, pp. 885–889.

[11] K. Shimada, Y. Koyama, S. Takahashi, N. Takahashi, E. Tsunoo, and Y. Mitsufuji, "Multi-ACCDOA: Localizing and detecting overlapping sounds from the same class with auxiliary duplicating permutation invariant training," in *Proc. IEEE ICASSP*, Singapore, May 2022, pp. 316–320.

[12] H. Taherian, K. Tan, and D. Wang, "Location-Based Training for Multi-Channel Talker-Independent Speaker Separation," in *Proc. IEEE ICASSP*, Singapore, SG, May 2022, pp. 696–700.

[13] R. Scheibler, T. Komatsu, Y. Fujita, and M. Hentschel, "Sound event localization and detection with pre-trained audio spectrogram transformer and multichannel separation network," in *DCASE Workshop*, Nov. 2022, submitted.

[14] A. Politis et al., "STARSS22: A dataset of spatial recordings of real scenes with spatiotemporal annotations of sound events," *arXiv preprint arXiv:2206.01948*, 2022.

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

[16] Y. Cao, T. Iqbal, Y. Zhong, W. Wang, and M. D. Plumbley, "Event-independent network for polyphonic sound event localization and detection," in *DCASE Workshop*, Nov. 2020, pp. 11–15.

[17] Y. Luo and N. Mesgarani, "Separating varying numbers of sources with auxiliary autoencoding loss," in *Proc. Interspeech*, Shanghai, CN, Oct. 2020, pp. 2622–2626.

[18] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1–2, pp. 83–97, Mar. 1955.

[19] A. Politis, "[DCASE2022 Task 3] Synthetic SELD mixtures for baseline training," Apr. 2022. [Online]. Available: https://doi.org/10.5281/zenodo.6406873

[20] E. Fonseca et al., "FSD50K: an open dataset of human-labeled sound events," *IEEE/ACM Trans. Audio Speech Lang. Process.*, vol. 30, pp. 829–852, 2022.

[21] E. Fonseca, X. Favory, J. Pons, F. Font, and X. Serra, "FSD50K," Oct. 2020. [Online]. Available: https://doi.org/10.5281/zenodo.4060432

[22] A. Politis, S. Adavanne, and T. Virtanen, "TAU Spatial Room Impulse Response Database (TAU- SRIR DB)," Apr. 2022. [Online]. Available: https://doi.org/10.5281/zenodo.6408611

[23] ——, "A dataset of reverberant spatial sound scenes with moving sources for sound event localization and detection," in *Proc. DCASE*, Tokyo, JP, Nov. 2020.

[24] D. Krause and A. Politis, "danielkrause/dcase2022-data-generator." [Online]. Available: https://github.com/danielkrause/DCASE2022-data-generator

[25] T. Nakatani et al., "Speech dereverberation based on variance-normalized delayed linear prediction," *IEEE Trans. Audio Speech Lang. Process.*, vol. 18, no. 7, pp. 1717–1731, Sep. 2010.

[26] R. Scheibler and M. Togami, "Surrogate source model learning for determined source separation," in *Proc. IEEE ICASSP*, Toronto, CA, Jun. 2021, pp. 176–180.

[27] K. Saijo and R. Scheibler, "Spatial loss for unsupervised multi-channel source separation," in *Proc. Interspeech*, Incheon, KR, Sep. 2022.

[28] A. Gulati et al., "Conformer: Convolution-augmented transformer for speech recognition," in *Proc. Interspeech*, 2020, pp. 5036–5040.

[29] R. Scheibler, T. Komatsu, Y. Fujita, and M. Hentschel, "3d cnn and conformer with audio spectrogram transformer for sound event detection and localization," DCASE Challenge, Tech. Rep., Jun. 2022.

[30] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR*, San Diego, CA, USA, May 2015.

[31] D. S. Park et al., "SpecAugment: A simple data augmentation method for automatic speech recognition," in *Proc. Interspeech*. ISCA, Sep. 2019, pp. 2613–2617.

[32] F. Ronchini, D. Arteaga, and A. Pérez-López, "Sound event localization and detection based on crnn using rectangular filters and channel rotation data augmentation," in *Proc. DCASE2020*, Tokyo, JP, Nov. 2020.

[33] Q. Wang, L. Chai, H. Wu, Z. Nian, S. Niu, S. Zheng, Y. Wang, L. Sun, Y. Fang, J. Pan, J. Du, and C.-H. Lee, "The NERC-SLIP system for sound event localization and detection of DCASE2022 challenge," DCASE2022 Challenge, Tech. Rep., Jun. 2022.