UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

SOUND EVENT DETECTION AND SEPARATION IN DOMESTIC ENVIRONMENTS USING DEEP LEARNING

Diploma Thesis

Vasileios Koutsoukis

Supervisor: Assoc. Prof. Gerasimos Potamianos

Volos, September 2021

UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

SOUND EVENT DETECTION AND SEPARATION IN DOMESTIC ENVIRONMENTS USING DEEP LEARNING

Diploma Thesis

Vasileios Koutsoukis

Supervisor: Assoc. Prof. Gerasimos Potamianos

Volos, September 2021

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΕΝΤΟΠΙΣΜΟΣ ΚΑΙ ΔΙΑΧΩΡΙΣΜΟΣ ΗΧΟΥ ΣΕ ΟΙΚΙΣΤΙΚΑ ΠΕΡΙΒΑΛΛΟΝΤΑ ΧΡΗΣΙΜΟΠΟΙΩΝΤΑΣ ΜΕΘΟΔΟΥΣ ΒΑΘΙΑΣ ΜΑΘΗΣΗΣ

Διπλωματική Εργασία

Βασίλειος Κουτσούκης

Επιβλέπων: Αναπλ. Καθ. Γεράσιμος Ποταμιάνος

Βόλος, Σεπτέμβριος 2021

Approved by the Examination Committee:

Supervisor **Gerasimos Potamianos**
Associate Professor,
Department of Electrical and Computer Engineering,
University of Thessaly

Member **George Stamoulis**
Professor,
Department of Electrical and Computer Engineering,
University of Thessaly

Member **Antonios Argyriou**
Associate Professor,
Department of Electrical and Computer Engineering,
University of Thessaly

Date of approval: 28-9-2021

Acknowledgements

First and foremost, I would like to express my most profound sense of gratitude and immeasurable appreciation to my thesis supervisor, Associate Professor Gerasimos Potamianos, for giving me the chance to work on this interesting research topic, as well as for his help, guidance, and insight throughout my thesis journey.

Furthermore, I would like to thank all my other tutors of the Electrical and Computer Engineering department for expanding my knowledge and teaching me valuable courses. Also, I would like to offer my special thanks to Emeritus Professor Elias Houstis for being a great influence in turning my research interests towards the applications of Machine Learning in the Computer Science field.

Finally, I am incredibly thankful to my family, my friends, and those people who supported and encouraged me.

DISCLAIMER ON ACADEMIC ETHICS AND INTELLECTUAL PROPERTY RIGHTS

«Being fully aware of the implications of copyright laws, I expressly state that this diploma thesis, as well as the electronic files and source codes developed or modified in the course of this thesis, are solely the product of my personal work and do not infringe any rights of intellectual property, personality and personal data of third parties, do not contain work / contributions of third parties for which the permission of the authors / beneficiaries is required and are not a product of partial or complete plagiarism, while the sources used are limited to the bibliographic references only and meet the rules of scientific citing. The points where I have used ideas, text, files and / or sources of other authors are clearly mentioned in the text with the appropriate citation and the relevant complete reference is included in the bibliographic references section. I fully, individually and personally undertake all legal and administrative consequences that may arise in the event that it is proven, in the course of time, that this thesis or part of it does not belong to me because it is a product of plagiarism».

The declarant

Vasileios Koutsoukis
28-9-2021

Abstract

In this Thesis, we propose state-of-the-art methods for the task of sound event detection. Specifically, we develop systems that can automatically detect sound events and their time boundaries. These systems include artificial neural network architectures based on conformers and residual convolutional recurrent neural networks. Since multiple sound events may occur simultaneously, we investigate the impact that source separation, as a preprocessing step, has on our models' performance. In our work we use all the proposed datasets of the DCASE Challenge 2021 Task 4. The audio clips that these datasets contain are either recorded in domestic environments or synthesized to simulate a domestic environment. Finally, we evaluate our approaches on the public evaluation dataset proposed for the DCASE Challenge 2021 Task 4, outperforming the task's baseline systems in all scenarios.

Περίληψη

Σε αυτήν τη διπλωματική, προτείνουμε προηγμένες μεθόδους για το πρόβλημα της ανίχνευσης ηχητικών συμβάντων. Συγκεκριμένα, αναπτύσσουμε συστήματα που μπορούν να ανιχνεύσουν με αυτόματο τρόπο ηχητικά γεγονότα και τα χρονικά τους όρια. Αυτά τα συστήματα περιλαμβάνουν αρχιτεκτονικές τεχνητών νευρωνικών δικτύων που βασίζονται σε conformers και υπολειπόμενα συνελικτικά επαναλαμβανόμενα νευρωνικά δίκτυα. Δεδομένου ότι πολλά ηχητικά συμβάντα μπορεί να λάβουν χώρα ταυτόχρονα, διερευνούμε τον αντίκτυπο που έχει ο διαχωρισμός πηγής, ως βήμα προεπεξεργασίας, στην απόδοση των μοντέλων μας. Στην εργασία μας χρησιμοποιούμε όλα τα σύνολα δεδομένων που προτείνονται από το διεθνή διαγωνισμό DCASE 2021 Task 4. Αυτά τα σύνολα δεδομένων περιέχουν ηχητικά κλιπ που είτε έχουν καταγραφεί σε οικιακά περιβάλλοντα ή έχουν κατασκευαστεί για να προσομοιώσουν ένα οικιακό περιβάλλον. Τέλος, αξιολογούμε τα μοντέλα μας χρησιμοποιώντας το σύνολο δοκιμής που προτείνεται από τον ίδιο τον διαγωνισμό, ξεπερνώντας τα βασικά συστήματα σε όλα τα σενάρια.

Table of contents

List of Figures

List of Tables

Abbreviations

BCE	Binary Cross Entropy
BGRU	Bidirectional Gated Recurrent Unit
CNN	Convolutional Neural Network
CT	Cross-Trigger
CRNN	Convolutional Recurrent Neural Network
DTC	Detection Tolerance Criterion
eFPR	effective False-Positive Rate
eTPR	effective True-Positive Rate
FP	False Positive
GTC	Ground-Truth intersection Criterion
HMM	Hidden Markov Model
MHA	Multi-Head Attention
MHSA	Multi-Head Self-Attention
MSE	Mean Square Error
PSDS	Polyphonic Sound Detection Score
ReLU	Rectified Linear activation Unit
ROC	Receiver Operating Characteristic
SED	Sound Event Detection
SSEP	Source Separation
STFT	Short-Time Fourier Transform
TDCN	Time-Dilated Convolutional Network
TP	True Positive

Chapter 1

Introduction

1.1 Motivation

In our everyday lives, sound conveys important information. Our auditory perception is incredibly skilled at separating different sound sources and directing attention to the one we are interested in. Automatic sound event detection (SED) [5] aims to recognize what is happening in an audio signal and when it is happening. In other words, its goal is to identify and classify sound events in a variety of sound classes and to detect the onset and offset time of each sound event.

Monophonic and polyphonic SED [6] are the two primary types that SED can be categorized to. Monophonic SED systems can only detect the most prominent sound event at a time. This is a significant drawback since sound events frequently occur concurrently in real life. For example, an audio signal captured at a street corner may include automobile horns, people speaking, sirens, and steadily falling rain, all at the same time. On the other hand, polyphonic SED aims to detect several sound events concurrently present at any time. This scenario is more suitable for real-world applications. The number of sound events to be identified in this scenario might vary between time instances. In this Thesis, we are interested in the polyphonic SED scenario.

SED is a rapidly growing research area with many applications. Such include noise monitoring in smart cities [7], where through SED and analysis in noisy data we can achieve noise pollution reduction. Surveillance [8] is another essential application that enables systems to classify correctly the sound events, such as screams and shouts in subway trains [9], or gunshots [10] and to take critical decisions. Smart home environments [11], [12] can optimize

our life using SED systems. In autonomous driving [13], smart cars should be able to detect environmental sounds (such as sirens of ambulances) and respond appropriately. Another interesting application is automatic subtitling in different movies or series (e.g., bell ringing), making the scenario more understandable. Other applications contain health monitoring systems [14] (e.g., a person falling downstairs) and multimedia information retrieval [15].

It is not surprising that SED has been the topic of several evaluation campaigns in the literature, including the well-known DCASE Challenges [16]. Recent DCASE Challenges include the task of "Sound Event Detection and Separation in Domestic Environments" [1]. The current Thesis presents a proposed set of methods and techniques for this task.

1.2 Thesis contribution

In this Thesis, we investigate the problem of SED in domestic environments by examining different approaches to detect sound events and their time boundaries. Our approaches include state-of-the-art artificial neural network architectures and many techniques for data preprocessing, data augmentation, and data postprocessing. Specifically, our architectures contain conformer-based neural networks [17] and residual convolutional recurrent neural networks [3]. Also, we investigate the impact that sound source separation [18], [19] has on our models' performance since many studies [20] have shown very promising results. We present our approaches in detail in the following chapters.

1.3 Related work

Several methods have been developed in recent years to address SED. An early approach used hidden Markov models (HMMs) [21], where every type of sound event was modeled by a left-to-right HMM topology with three states. Other approaches used probabilistic component analysis models [22], aiming to detect overlapping acoustic events, modeling the temporal evolution of sound events. Another early approach employed the Hough transform [23].

More recent approaches though, have been focusing on neural network architectures. The first networks were feed-forward ones [24], using acoustic features of sound frames as input in order to categorize acoustic events. The performance of the elementary feed-

forward neural network was much better in polyphonic situations in contrast to the old HMM system [25].

Although feed-forward neural networks outperformed HMM systems, these networks used each input value independently of each other. In order to exploit the dependence between input values, spectrograms were used. So, convolutional neural networks (CNNs) [26] were the suitable architectures to exploit this property, treating spectrograms just like images. Some of these networks were employed to categorize isolated events [27], and others were used to categorize events of sound mixtures [28].

The drawback of a CNN is that it uses input frames within a limited range at a specific time. In contrast, a recurrent neural network (RNN) [29] uses input frames with no time-duration constraints. This fact enables RNNs to exploit more temporal information in order to provide more accurate predictions. Many approaches to SED employ RNNs [30]. The next step in the evolution of SED architectures was the combination of CNNs and RNNs, giving rise to the CRNN network [31], outperforming both CNNs and RNNs alone.

In the meanwhile, many preprocessing and postprocessing techniques were deployed in order to achieve better results in the SED task. A critical one to polyphonic SED is sound source separation (SSEP) [18], [19], [32] that is currently included in Task 4 of the DCASE challenge [1]. Having this mechanism separate sound sources, enables the SED model to provide more accurate predictions.

The latest approaches in this scientific area include the use of an attention mechanism. Such architectures with attention layers are the Transformers [33] and the Conformers [4]. Specifically, neural networks based on the combination of CNNs and Transformers have shown very promising results. CNNs are responsible for capturing the local input context and the Transformers for capturing global interactions. In a similar way, a Conformer is a neural module that combines CNNs and Transformers in order to capture both local and global dependencies. According to [4] the Conformer achieves better results than a neural network with Transformers and CNNs. This experimental result was confirmed by the winner of the DCASE Challenge 2020 Task 4 [17].

1.4 Thesis structure

The rest of this Thesis is organized as follows:

- Chapter 2 describes the datasets we use for our systems development.
- Chapter 3 presents our methods in order to solve the SED problem.
- Chapter 4 describes the experimental framework and the results of our experiments.
- Chapter 5 provides the conclusions of this Thesis, a summary, and directions of future work.

Chapter 2

Description of DCASE Task 4 Data

In order to solve the problem of SED we used multiple datasets, which were allowed for the DCASE 2021 challenge Task 4, specifically: DESED [34], SINS [35], TUT-Acoustic-scenes-2017 [36], and FSD50K [37], while for the problem of SSEP the YFCC100M [38] and the FUSS [39] datasets were suggested. Details about these datasets are provided in Section 2.1.

All mentioned datasets include audio recordings that follow the (wav) audio file format standard. These audio recordings were sampled either at 44.1 kHz or at 16 kHz. Also, some of these datasets contain files with labels. These labels are either weak labels with no timestamps or strong labels with timestamps. Details about labeling are provided in Section 2.2.

After describing the datasets and their labels in Sections 2.1 and 2.2, respectively, in Section 2.3 we present the development set that includes the training set and the validation set. In Section 2.4 we discuss the evaluation set, and in Section 2.5 the data distribution of the training, validation, and evaluation sets.

2.1 Datasets

Table 2.1 below provides useful information about the datasets that are used. As shown in the table, column one lists the names of the datasets (DESED, SINS, TUT, FUSS, FSD50K, YFCC100M). Column two tabulates the corresponding subset name if the subset exists. Then, the subset type is presented, followed by the dataset usage (training/validation/evaluation), the subset annotations, the event type, and finally the original sampling rate of audio recordings.

Table 2.1: Datasets information (Table from [1]).

Dataset	Subset	Type	Usage	Annotations	Event type	Sampling frequency
DESED	Real:weakly labeled	Recorded soundscapes	Training	Weak labels(no timestamps)	Target	44.1 kHz
	Real: unlabeled	Recorded soundscapes	Training	No annotations	Target	44.1 kHz
	Real: validation	Recorded soundscapes	Validation	Strong Labels(with timestamps)	Target	44.1 kHz
	Real Public evaluation	Recorded soundscapes	Evaluation	Strong labels (with timestamps)	Target	44.1 kHz
	Synthetic training	Isolated events + synthetic soundscapes	Training/ validation	Strong labels (with timestamps)	Target	16 kHz
	Synthetic evaluation	Isolated events + backgrounds	Evaluation	Event level labels (no timestamps)	Target	16 kHz
SINS		Background	Training validation	No annotations	N/A	16 kHz
TUT Acoustic scenes 2017, development dataset		Background	Training validation	No annotations	N/A	44.1 kHz
FUSS dataset		Isolated events + synthetic soundscapes	Training validation	Weak annotations from FSD50K(no timestamps)	Target and non-target	16 kHz
FSD50K dataset		Isolated events + recorded soundscapes	Training validation	Weak annotations (no timestamps)	Target and non-target	44.1 kHz
YFCC100M dataset		Recorded soundscapes	Training validation	No annotations	Sound sources	44.1 kHz

• DESED dataset

The DESED dataset [34] was designed for the task of SED in domestic environments, containing 10-sec audio clips. These audio clips are separated into two categories. The first

one contains audio clips recorded in domestic environments, and the second one synthesized data that simulate a domestic environment. Finally, this dataset contains sound events that belong to the following ten classes: alarm/bell/ringing, blender, cat, dog, dishes, electric shaver/toothbrush, frying, running water, speech, vacuum cleaner. These are also the classes that we focus on.

• **SINS database**

The SINS database [35] contains a one-week continuous recording (approximately 200 hours) of one person living in a vacation home. The sound events of this dataset were annotated manually. It is used as a background dataset.

• **TUT Acoustic Scenes 2017**

TUT Acoustic Scenes 2017 [36] is another background dataset that includes recordings of different acoustic scenes, such as bus (vehicle's sound) and train. Each recording is between 3 and 5 minutes long and is split to 10-sec audio clips.

• **FSD50K dataset**

FSD50K [37] is an open dataset that includes approximately 51k Freesound clips [40]. The length of each audio clip is between 0.3 and 30 sec. This dataset also contains both labeled and unlabeled data. The labeled data are manually annotated and belong to one of the 200 AudioSet classes [41].

• **Free Universal Sound Separation (FUSS) dataset**

The Free Universal Sound Separation (FUSS) dataset [39] contains sound mixtures and source-level references. It is mainly used for the problem of sound source separation. Its audio data and labels are derived from a subset of the FSD50K dataset. All mixtures last 10 sec and contain 1 to 4 sound sources. Each mixture contains one background source, which is active for the whole duration of 10 sec.

• **YFCC100M dataset**

YFCC100M [38] is a multimedia dataset containing approximately 99.2 million photos and 0.8 million videos. This dataset is related to the task of sound source separation.

2.2 Labels information

The datasets contain strongly labeled, weakly labeled, and unlabeled data. The structure of the labeled/annotated data is described analytically below.

2.2.1 Weak annotations

The weak annotations are provided in tab-separated files with the following format [1]:

[filename (string)][tab][event_labels (strings)]

For instance, in annotation:

```
YL7bzKI26Pek_190.000_200.000.wav Alarm_bell_ringing,Speech
```

- YL7bzKI26Pek_190.000_200.000.wav is the filename. Indexes 190.000 and 200.000 inform us that this clip lies between 190 and 200 sec of the corresponding Youtube video.
- Alarm_bell_ringing and Speech are the sound classes that exist within the sound clip.

2.2.2 Strong annotations

The strong annotations are also provided in tab-separated files. Their format is structured in the following way [1]:

[filename (string)][tab][onset (in sec) (float)][tab][offset (in sec) (float)][tab][event_label (string)]

For example, in annotation:

```
1000.wav 4.508 5.273 Speech
```

- 1000.wav is the filename.
- 4.508 is the onset time measured in sec.
- 5.273 is the offset time in sec.
- Speech is the sound class of the corresponding event.

2.3 Development dataset

In Figure 2.1, we show the development dataset. It is separated into two categories: data for the SED problem and data for the SSEP problem. In each of these categories, the data are also separated into training and validation sets. In the SED problem, the training set comprises real unlabeled data, weakly labeled data, and strongly annotated synthetic data. Instead, the validation set contains only strongly labeled data. Additionally, in the SSEP problem, there is the corresponding split into a training and a validation set.

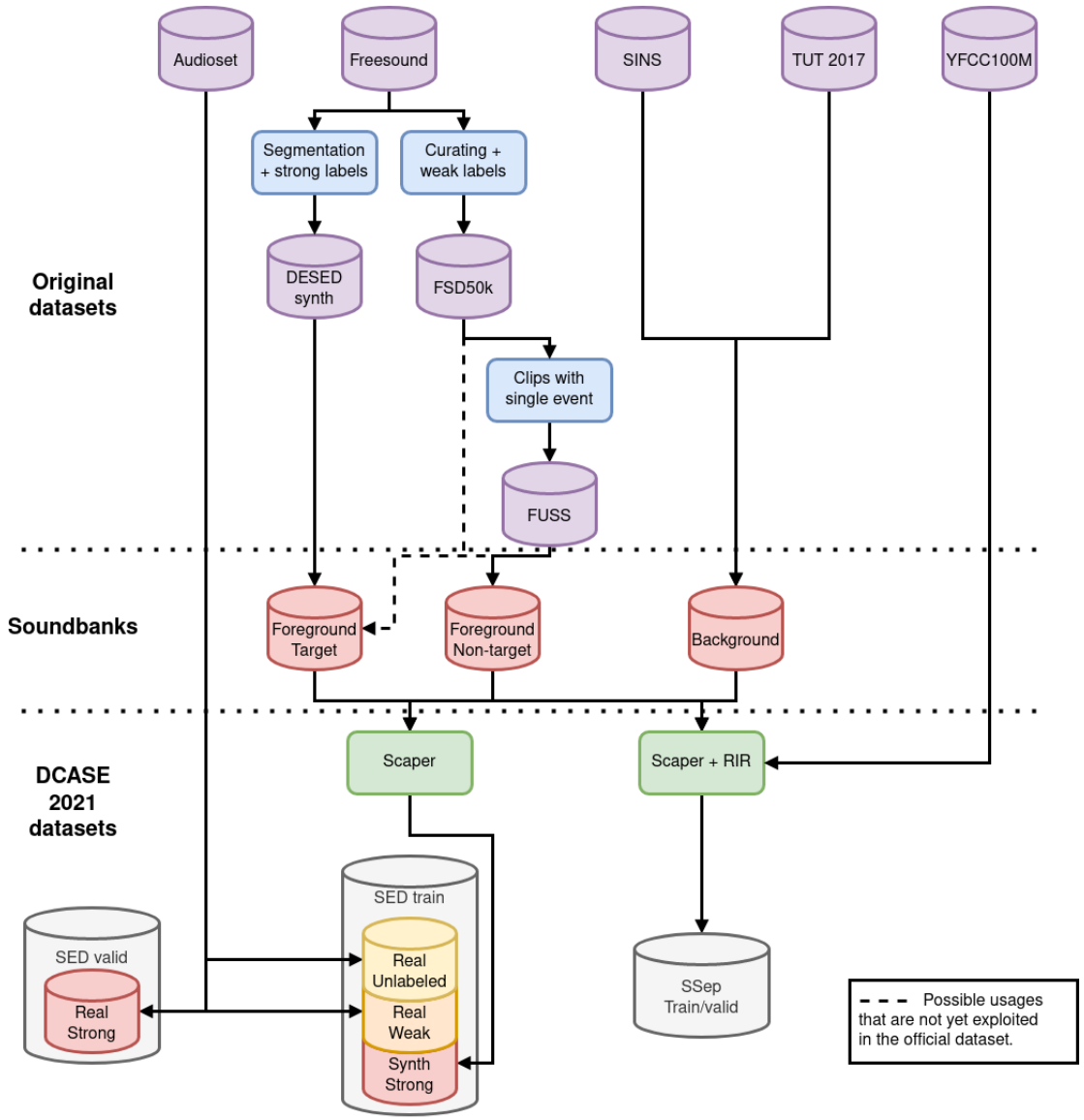

Figure 2.1: Development dataset (figure from [1]).

2.4 Evaluation dataset

The evaluation dataset is also separated according to the problem of SED or SSEP. In the first case, the evaluation set contains the public evaluation dataset, the private evaluation dataset, and the synthetic evaluation dataset. Specifically, the last two datasets are undisclosed in order to be used by the task coordinators to evaluate the submitted models of each participant. In our case, we use only the public evaluation dataset. Finally, in the SSEP problem, there is the corresponding dataset for evaluation purposes.

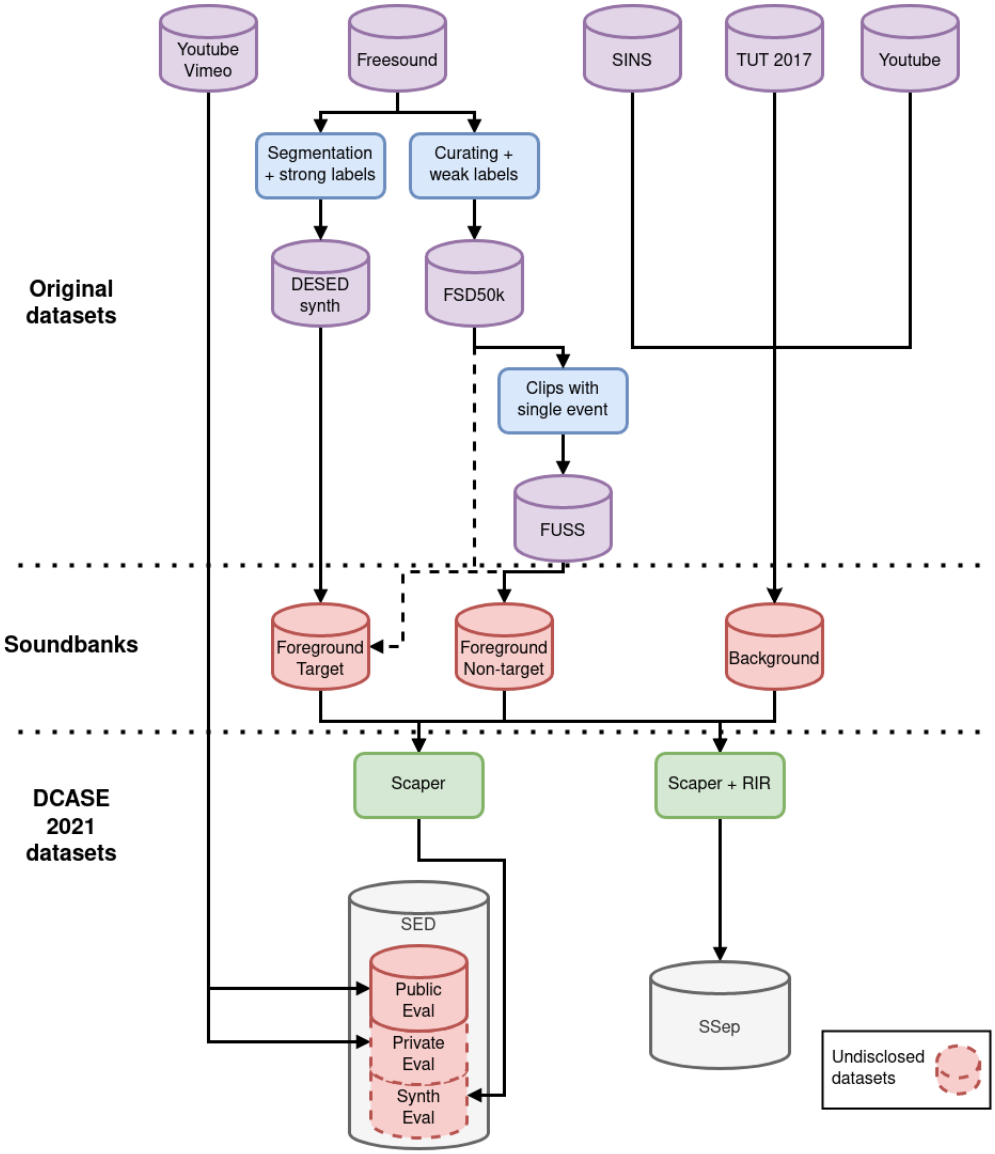

Figure 2.2: Evaluation dataset (figure from [1]).

2.5 Data distribution

Table 2.2 shows the data distribution of the training set, validation set, and evaluation set for DCASE 2021 Challenge Task 4. As presented in the table, DCASE 2021 Challenge Task 4 contains five final datasets: a strongly labeled synthetic dataset, a weakly labeled dataset, an unlabeled dataset without timestamps, a validation dataset, and a public evaluation dataset. As shown in Figure 2.1, the weakly labeled, the unlabeled, and the validation datasets were taken from the Audioset dataset [41]. The validation set was annotated with strong labels, with timestamps (obtained by human annotators). Instead, the strongly labeled dataset was generated using Scaper soundscape synthesis and augmentation library [42]. As illustrated in Figure 2.2, the sound clips of the public evaluation dataset were extracted from YouTube and Vimeo, containing strong labels with timestamps. The audio clips are 10000, 1578, 14412, 1168, and 692 in the strongly labeled, weakly labeled, unlabeled, validation, and public evaluation dataset, respectively. Each audio clip has a maximum duration of 10 sec.

Table 2.2: Data distribution of the training set, validation set, and evaluation set.

Set	Dataset	Audio clips
Training Set	Strongly labeled dataset	10000
	Weakly labeled dataset	1578
	Unlabeled dataset	14412
Val set	Validation dataset	1168
Eval set	Public evaluation dataset	692

Chapter 3

Our Methodology

This chapter presents the methods that we deployed to tackle the problem of SED. Firstly, we introduce the baseline systems, then the audio preprocessing techniques, and finally the following concepts: the data augmentation technique, the postprocessing technique, the RCRNN model, the Conformer-based model, the semi-supervised learning technique, the ensemble learning technique, and SSEP.

3.1 Baseline systems

DCASE challenge proposes two baseline systems. The first one tackles the SED problem, and the second one tackles the same problem using SSEP as a preprocessing step.

3.1.1 Sound event detection baseline system

The SED baseline system is an improvement of the DCASE 2020 baseline [2]. It uses a mean-teacher model [43] that contains two submodels: a student model and a teacher model, both with the same architecture. The student model is the only model that is trained on the strongly and weakly labeled data. Its loss function (in our case, the binary cross-entropy (BCE)) is calculated at the clip level for the weakly labeled data and at the frame level for the strongly labeled data. Instead, the teacher model is not trained, and its weights are the exponential moving average of the student model's weights. In the training phase, it takes as input the same data as the student model but with added Gaussian noise. Its role is to help the student model get trained through a consistency loss function (in our case, the mean square error (MSE)) for both strong and weak predictions that come through unlabeled, weakly, and

strongly labeled data. In Figure 3.1 below, this mean-teacher model is shown.

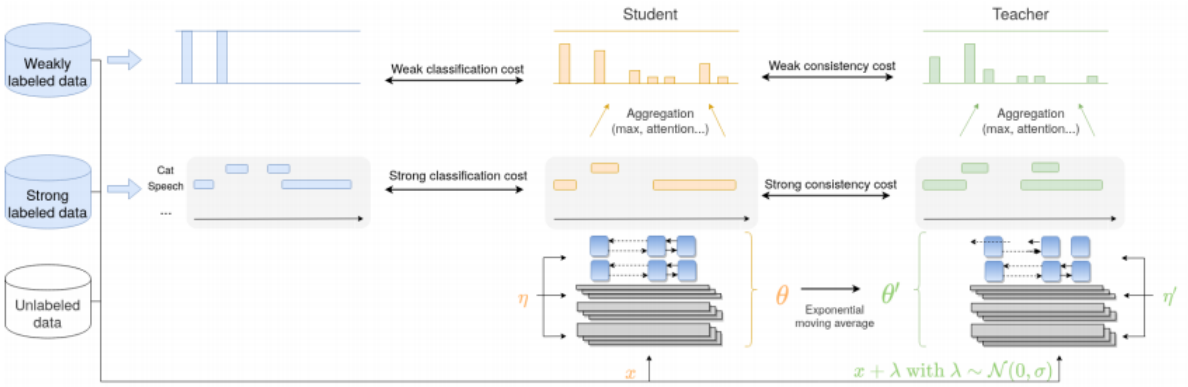

Figure 3.1: Mean-teacher model as the SED baseline (figure from [2]).

The architecture for both student and teacher models is a CRNN. Table 3.1 shows the network architecture and the hyperparameters of this CRNN. As shown in the table, the input features of the network are mel-spectrograms with 128 mel bands. The CNN part of the CRNN model is composed of 7 convolutional layers with [16, 32, 64, 128, 128, 128, 128] kernels per layer, respectively. Each convolutional layer has 3x3 kernels with a stride of 1x1, padding 1x1, and it is followed by batch normalization, ReLU activation, and an average pooling layer with [[2,2], [2,2], [1,2], [1,2], [1,2], [1,2], [1,2]] pool sizes per layer, respectively. After finishing all the convolutional blocks, the 128x156x1-dimensional output is processed by a recurrent block. The recurrent block is composed of two bidirectional gated recurrent units (BGRUs), and its role is to capture the temporal context information. The 256x156-dimensional output is applied to a fully connected layer and then processed by a sigmoid activation function, resulting in an 156x10-dimensional output related to strong predictions. Then, in a parallel way, the output of the recurrent block is processed by an attention pooling layer that is the multiplication between the output of the fully-connected layer with sigmoid activation and the output of a fully-connected layer with softmax activation, resulting in an 1x10-dimensional output. This output is related to weak predictions of an audio clip, where 10 indicates the number of sound event classes that the system detects.

Table 3.1: The CRNN architecture of the baseline system (modified table from [3]).

Name	Layers	Output shape
Input layer	Input: log-mel spectrogram	1×626×128
Convolutional block	3 × 3, Conv2D, @16, BN, ReLU 2×2 average pooling layer	16×313×64
	3 × 3, Conv2D, @32, BN, ReLU 2×2 average pooling layer	32×156×32
	3 × 3, Conv2D, @64, BN, ReLU 1×2 average pooling layer	64×156×16
	3 × 3, Conv2D, @128, BN, ReLU 1×2 average pooling layer	128×156×8
	3 × 3, Conv2D, @128, BN, ReLU 1×2 average pooling layer	128×156×4
	3 × 3, Conv2D, @128, BN, ReLU 1×2 average pooling layer	128×156×2
	3 × 3, Conv2D, @128, BN, ReLU 1×2 average pooling layer	128×156×1
Recurrent block	(128 BiGRU cells) ×2	256×156

Since the network's outputs are probabilities, thresholding is used, where the threshold value for a class to be active is set to 0.5. Finally, the model is trained for 200 epochs using the Adam optimizer [44], and the best epoch on the validation set is kept. Meanwhile, the learning rate is set according to the ramp-up strategy, reaching the highest value (0.001) after 50 epochs (warm-up period). For post-processing a median filter is used on 0.45 sec approximately (or 27 frames using a 16 kHz sampling rate).

3.1.2 Sound event detection with source separation baseline system

Figure 3.2 below depicts the SED with SSEP baseline system. It consists of two fundamental blocks. The first block contains a pre-trained SSEP model. In our case, this model is an ”improved time-dilated convolutional network” (TDCN++) [19] and is trained in an unsupervised way with MixIT on the YFCC100M dataset. More details about this model can be found in [18]. The second block is responsible for tackling the SED problem. It uses a pre-trained SED model (in our case, the SED baseline system). This SED model is fine-tuned, taking the separated data produced from the SSEP model as input. The final predictions arise by ensembling the original SED model with the fine-tuned SED model. Ensembling is performed using the weighted average of the predictions of these two models.

Figure 3.2: The SED+SSEP baseline system (figure from [1]).

3.2 Audio preprocessing

First, we resample the audio clips at 16,000 Hz. Then, we perform the feature extraction technique. The features that are provided as input to all implemented models are mel-spectrograms. After a number of experiments, we found that the best setup for all of our implemented models is this with 128 mel bins and frequencies between 0 and 8 kHz. The mel-spectrogram features are obtained from the Short-time Fourier transform (STFT) calculated on N-sample windows with 256-sample hop size. In Chapter 4, we present the results of the experiments that we conducted in order to find the best-performing model, trying different values for parameter N. Finally, for each mel-bin the mel-spectrograms are normalized using the global

mean and the standard deviation of the value of the bin.

3.3 Data augmentation

For data augmentation, we used Gaussian noise and mixup [45]. These techniques augment the input data, making the models more robust against unseen data and help these models avoid overfitting. The Gaussian noise is added to the normalized mel-spectrogram using a randomly selected signal-to-noise ratio within the range of 6 and 30 dB. On the other hand, the mixup technique creates pseudo-data using the following equations:

$$\bar{x} = \lambda x_i + (1 - \lambda) x_j \ , \tag{3.1}$$

$$\bar{y} = \lambda y_i + (1 - \lambda) y_j \ , \tag{3.2}$$

where (x_i, y_i) and (x_j, y_j) are two randomly selected examples from our training set and $\lambda \in$ [0,1]. In our case, we randomly choose the λ value by sampling from a beta distribution with parameters $\alpha = \beta = 0.2$.

3.4 Postprocessing technique

A median window [46] is used in order to postprocess the output data. This window post-processes the outputs at the frame level. For each class, the output of the network is a probability that indicates the chances of detecting the corresponding class. Then, using thresholding, a detection indicator is calculated: it is set to 0, if the probability is less than 0.5, and 1 otherwise. Then, median filtering is performed along the time axis to smooth this binary sequence. This happens in order to avoid false detections. For instance, ”running water” is improbable to follow the pattern ”detected-notDetected-detected” in three consecutive frames. In order to select the optimal length (in frames) for this median window, we tried different values (3, 7, 15, 25, 40, 50). In Chapter 4, we present corresponding experiments with these filter lengths.

3.5 RCRNN model

Our residual convolutional recurrent neural network (RCRNN) was inspired by Nam et al. [3]. In contrast to [3], we implemented our neural network without the Convolutional Block Attention Module [47] since we observed better results without it. In Figure 3.4 below we depict our architecture. As shown in the figure, the input layer, the recurrent block, and the prediction block are the same as the baseline model. Instead, the CNN part of our network is composed of one stem block and five residual convolutional blocks. The stem block consists of two convolutional blocks with 16 and 32 kernels per block, respectively. Each of these convolutional blocks has 7x7 kernels with a stride of 1x1, padding 3x3, and it is followed by batch normalization, ReLU activation, and an average pooling layer with [2,2] pool size. The structure of the residual convolutional block is presented in Figure 3.3 below. Each residual convolutional block consists of two convolutional layers, each of them followed by batch normalization and ReLU activation. Then, after the residual connection, there is an average pooling layer with [1,2] pool size.

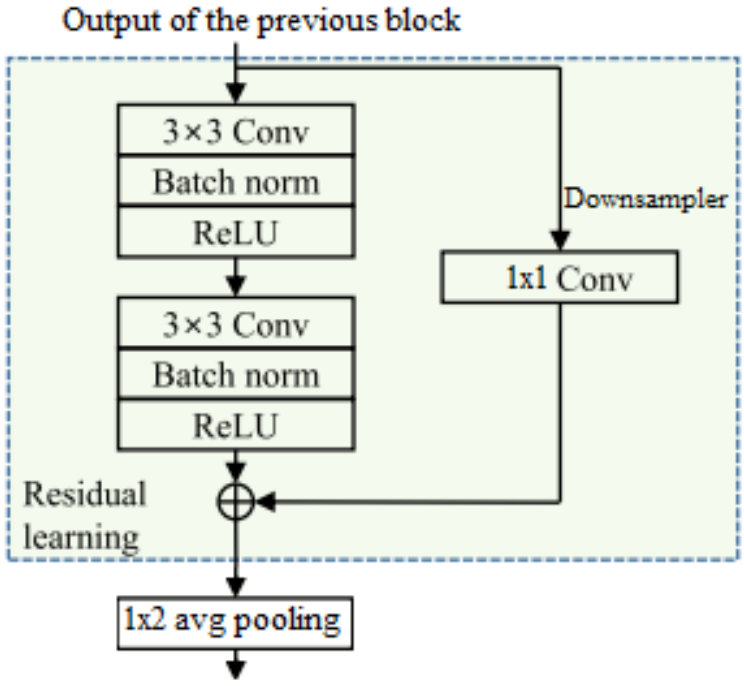

Figure 3.3: The structure of the residual convolutional block (modified figure from [3]).

The downsampler (which is a convolutional layer with 1x1 kernels), as presented in Figure 3.3, is applied to the first two residual convolutional blocks due to differences in the input and output dimension.

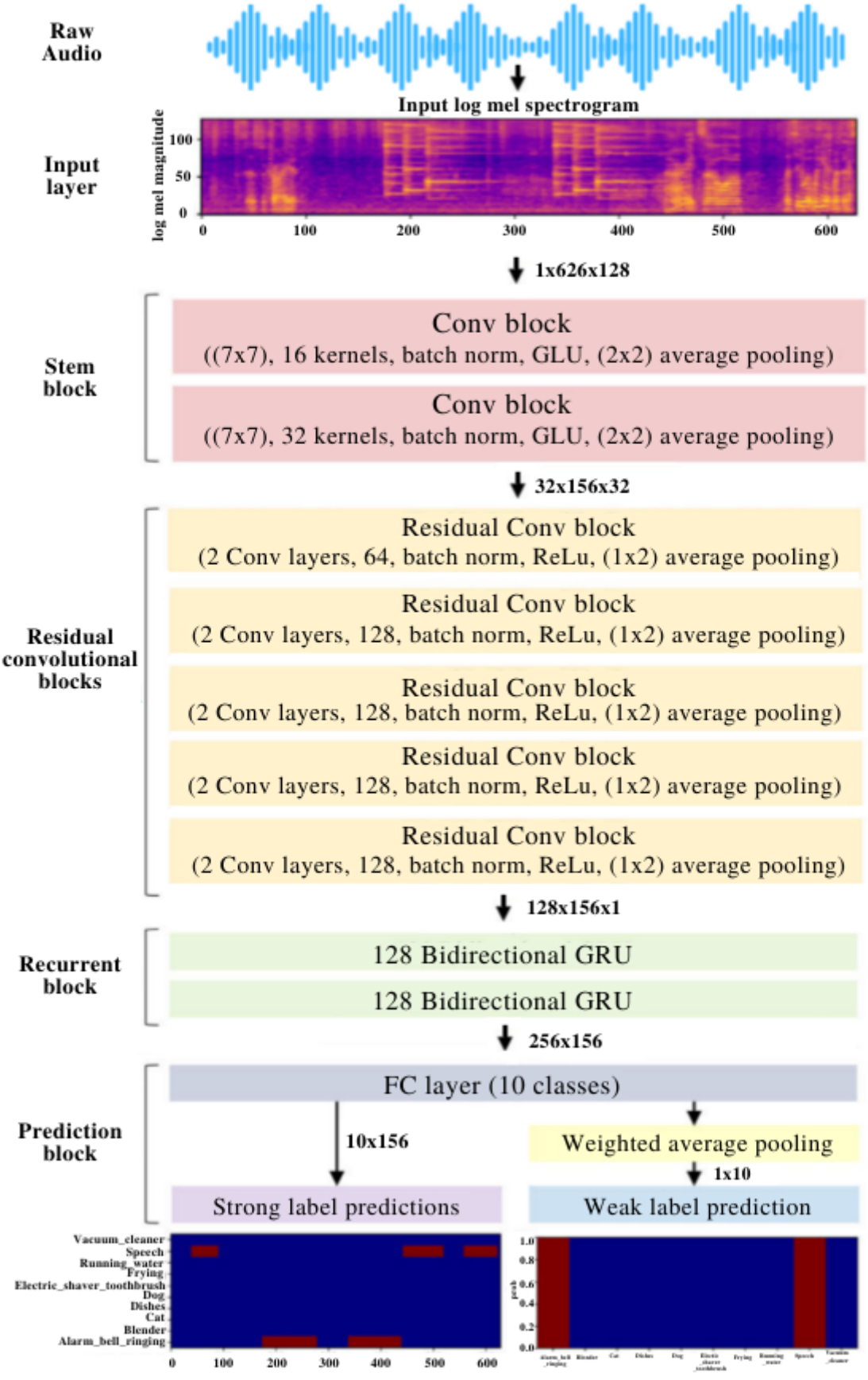

Figure 3.4: The architecture of our RCRNN model (modified figure from [3]).

3.6 Conformer-based Model

Our Conformer-based neural network is another architecture that we implemented since it combines CNNs and Transformers to capture both local and global input information of an audio sequence in a very efficient way. This network contains three fundamental components: a CNN, a sequence of Conformer blocks, and a prediction block. The CNN and the prediction block have the same architecture as the RCRNN neural network. In Figure 3.9 we depict the structure of the Conformer-based neural network, and in Section 3.6.1 we discuss the structure of a Conformer block.

3.6.1 Conformer block

The Conformer block is responsible for capturing both local and global context input information. The ”Conformer: Convolution-augmented Transformer for Speech Recognition” paper by Gulati et al. [4] inspired our implementation. This block includes feed-forward modules, a multi-head self-attention module, a convolution module, and a normalization layer, as demonstrated in Figure 3.5. The convolutional module captures the local context information of the sound spectrogram, and the multi-head self-attention (MHSA) module captures the global context information. Also, in the same figure, it can be seen how the modules are connected. Moreover, if x is the input to the Conformer block, then the output y can be described mathematically as follows:

$$\begin{aligned} x_1 &= x + \frac{1}{2}FFN(x) \\ x_2 &= x_1 + MHSA(x_1) \\ x_3 &= x_2 + CONV(x_2) \\ y &= LayerNorm(x_3 + \frac{1}{2}FNN(x_3)) \end{aligned} \tag{3.3}$$

where FFN is the feed-forward module, MHSA is the multi-head self-attention module, and CONV is the convolution module.

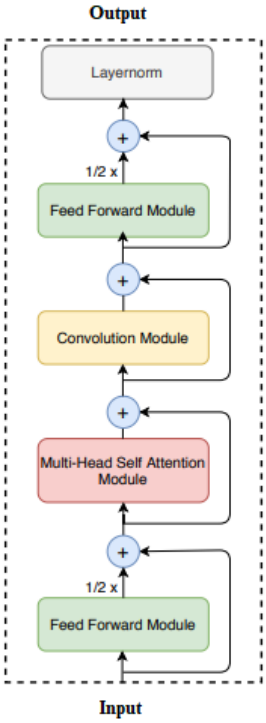

Figure 3.5: A Conformer block (figure from [4]).

3.6.1.1 Feed-forward module

The first feed-forward module is placed before the MHSA module and the second one after the convolution module. This module contains: a normalization layer, two linear layers, a Swish activation function, and two dropout layers that are used for regularization purposes. Also, layer normalization is applied after the second feed-forward module, as depicted in Figure 3.5. Figure 3.6 demonstrates this feed-forward module.

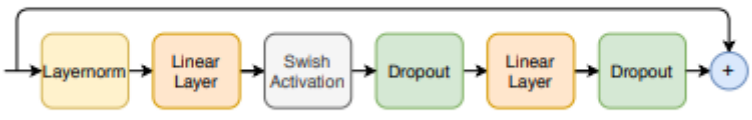

Figure 3.6: The feed-forward module (figure from [4]).

3.6.1.2 Multi-head self-attention module

The MHSA module uses the relative sinusoidal positional encoding scheme that is described in [48]. Using this technique, the self-attention module generalizes better on data with different input lengths, and the resulting encoder is more robust. Moreover, this module contains pre-norm residual units [49], [50] and a dropout layer to avoid overfitting. In Figure 3.7 below we depict this MHSA module.

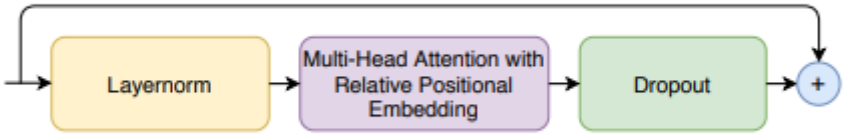

Figure 3.7: The multi-head self-attention module (figure from [4]).

3.6.1.3 Convolution module

In Figure 3.8 we depict the convolution module that is based on [51]. It starts with a normalization layer and a pointwise convolution with a gated linear unit activation function, followed by a 1-D depthwise convolution layer and a batch normalization layer to help us train a very deep model. Finally, there is a Swish activation function, a pointwise convolution, and a dropout layer.

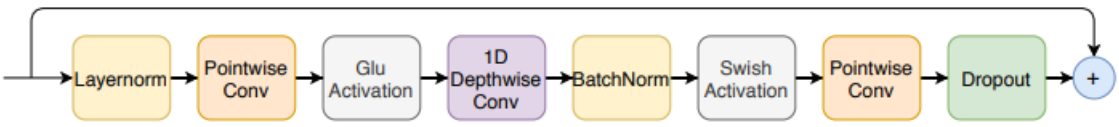

Figure 3.8: The convolution module (figure from [4]).

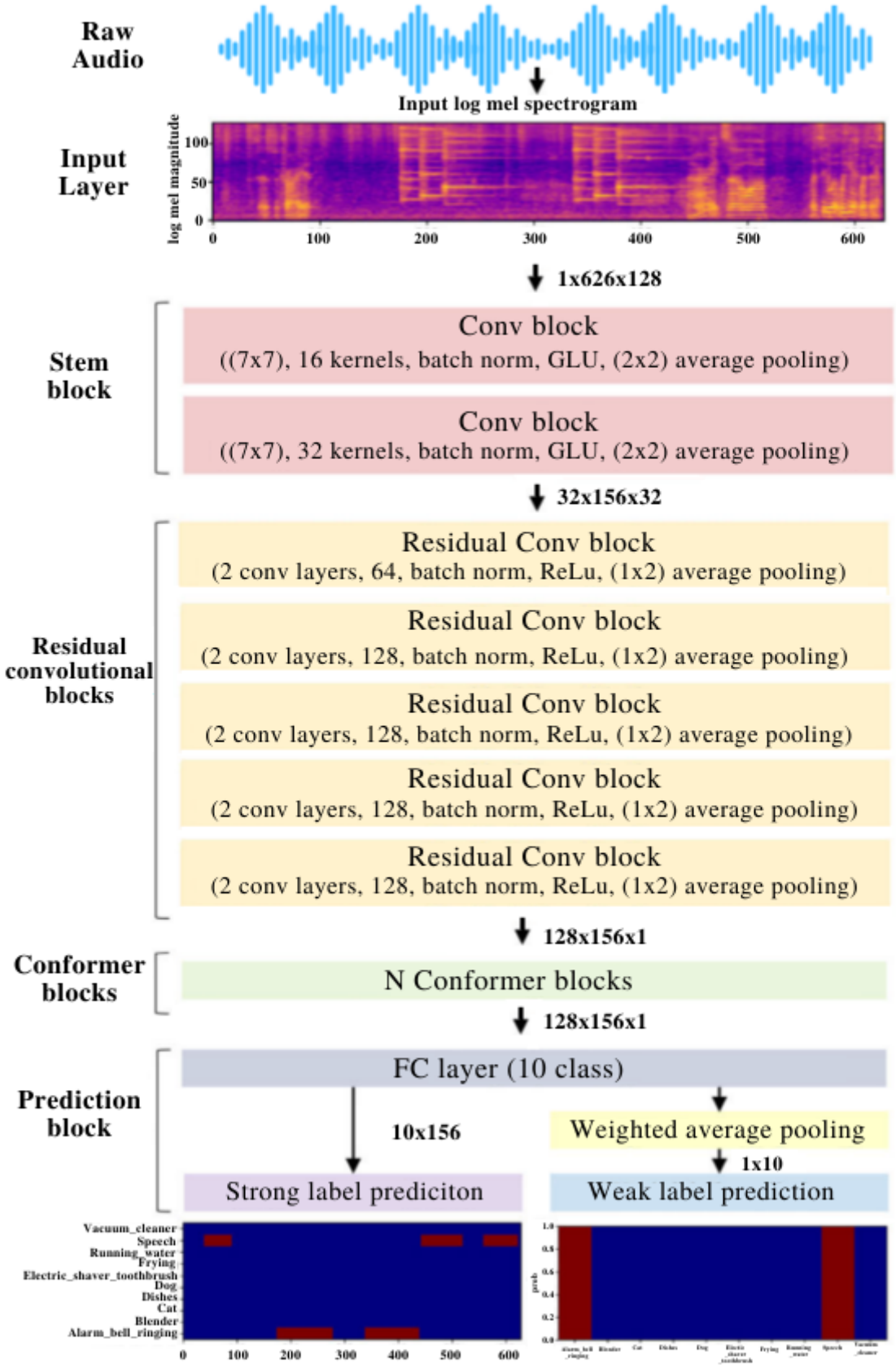

Figure 3.9: The architecture of our Conformer-based neural network (modified figure from [3]).

3.7 Semi-supervised learning

To further improve the performance of our SED models, exploiting the unlabeled data, we employed the mean-teacher semi-supervised technique [43] in the same way as the baseline system. This technique belongs to the semi-supervised learning methods since it uses both labeled and unlabeled data during training. We described this technique in Section 3.1.1 in detail. The hyperparameters for this method for all of our implemented models are set as follows:

Hyperparameter name	value
Self-supervised mean-teacher loss	Binary cross entropy (BCE)
Consistency criterion	Mean square error (MSE)
Consistency cost	2.0
Exponential moving average (EMA) factor	0.999

3.8 Ensemble learning

Ensemble learning [52] is another technique we implemented to improve the performance of our models further. This method works by fusing different models. In our case, these models are SED models trained with different hyperparameters. The final predictions of the ensemble model arise as the average of the predictions of each model. In Chapter 4, we describe in detail the SED models that compose the ensemble models. Figure 3.14 below illustrates the general structure of an ensemble model.

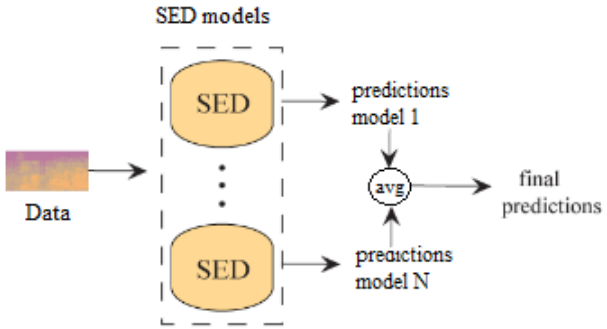

Figure 3.10: Ensemble model architecture (modified figure from [1]).

3.9 Source Separation

The final technique that is used to further improve the performance of our SED models is SSEP. This technique separates the overlapping sound events making the detection process easier. We use the same SSEP pre-trained model as the baseline system. The architecture of the SED system that uses SSEP as a preprocessing step was described in Section 3.1.2. We use the same architecture, but we apply our implemented RCRNN and Conformer-based neural network instead of the SED baseline model. SSEP is used by first separating the sound mixtures and then by applying SED to the isolated soundtracks. The decisions obtained for the separated sound events may be more accurate than those for the overlapping sound events. In Chapter 4, we will see that SSEP is a helpful technique for some models, but for some others it has negative effects.

Chapter 4

Experimental Results

This chapter presents the experimental setup, as well as the software and hardware framework we used in order to run our experiments, the evaluation metrics, and finally the experiments we ran in order to tune and evaluate our models.

4.1 Experimental setup

All of our implemented models were trained for 200 epochs using the Adam optimizer [44], and the best epoch on the validation set was kept. The learning rate was set according to the ramp-up strategy, reaching the highest value (0.001) after 50 epochs (warm-up period). The two different self-supervised mean-teacher loss functions for the RCRNN and for the Conformer-based neural network were the MSE and BCE, respectively. The batch size, after experiments, was set to 48 for all of our implemented models, containing 12 strongly labeled, 12 weakly labeled, and 24 unlabeled training examples.

4.1.1 Software and hardware framework

The experiments ran on two different Google Colab Pro environments, using either a Tesla T4 (16 GB), or a Tesla P100-PCIE (16 GB). For storage we used Google Drive (2 TB). All of our code was written in the python language and run on Google's Colaboratory. All of our models were implemented in PyTorch [53], except the SSEP baseline model that was implemented in TensorFlow [54].

4.2 Evaluation metric

In order to evaluate our models, we used the polyphonic sound detection score (PSDS) [55]. The PSDS is the normalized area under the polyphonic sound detection receiver operating characteristic (PSD-ROC) curve. Mathematically, it can be described as follows:

$$PSDS = \frac{1}{e_{max}} \int_0^{e_{max}} r(e)de \ , \tag{4.1}$$

where $r(e)$ is the PSD-ROC curve and e_{max} is the maximum effective false-positive rate (eFPR) value. The eFPR and effective true-positive rate (eTPR) are defined as:

$$eFPR : e_c^* \triangleq R_{FP,c}^* + \alpha_{CT} \frac{1}{|C| - 1} \sum_{\substack{\hat{c} \in C \\ \hat{c} \neq c}} R_{CT,c,\hat{c}}^* \ , \tag{4.2}$$

$$eTPR : r(e) \triangleq \mu_{TP}(e) - \alpha_{ST} * \sigma_{TP}(e) \ , \tag{4.3}$$

where $R_{FP,c}^*$ is the false positive (FP) rate for the c-th sound class and $R_{CT,c,\hat{c}}^*$ is the cross-trigger (CT) rate. Specifically, $R_{FP,c}^*$ corresponds to the rate of incorrectly predicting a sound event type c as $\hat{c}$ and $R_{CT,c,\hat{c}}^*$ is related to the rate of wrongly substituting a sound event type c as $\hat{c}$. Also, α_{CT} is a weighting parameter that represents the effect of CTs over the overall false detections. In (4.3), the $\mu_{TP}(e)$ and $\sigma_{TP}(e)$ are the mean and standard deviation of the true positive (TP) rate across all sound classes, respectively, and α_{ST} is another weighting parameter that controls the instability cost between all sound classes. Mathematically, $\mu_{TP}(e)$ and $\sigma_{TP}(e)$ are defined as:

$$\mu_{TP} = \frac{1}{|C|} \sum_{c \in C} r_{TP,c} \ , \tag{4.4}$$

$$\sigma_{TP} = \sqrt{\frac{1}{|C|} \sum_{c \in C} (r_{TP,c} - \mu_{TP})^2} \ . \tag{4.5}$$

In order to calculate the values of this score, we use 50 linearly distributed operation points between 0.01 and 0.99. According to Task 4 of DCASE Challenge 2021, to better understand the behavior of our models, we evaluated them for two different scenarios that bring out different system properties.

4.2.1 Scenario 1

In scenario 1, our system needs to react fast during the process of event detection. High values of PSDS in this scenario indicate that our system reacts fast upon an event detection (e.g., to trigger an alarm). A significant characteristic of this scenario is the localization of the sound event. The PSDS parameters that simulate this scenario are:

- Detection Tolerance criterion (DTC): 0.7
- Ground Truth intersection criterion (GTC): 0.7
- Cost of instability across class (a_{ST}): 1
- Cost of CTs on user experience (a_{CT}): 0
- Maximum FP rate (e_{max}): 100

4.2.2 Scenario 2

In scenario 2, our system must avoid incorrect class prediction. The reaction time in this scenario is not so crucial. The PSDS parameters that simulate this scenario are:

- DTC: 0.1
- GTC: 0.1
- a_{ST}: 1
- Cross-Trigger Tolerance criterion: 0.3
- a_{CT}: 0.5
- e_{max}: 100

4.3 Experiments

Multiple experiments have been conducted for tuning and evaluation processes. We used the development set (training set + validation set) for the tuning process, and for the evaluation process, we used the public evaluation dataset. The following subsections present the experiments that we conducted to tune the RCRNN and the Conformer-based neural network. Then, we present the ensemble models and the models that use SSEP as a pre-processing step. Finally, the fine-tuned models are evaluated using the public evaluation dataset.

4.3.1 RCRNN tuning

The tuning process for the RCRNN aims to enhance the system's performance using different hyperparameter values, keeping these that provide the highest PSDS in scenario 1, regardless of the PSDS in scenario 2.

4.3.1.1 Window size tuning

This experiment aims to find the best STFT window size (in samples). As shown in Table 4.1, we investigate different sizes, such as 256, 512, 1024, 2048, and 4096. It is observed that the PSDS in scenario 1 for the windows with size 512, 1024, and 2048 is almost the same, but the window with 2048 samples size achieves the highest PSDS that is equal to 0.356. It is also the same window size that achieves the highest PSDS in scenario 2.

Table 4.1: PSDS in scenarios 1 and 2 using different window sizes for the RCRNN.

Window size	PSDS scenario 1	PSDS scenario 2
256	0.334	0.555
512	0.354	0.564
1024	0.355	0.561
2048	**0.356**	**0.565**
4096	0.345	0.548

4.3.1.2 Median filter length tuning

Having found the best window size for the RCRNN model, in this experiment, we try to find the best median filter length (in frames). As depicted in Figures 4.1 and 4.2, we investigate different lengths, such as 3, 7, 15, 25, 40, and 50. In scenario 1, as shown in the first barplot, the model that achieves the highest PSDS is this with 7 frames median filter length, achieving a score that is equal to 0.356. Instead, in scenario 2, as illustrated in the second barplot, the model that achieves the highest PSDS is this with 25 frames median filter length. Since the tuning process for the RCRNN aims to find the hyperparameter values that achieve the highest PSDS in scenario 1, the 7-frame median filter is kept.

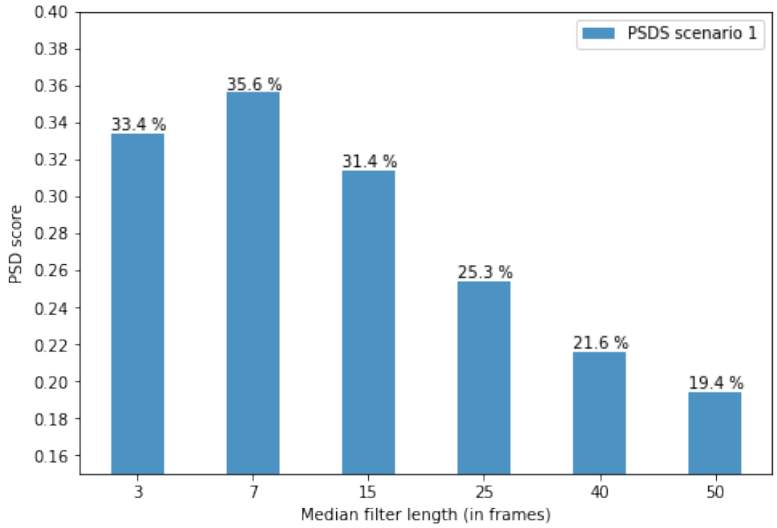

Figure 4.1: PSDS in scenario 1 using different median filter lengths for the RCRNN.

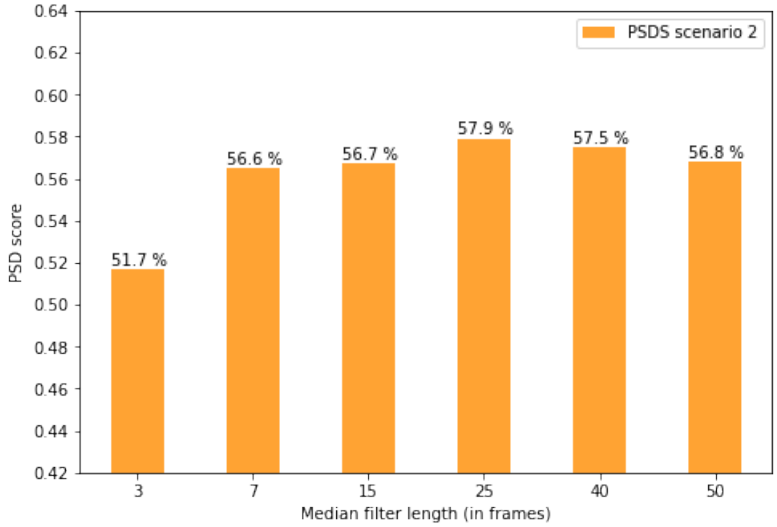

Figure 4.2: PSDS in scenario 2 using different median filter lengths for the RCRNN.

4.3.2 Conformer-based neural network tuning

The tuning process of our Conformer-based neural network aims to increase the PSDS in scenario 2, regardless of the PSDS in scenario 1. Tuning is performed investigating different hyperparameter values. Our experiments are presented below.

4.3.2.1 Window type tuning

This experiment aims to find the best STFT window type. As shown in Table 4.2, we try the Hamming, Hanning, Barlett, Blackman, and Kaiser windows. It is observed that the model with the Blackman window achieves the highest PSDS in scenario 1 (0.194) and the model with the Hamming window in scenario 2 (0.561). Since the tuning process for the Conformer-based neural network aims to achieve the highest PSDS in scenario 2, the Hamming window is kept.

Table 4.2: PSDS in scenarios 1 and 2 using different window types.

Window type	PSDS scenario 1	PSDS scenario 2
Hamming	0.189	**0.561**
Hanning	0.189	0.552
Barlett	0.181	0.542
Blackman	**0.194**	0.556
Kaiser	0.121	0.559

4.3.2.2 Window size tuning

Knowing that the model with the Hamming window achieves the highest PSDS in scenario 2, in this experiment we try to find the best STFT window size. As with the RCRNN, we try different sizes, such as 256, 512, 1024, 2048, and 4096. As shown in Table 4.3, in scenario 1 the model that achieves the highest PSDS is this with a 1024-size window. Instead, in scenario 2, the model with a 512-size window outperforms others. The window with 512 samples is kept.

Table 4.3: PSDS in scenarios 1 and 2 using different window sizes for the Conformer-based neural network.

Window size	PSDS scenario 1	PSDS scenario 2
256	0.204	0.525
512	0.209	**0.569**
1024	**0.210**	0.566
2048	0.189	0.561
4096	0.192	0.535

4.3.2.3 Median filter length tuning

Having found the best window type and the best window size, in this experiment we investigate the median filter length. As with the RCRNN, we try different lengths, such as 3, 7, 15, 25, 40, and 50. In scenario 1, as demonstrated in the first barplot (Figure 4.3), the filter that achieves the highest PSDS is this with 7 frames length, achieving a score that is equal to 0.209. On the other hand, in scenario 2, as presented in the second barplot (Figure 4.4), the filter that achieves the highest PSDS is this with 40 frames length, achieving a score that is equal to 0.681. We keep the filter with 40 frames length.

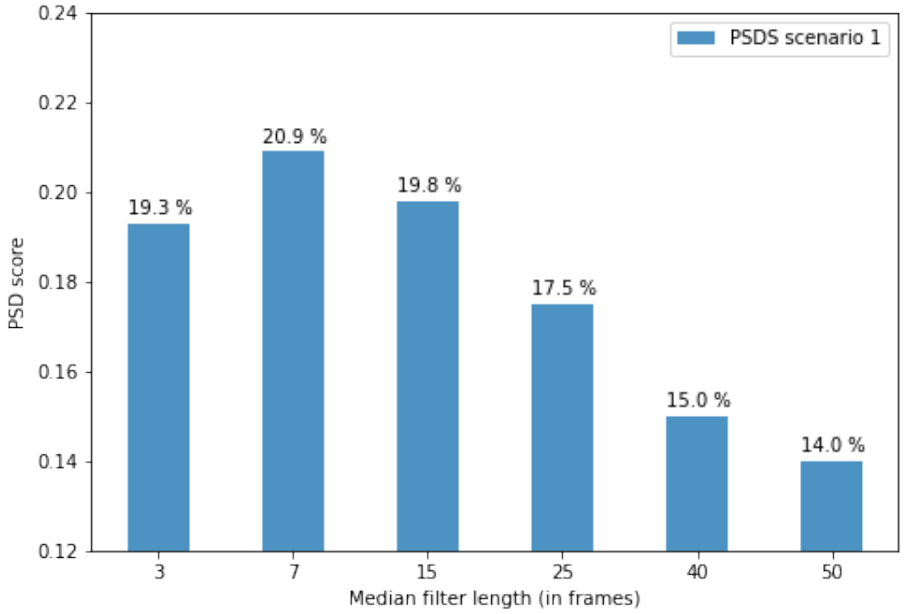

Figure 4.3: PSDS in scenario 1 using different median filter lengths for the Conformer-based neural network.

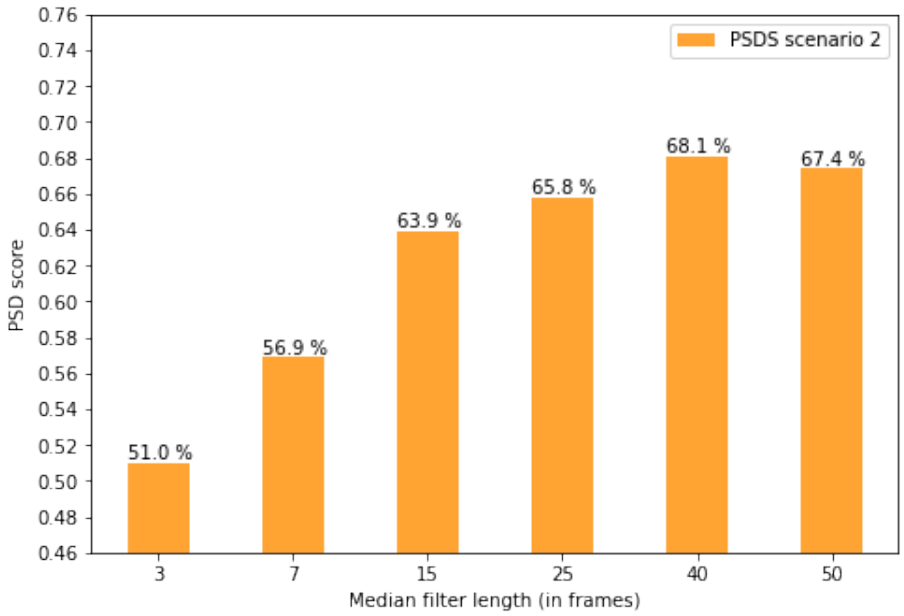

Figure 4.4: PSDS in scenario 2 using different median filter lengths for the Conformer-based neural network.

4.3.2.4 Number of Conformer blocks tuning

This experiment aims to find a suitable number of Conformer blocks, providing the best performance in scenario 2. As shown in Table 4.4, firstly, we try only 1 Conformer block, then 2, and finally 3 Conformer blocks. It is observed that the highest PSDS in both scenarios is achieved using only one Conformer block. The PSDS in scenario 1 is equal to 0.150 and in scenario 2 equal to 0.681.

Table 4.4: Tuning the number of Conformer blocks.

Number of Conformer blocks	PSDS scenario 1	PSDS scenario 2
1	**0.150**	**0.681**
2	0.127	0.678
3	0.111	0.664

4.3.3 Ensemble models tuning

In this experiment we evaluate the performance of the ensemble models. Table 4.5 presents the models that participate in each ensemble model, and Table 4.6 shows the parameters of each ensemble model, the additional number of epochs that the ensemble model fine-tuned, and the achieved PSDS in scenarios 1 and 2, respectively. Finally, the ensemble model that provides the highest PSDS in scenario 1 is the ”RCRNNbest+RCRNN1” and in scenario 2 is the ”Confbest+Conf1”.

Table 4.5: Characteristics of pretrained SED models that participate in ensemble models.

SED model	Window type	Window size (in samples)	Median filter length (in frames)
RCRNNbest	Hamming	2048	7
RCRNN1	Hamming	2048	15
RCRNN2	Hamming	512	7
Confbest	Hamming	512	40
Conf1	Hamming	512	25
Conf2	Hamming	512	15

Table 4.6: Performance of ensemble models on validation set in terms of PSDS.

Ensemble model	epochs	parameters (M)	PSDS scenario 1	PSDS scenario 2
RCRNNbest + RCRNN1	17	6.8	**0.370**	0.586
RCRNNbest + RCRNN2	17	6.8	0.368	0.589
Confbest + Conf1	10	25.6	0.153	**0.684**
Confbest + Conf2	16	25.6	0.153	0.683

4.3.4 Models using sound source separation

This experiment presents the impact that SSEP had on our models. As shown in Table 4.7, there are four models that we are interested in. The ”RCRNNbest” model and the ”Confbest” model were described in the previous section, and the ”baseline” model in Sections 3.1.1 and 3.2.2. Instead, the ”Confsimple” model is a Conformer-based model that contains a window with 2048 frames size and a median filter with 7 frames length. As presented in Table 4.7, SSEP had a positive impact on some models and a negative on others. The highest positive impact was detected for the ”Confsimple” model in scenario 1 and the highest negative impact for the ”RCRNNbest” model in scenario 2. In scenario 1, without SSEP, the ”RCRNNbest” model achieved the highest PSDS (0.356), and with SSEP, the ”baseline” model 0.373. Instead, in scenario 2 the ”Confbest” model achieved the highest PSDS with and without SSEP. Figure 4.5 demonstrates the PSD scores in scenario 2 with and without SSEP for the same models.

Table 4.7: Models performance with and without source separation.

Model	PSDS scen 1 without SSEP	PSDS scen 1 with SSEP	diff	PSDS scen 2 without SSEP	PSDS scen 2 with SSEP	diff
RCRNNbest	**0.356**	0.350	-0.6%	0.565	0.524	-4.1%
Confbest	0.150	0.173	+2.3%	**0.681**	**0.672**	-0.9%
Confsimple	0.209	0.244	+5.5%	0.569	0.604	+3.5%
baseline	0.342	**0.373**	+3.1%	0.527	0.549	+2.2%

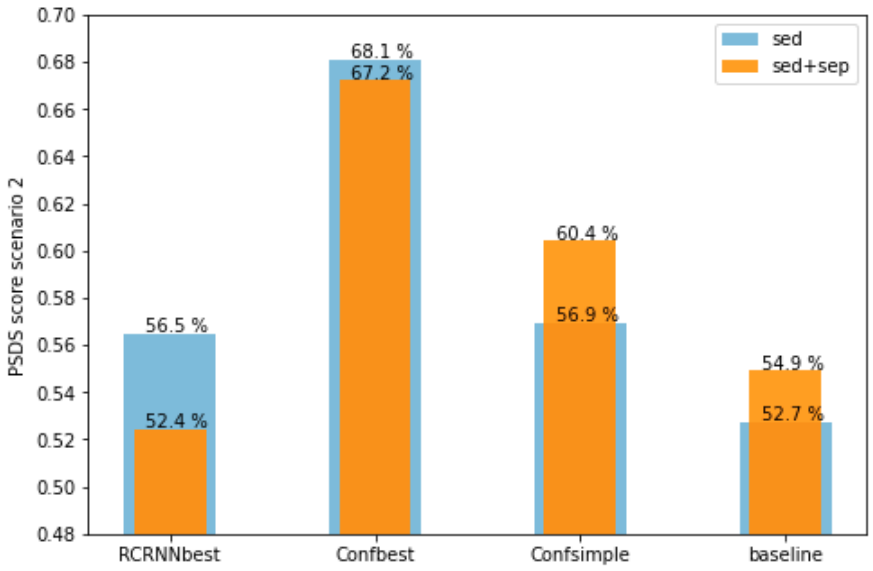

Figure 4.5: PSDS in scenario 2 with and without source separation.

4.3.5 Evaluation process

To evaluate our systems, we used both validation and public evaluation datasets. First are evaluated the best SED models and then the models with SSEP as a preprocessing step.

4.3.5.1 SED evaluation

In Table 4.8 we present the achieved PSDS on the validation and evaluation sets for the SED models that we implemented. It is observed that ensemble models achieve the highest score in most cases. On the validation set in scenario 1 the model that achieves the highest PSDS is the ”RCRNNbest+RCRNN1”, yielding a score equal to 0.370. In scenario 2 on the same dataset, the ”Confbest+Conf1” model outperforms others with a score of 0.684. Instead, in scenario 1 on the evaluation set, the model with the highest score is again the ”RCRNNbest+RCRNN1” with a score equal to 0.398. Finally, in scenario 2 on the evaluation set, the model that outperforms others is the ”Confbest”, providing a score equal to 0.679.

Table 4.8: Performance of SED models on the validation and on the public evaluation dataset in terms of PSDS.

	Validation Set		**Evaluation Set**	
SED models	scenario 1	scenario 2	scenario 1	scenario 2
RCRNNbest	0.356	0.565	0.384	0.620
Confbest	0.150	0.681	0.135	**0.679**
RCRNNbest+RCRNN1	**0.370**	0.586	**0.398**	0.638
Confbest+Conf1	0.153	**0.684**	0.134	0.672
baseline	0.342	0.527	0.359	0.596

4.3.5.2 SED + SSEP evaluation

Table 4.9 demonstrates the PSD scores on the validation and evaluation sets for the SED models that use SSEP as a preprocessing step. The table shows that most models score better on the evaluation set than on validation. On the validation set, the best model in scenario 1 is the ”baseline” and in scenario 2 the ”Confbest.” Instead, in scenario 1 the model that achieves the highest PSDS on the evaluation set is the ”RCRNNbest” with a score of 0.396.

In scenario 2 on the same set, the model that outperforms others is the ”Confbest,” providing a score equal to 0.670.

Table 4.9: Performance of SED+SSEP models on the validation and on the public evaluation dataset in terms of PSDS.

	Validation Set		**Evaluation Set**	
SED+SSEP models	scenario 1	scenario 2	scenario 1	scenario 2
RCRNNbest	0.350	0.524	**0.396**	0.612
Confbest	0.173	**0.672**	0.150	**0.670**
Confsimple	0.244	0.604	0.235	0.626
baseline	**0.373**	0.549	0.393	0.620

Chapter 5

Conclusions

5.1 Summary and conclusions

In this Thesis, we addressed the problem of SED by proposing state-of-the-art solutions for it. These solutions include neural network architectures based on Conformers and RCRNNs. In order to further improve the performance of our SED models, we implemented some preprocessing techniques, data augmentation, postprocessing techniques, and ensemble learning. Preprocessing techniques include downsampling, normalization, and the fundamental one-source separation. Data augmentation contains mixup and addition of Gaussian noise, and finally, postprocessing techniques include median filtering.

We evaluated our systems on the public evaluation dataset used in Task 4 of DCASE Challenge 2021, outperforming the baseline systems in all scenarios.

5.2 Future extensions

Due to the wide area of this research topic, there are many approaches that we did not explore. First of all, the use of different data augmentation techniques such as time-shifting and pitch shifting [56] may provide better results. Also, it is possible to jointly train the SED and SSEP model instead of using SSEP as a preprocessing step. Finally, our models should be trained on more datasets in order to have more accurate results.

Bibliography

[1] DCASE challenge 2021 task 4. `http://dcase.community/challenge2021/task-sound-event-detection-and-separation-in-domestic-environments`. Date of access: 15-9-2021.

[2] Nicolas Turpault and Romain Serizel. Training sound event detection on a heterogeneous dataset. *CoRR*, abs/2007.03931, 2020.

[3] Nam Kyun Kim and Hong Kook Kim. Polyphonic sound event detection based on residual convolutional recurrent neural network with semi-supervised loss function. *IEEE Access*, 9:7564–7575, 2021.

[4] Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, et al. Conformer: Convolution-augmented transformer for speech recognition. *CoRR, abs/2005.08100*, 2020.

[5] Annamaria Mesaros, Toni Heittola, Tuomas Virtanen, and Mark D. Plumbley. Sound event detection: A tutorial. *IEEE Signal Processing Magazine*, 38(5):67–83, 2021.

[6] T. K. Chan and Cheng Siong Chin. A comprehensive review of polyphonic sound event detection. *IEEE Access*, 8:103339–103373, 2020.

[7] Justin Salamon, Juan Bello, Claudio Silva, Oded Nov, R. Luke DuBois, Anish Arora, Charlie Mydlarz, and Harish Doraiswamy. SONYC: A system for the monitoring, analysis and mitigation of urban noise pollution. *Communications of the ACM*, 2018.

[8] S. Chandrakala and S. L. Jayalakshmi. Environmental audio scene and sound event recognition for autonomous surveillance: A survey and comparative studies. *ACM Comput. Surv.*, 52(3), June 2019.

[9] Pierre Laffitte, David Sodoyer, Charles Tatkeu, and Laurent Girin. Deep neural networks for automatic detection of screams and shouted speech in subway trains. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6460–6464, 2016.

[10] Chloé Clavel, Thibaut Ehrette, and Gaël Richard. Events detection for an audio-based surveillance system. In *2005 IEEE International Conference on Multimedia and Expo*, pages 1306–1309, 2005.

[11] Panagiotis Giannoulis, Gerasimos Potamianos, Athanasios Katsamanis, and Petros Maragos. Multi-microphone fusion for detection of speech and acoustic events in smart spaces. In *2014 22nd European Signal Processing Conference (EUSIPCO)*, pages 2375–2379, 2014.

[12] Sacha Krstulović. Audio event recognition in the smart home. In Tuomas Virtanen, Mark D. Plumbley, and Dan Ellis, editors, *Computational Analysis of Sound Scenes and Events*, pages 335–371. Springer International Publishing, Cham, 2018.

[13] Mahesh Kumar Nandwana and T. Hasan. Towards smart-cars that can listen: Abnormal acoustic event detection on the road. In *INTERSPEECH*, 2016.

[14] Ya-Ti Peng, Ching-Yung Lin, Ming-Ting Sun, and Kun-Cheng Tsai. Healthcare audio event classification using hidden Markov models and hierarchical hidden Markov models. In *2009 IEEE International Conference on Multimedia and Expo*, pages 1218–1221, 2009.

[15] Qin Jin, Peter F. Schulam, Shourabh Rawat, Susanne Burger, Duo Ding, and Florian Metze. Event-based video retrieval using audio. In *Annual Conf. of the International Speech Communication Association*, 2012.

[16] DCASE: Detection and classification of acoustic sound events. `http://dcase.community/`. Date of access: 15-9-2021.

[17] Koichi Miyazaki, Tatsuya Komatsu, Tomoki Hayashi, Shinji Watanabe, Tomoki Toda, and Kazuya Takeda. Convolution-augmented transformer for semi-supervised sound event detection. Technical report, DCASE2020 Challenge, June 2020.

[18] Scott Wisdom, Efthymios Tzinis, Hakan Erdogan, Ron J. Weiss, Kevin Wilson, and John R. Hershey. Unsupervised sound separation using mixtures of mixtures. *CoRR, abs/2006.12701*, 2020.

[19] Ilya Kavalerov, Scott Wisdom, Hakan Erdogan, Brian Patton, Kevin Wilson, Jonathan Le Roux, and John R. Hershey. Universal sound separation. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pages 175–179, 2019.

[20] Nicolas Turpault, Scott Wisdom, Hakan Erdogan, John Hershey, Romain Serizel, Eduardo Fonseca, Prem Seetharaman, and Justin Salamon. Improving sound event detection in domestic environments using sound separation. *CoRR, abs/2007.03932*, 2020.

[21] Xi Zhou, Xiaodan Zhuang, Ming Liu, Hao Tang, Mark Hasegawa-Johnson, and Thomas Huang. HMM-based acoustic event detection with adaboost feature selection. In Rainer Stiefelhagen, Rachel Bowers, and Jonathan Fiscus, editors, *Multimodal Technologies for Perception of Humans*, pages 345–353, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[22] Emmanouil Benetos, Grégoire Lafay, Mathieu Lagrange, and Mark D. Plumbley. Detection of overlapping acoustic events using a temporally-constrained probabilistic model. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6450–6454, 2016.

[23] Jonathan Dennis, Tran Dat, and Eng Chng. Overlapping sound event recognition using local spectrogram features with the generalised Hough transform. *Pattern Recognition Letters*, 34:1085–1093, 2013.

[24] Mirco Ravanelli, Benjamin Elizalde, Karl Ni, and Gerald Friedland. Audio concept classification with hierarchical deep neural networks. In *European Signal Processing Conference*, 2014.

[25] Toni Heittola, Annamaria Mesaros, Tuomas Virtanen, and Moncef Gabbouj. Supervised model training for overlapping sound events based on unsupervised source separation. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8677–8681, 2013.

[26] Justin Salamon and Juan Pablo Bello. Deep convolutional neural networks and data augmentation for environmental sound classification. *IEEE Signal Processing Letters*, 24(3):279–283, 2017.

[27] Karol J. Piczak. Environmental sound classification with convolutional neural networks. In *2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6, 2015.

[28] Miquel Espi, Masakiyo Fujimoto, Keisuke Kinoshita, and Tomohiro Nakatani. Exploiting spectro-temporal locality in deep learning based acoustic event detection. *EURASIP J. Audio Speech Music. Process.*, 2015:26, 2015.

[29] Giambattista Parascandolo, Heikki Huttunen, and Tuomas Virtanen. Recurrent neural networks for polyphonic sound event detection in real life recordings. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016.

[30] Yun Wang, Leonardo Neves, and Florian Metze. Audio-based multimedia event detection using deep recurrent neural networks. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2742–2746, 2016.

[31] Emre Cakir, Giambattista Parascandolo, Toni Heittola, Heikki Huttunen, and Tuomas Virtanen. Convolutional recurrent neural networks for polyphonic sound event detection. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(6):1291–1303, 2017.

[32] Panagiotis Giannoulis, Gerasimos Potamianos, and Petros Maragos. Overlapped sound event classification via multi-channel sound separation network. In *European Signal Processing Conference (EUSIPCO)*, 2021.

[33] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[34] Nicolas Turpault, Romain Serizel, Ankit Parag Shah, and Justin Salamon. Sound event detection in domestic environments with weakly labeled data and soundscape synthesis. In *Workshop on Detection and Classification of Acoustic Scenes and Events*, 2019.

[35] Gert Dekkers, Steven Lauwereins, Bart Thoen, Mulu Weldegebreal Adhana, Henk Brouckxon, Bertold Van den Bergh, Toon van Waterschoot, Bart Vanrumste, Marian Verhelst, and Peter Karsmakers. The SINS database for detection of daily activities in a home environment using an acoustic sensor network. pages 1–5. DCASE Workshop, 2017.

[36] Annamaria Mesaros, Toni Heittola, and Tuomas Virtanen. TUT database for acoustic scene classification and sound event detection. In *2016 24th European Signal Processing Conference (EUSIPCO)*, pages 1128–1132, 2016.

[37] Eduardo Fonseca, Xavier Favory, Jordi Pons, Frederic Font, and Xavier Serra. FSD50K: an open dataset of human-labeled sound events. *CoRR*, abs/2010.00475, 2020.

[38] Bart Thomee, David A Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li. YFCC100M: The new data in multimedia research. *Communications of the ACM*, 59(2):64–73, 2016.

[39] Scott Wisdom, Hakan Erdogan, Daniel P. W. Ellis, Romain Serizel, Nicolas Turpault, Eduardo Fonseca, Justin Salamon, Prem Seetharaman, and John R. Hershey. What's all the fuss about free universal sound separation data? In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 186–190, 2021.

[40] Frederic Font, Gerard Roma, and Xavier Serra. Freesound technical demo. In *Proceedings of the 21st ACM International Conference on Multimedia*, MM '13, page 411–412, 2013.

[41] Jort F. Gemmeke, Daniel P. W. Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R. Channing Moore, Manoj Plakal, and Marvin Ritter. Audio Set: An ontology and human-labeled dataset for audio events. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 776–780, 2017.

[42] Justin Salamon, Duncan MacConnell, Mark Cartwright, Peter Li, and Juan Pablo Bello. Scaper: A library for soundscape synthesis and augmentation. In *2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pages 344–348, 2017.

[43] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 1195–1204, 2017.

[44] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2014.

[45] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018.

[46] Lu JiaKai. Mean teacher convolution system for dcase 2018 task 4. Technical report, DCASE, 2018.

[47] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. CBAM: Convolutional block attention module. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[48] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. Transformer-XL: Attentive language models beyond a fixed-length context. *CoRR*, abs/1901.02860, 2019.

[49] Qiang Wang, Bei Li, Tong Xiao, Jingbo Zhu, Changliang Li, Derek F. Wong, and Lidia S. Chao. Learning deep transformer models for machine translation. *CoRR*, abs/1906.01787, 2019.

[50] Toan Q. Nguyen and Julian Salazar. Transformers without tears: Improving the normalization of self-attention. *CoRR*, abs/1910.05895, 2019.

[51] Zhanghao Wu, Zhijian Liu, Ji Lin, Yujun Lin, and Song Han. Lite transformer with long-short range attention. *CoRR*, abs/2004.11886, 2020.

[52] Ziqiang Shi, Liu Liu, Huibin Lin, Rujie Liu, and Anyan Shi. HODGEPODGE: sound event detection based on ensemble of semi-supervised learning methods. *CoRR*, abs/1907.07398, 2019.

[53] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, and Luca Antiga et al. PyTorch: An imperative style, high-performance deep learning library. *CoRR*, abs/1912.01703, 2019.

[54] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. TensorFlow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation*, volume 16, pages 265–283, 2016.

[55] Cagdas Bilen, Giacomo Ferroni, Francesco Tuveri, Juan Azcarreta, and Sacha Krstulovic. A framework for the robust evaluation of sound event detection. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 61–65, 2020.

[56] Szymon Buś and Konrad Jędrzejewski. Digital signal processing techniques for pitch shifting and time scaling of audio signals. In Ryszard S. Romaniuk, editor, *Photonics Applications in Astronomy, Communications, Industry, and High-Energy Physics Experiments 2016*, volume 10031, pages 1484 – 1493. International Society for Optics and Photonics, SPIE, 2016.