Advanced Informatics For Event Detection And Temporal Localization

Chan Teck Kai

Supervisor: Prof. Chin Cheng Siong

Faculty of Science, Agriculture and Engineering

Newcastle University Upon Tyne

This dissertation is submitted for the degree of

Doctor of Philosophy

August 2021

Abstract

The primary objective of a Sound Event Detection (SED) system is to detect the prescene of an acoustic event (i.e., audio tagging) and to return the onset and offset of the identified acoustic event within an audio clip (i.e., temporal localization). Such a system can be promising in wildlife and biodiversity monitoring, surveillance, and smart-home applications.

However, developing a system to be adept at both subtasks is not a trivial task. It can be hindered by the need for a large amount of strongly labeled data, where the event tags and the corresponding onsets and offsets are known with certainty. This is a limiting factor as strongly labeled data is challenging to collect and is prone to annotation errors due to the ambiguity in the perception of onsets and offsets.

In this thesis, we propose to address the lack of strongly labeled data by using pseudo strongly labeled data, where the event tags are known with certainty while the corresponding onsets and offsets are estimated. While Nonnegative Matrix Factorization can be used directly for SED but with limited accuracy, we show that it can be a useful tool for pseudo labeling. We further show that pseudo strongly labeled data estimated using our proposed methods can improve the accuracy of a SED system developed using deep learning approaches.

Subsequent work then focused on improving a SED system as a whole rather than a single subtask. This leads to the proposal of a novel student-teacher training framework that incorporates a noise-robust loss function, a new cyclic training scheme, an improved depthwise separable convolution, a triple instance-level temporal pooling approach, and an improved Transformer encoding layer. Together with synthetic strongly labeled data and a large corpus of unlabeled data, we show that a SED system developed using our proposed method is capable of producing state-of-the-art performance.

Acknowledgements

The completion of this study could not have been possible without the support and help of many individuals.

First and foremost, I would like to express my deepest gratitude to my supervisor, Prof Chin Cheng Siong, who has been my supervisor since my undergraduate days. I would not have made it this far without his supervision and guidance.

I am also thankful to my fellow schoolmate, Kek Xing Yong, for his technical help whenever I ran into problems with Amazon Web Service or other database-related problems. The words of encouragement that we exchanged have helped me endure these few years.

I am also very grateful to my parents, who have always offered their listening ears and providing emotional support.

Special thanks to my loving wife, Emily Chua, who has never doubted me during this journey. She has been my pillar of strength, whose words of encouragement and support during difficult times are much appreciated and duly noted.

Table of Contents

List of Figures

List of Tables

Nomenclature

List of Symbols

$\mathbf{b}$	Bias
C	Cost
d	Index of dimension
d_{k}	Dimension of key
d_{model}	Dimension of model
EF	Expansion factor
ε	Weighing parameter
$f(.)$	Activation function
$g_{i,j}$	Ground truth for event i at frame j
$\ddot{\mathbf{H}}$	Activation matrix
$\mathbf{H}$	Head
$\overset{k\rightarrow}{\ddot{\mathbf{H}}}$	$\ddot{\mathbf{H}}$ which is shifted k steps to the right
$\overset{k\rightarrow}{\ddot{\mathbf{H}}}_t$	$\ddot{\mathbf{H}}$ which is shifted k steps to the right at time step t
$\ddot{\mathbf{H}}_t$	Activation matrix at time step t
$\ddot{\mathbf{H}}_{t-1}$	Activation matrix at time step $t-1$
$J(.)$	Cost function
$\mathbf{K}$	Key
κ	Switching factor
λ	Confidence threshold

λ_{curr} Current confidence threshold

λ_{max} Maximum confidence threshold

λ_{min} Minimum confidence threshold

l_{c} Clip-level loss

l_{ce} Cross Entropy Loss

l_{con} Consistency cost on labeled sample

l_{unlabel} Consistency cost on unlabeled sample

l_{f} Frame-level loss

l_{i} Interpolated consistency loss

l_{i}^{L} Interpolated consistency loss on labeled data

l_{ri}^{L} Regularized interpolated consistency loss on labeled data

l_{ce} Reverse Cross Entropy Loss

LR_{curr} Current learning rate

l_{ri} Regularized interpolated consistency loss

LR_{max} Maximum learning rate

LR_{min} Minimum learning rate

l_{sbce} Symmetrical Binary Cross Entropy Loss

l_{ce} Symmetrical Cross Entropy Loss

$M_{i,j}$ Predicted probabilty for event i at frame j

μ Weighing parameter

N_{b} Number of frequency bins

N_{BR} Number of branches

N_{c} Number of channels

N_{de} Number of decoding layers

N_{e}	Number of events
N_{en}	Number of encoding layers
N_{f}	Number of frames
N_{FBR}	Number of positionwise feedforward branches
N_{h}	Number of heads
ν	Weighing parameter
P	Positive value representing training progression
P_{curr}	Current iteration
PE	Positional encoding
P_{mult}	Positive integer to delay the next learning rate reset
pos	Index of position
ψ	Mixing factor
P_{tot}	Total iteration before a learning rate reset
Q	Query
r	Number of component
RF	Reduction factor
S_i^{c}	Student model's predicted probability of event i in a labeled sample
$\tilde{S}_i^{\mathrm{c}}$	Student model's predicted probability of event i in an unlabeled sample
σ	Sigmoid function
$S_{i,j}$	Student model's predicted probability of event i at frame j
S^{mc}	Student model's predicted probability of event i in a mixed labeled sample
$\tilde{S}^{\mathrm{mc}}$	Student model's predicted probability of event i in a mixed unlabeled sample
$\tilde{\mathbf{S}}^{\mathrm{mc}}$	Vector representing student model's predicted probabilities in a mixed unlabeled sample

$\mathbf{S}^{\mathrm{mc}}$ Vector representing student model's predicted probabilities in a mixed labeled sample

$\ddot{\mathbf{S}}_k$ Vector representing clip-level prediction from student model on sample k

$\ddot{\mathbf{S}}_k^{\mathrm{u}}$ Vector representing clip-level prediction from student model on unlabeled sample k

$\mathbf{T}$ Vector representing teacher model's predicted probabilities of all events in a labeled sample

τ Weighing parameter

T_i Teacher model's predicted probability of event i in a labeled sample

$\mathbf{T}^{\mathrm{m}}$ Vector representing teacher model's interpolated predicted probabilties in a labeled mixed sample

$\acute{\mathbf{T}}^{\mathrm{m}}$ Vector representing teacher model's combined predicted probabilties in an unlabeled mixed sample

$\acute{T}_i^{\mathrm{m}}$ Teacher model's combined predicted probability of event i in an unlabeled sample

$\hat{\mathbf{T}}^{\mathrm{m}}$ Vector representing teacher model's predicted probabilities in a mixed unlabeled sample

$\hat{T}_i$ Teacher model's predicted probability of event i in a mixed unlabeled sample

T_i^{m} Teacher model's interpolated predicted probability of event i in a labeled mixed sample

$\ddot{\mathbf{T}}_k$ Vector representing clip-level prediction from teacher model on sample k

$\tilde{\mathbf{T}}^{\mathrm{m}}$ Vector representing teacher model's interpolated predicted probabilties in an unlabeled mixed sample

$\tilde{T}_i^{\mathrm{m}}$ Teacher model's interpolated predicted probability of event i in an unlabeled mixed sample

$\tilde{\mathbf{T}}$ Vector representing teacher model's predicted probabilities of all events in an unlabeled sample

$\tilde{T}_i$ Teacher model's predicted probability of event i in an unlabeled sample

$\ddot{\mathbf{T}}_k^{\mathrm{u}}$ Vector representing clip-level prediction from teacher model on unlabeled sample k

$\mathbf{U}_1^{\mathrm{A}}$ Augmented feature representation of unlabeled sample 1

$\bar{\mathbf{U}}_1^{\mathrm{A}}$ Portion of $\mathbf{U}_1^{\mathrm{A}}$

$\mathbf{U}_1^{\mathrm{U}}$ Unaugmented feature representation of unlabeled sample 1

$\mathbf{U}_2^{\mathrm{A}}$ Augmented feature representation of unlabeled sample 2

$\bar{\mathbf{U}}_2^{\mathrm{A}}$ Portion of $\mathbf{U}_2^{\mathrm{A}}$

$\mathbf{U}_2^{\mathrm{U}}$ Unaugmented feature representation of unlabeled sample 2

$\mathbf{U}_{\mathrm{m}}$ Feature representation of a mixed unlabeled sample

$\mathbf{V}$ Value

ς Trainable parameter in meta-ACON

$\ddot{\mathbf{V}}$ Nonnegative matrix

$\tilde{\mathbf{V}}$ Estimated nonnegative matrix of $\ddot{\mathbf{V}}$

$\mathbf{W}$ Weights

$\mathbf{W}_i^{\mathrm{K}}$ Parameter matrix for the i head

$\ddot{\mathbf{W}}$ Basis/Dictionary matrix

$\ddot{\mathbf{W}}_k$ $\ddot{\mathbf{W}}$ at k-step shift

$\ddot{\mathbf{W}}_{k,t}$ $\ddot{\mathbf{W}}$ at k-step shift and time step t

$\ddot{\mathbf{W}}_{k,t-1}$ $\ddot{\mathbf{W}}$ at k-step shift and time step $t-1$

$\ddot{\mathbf{W}}_t$ Basis/Dictionary matrix at time step t

$\ddot{\mathbf{W}}_{t-1}$ Basis/Dictionary matrix at time step $t-1$

$\mathbf{W}^{\mathrm{O}}$ Parameter matrix for the concatenated feature vector (i.e., head)

$\mathbf{W}_i^{\mathrm{Q}}$ Parameter matrix for the i head

w Weighing parameter

$\mathbf{W}_i^{\mathrm{V}}$ Parameter matrix for the i head

$\mathbf{x}$ Input vector/input matrix/input features

$\bar{\mathbf{x}}$ Output matrix from the MHA

$\mathbf{x}_{\text{FM}}$ Feature maps from the last convolutional layer

$\hat{\mathbf{x}}$ Output matrix from the positional encoding

$\tilde{\mathbf{x}}$ Output matrix from the first Macaron Net positionwise FF module

$\mathbf{Y}$ Output vector/output matrix/output features

$\mathbf{Y}_{\text{en}}$ Output matrix from an encoding layer

$\hat{z}_i$ Predicted probability of event i in an audio clip

z_i Ground truth of event i in a labeled sample

$\mathbf{Z}_k$ Vector representing clip-level ground truth of sample k

List of Abbreviations

AC-GAN Auxiliary Classifier Generative Adversarial Network

ACR Autocorrelation

AED Acoustic Event Detection

ASA Auditory Scene Analysis

ASR Automatic Speech Recognition

AUC Area Under Curve

BCE Binary Cross Entropy

BGRU Bidirectional GRU

BLSTM Bidirectional Long Short Term Memory

BN Batch Normalization

CapsNet Capsule Neural Network

CASA Computational Auditory Scene Analysis

CBLSTM Convolutional Bidirectional Long Short Term Memory

CD Contrastive Divergence

CE Cross Entropy

CG Context Gating

CNMF Convolutive Nonegative Matrix Factorization

CNN Convolutional Neural Network

CNNT Convolutional Neural Network-Transformer

cRBM Conditional Restricted Boltzmann Machine

CRNN Convolutional Recurrent Neural Network

CRP Chinese Restaurant Process

DCASE Detection and Classification of Acoustic Scenes and Events

DCT Discrete Cosine Transform

DNN Deep Neural Network

dom-freq Dominant frequency and its amplitude

EER Equal Error Rate

EM Expectation-Maximization

ER Error Rate

FC Fully Connected

FF Feedforward

FFT Fast Fourier Transform

FN False negative

FNN Feedforward Neural Network

FP False positive

GAN Generative Adversarial Network

GCC-PHAT Generalized cross-correlation with phase-based weighting

GLU Gated Linear Unit

GMM Gaussian Mixture Model

GMM-HMM Gaussian Mixture Model-Hidden Markov Model

GPU Graphics Processing Unit

GRU Gated Recurrent Unit

HMM Hidden Markov Model

iConformer Improved-Conformer

LN Layer Normalization

MAE Mean Absolute Error

meta-ACON meta-ACtivate-Or-Not

MFCCs Mel-Frequency Cepstral Coefficients

MHA Multi-Head Attention module

mIBP Markov Indian Buffet Process

mixup_c Mixup by concatenation

MLP Multi-Layer Perceptron

NMF Nonnegative Matrix Factorization

NN Neural Network

PCA Principle Component Analysis

RBCE Reverse Binary Cross Entropy

RBCE Reverse Binary Cross Entropy

RBM Restricted Boltzmann Machine

RCE Reverse Cross Entropy

ReLU Rectified Linear Unit

RF Random Forest

RMSE Root Mean Square Error

RNN Recurrent Neural Network

SBCE Symmetrical Binary Cross Entropy

SCE Symmetrical Cross Entropy

SED Sound Event Detection

SE Squeeze and Excite

SM Student Model

SNR Signal-to-Noise Ratio

SOTA State-Of-The-Art

STFT Short Time Fast Fourier Transform

TDoA Time Difference of Arrival

TF Time-Frequency

TM Teacher Model

TP True positive

Chapter 1. Introduction

1.1. Background

Sound phenomena can be described as a series of events that begin with the mechanical disturbance in a medium. For example, a vibrating sound source sets the molecules in a medium into motion, leading to deviations from the static pressure. These deviations then propagate in the form of longitudinal waves known as sound waves (Thewissen and Nummela, 2008). These waves are received by the human ears, which cause us to react differently depending on the situation and individual preference.

The environment surrounding us usually comprises several different sounds interleaving and overlapping in time and frequency, resulting in a complex array of acoustic information (Virtanen et al., 2017). Nevertheless, such complex information can be easily deciphered by our auditory system allowing us to make sense of the acoustic information. This is made possible through a fundamental skill known as Auditory Scene Analysis (ASA). The analytical framework can be described in two stages: decomposing an acoustic signal into different sensory components, followed by combining components from similar sources into a perceptual structure that can be interpreted by higher-level processes (Bregman, 1990). Subsequent research efforts to replicate ASA using computational means were then termed Computational Auditory Scene Analysis (CASA). One major research area in this domain is Sound Event Detection (SED), which may also be referred to as Acoustic Event Detection (AED).

A SED system can be described as an intelligent system that mimics an aspect of the auditory system where the primary objective is to perform audio tagging and temporal localization simultaneously. Audio tagging refers to the detection of an acoustic event of interest within an audio clip. In contrast, temporal localization refers to the annotation of the onset and offset of the identified acoustic event of interest.

A SED system can be categorized into a monophonic SED system or a polyphonic SED system. As illustrated in Figure 1.1, a monophonic SED system can only detect a single event within a specific time frame. In comparison, a polyphonic SED system has the capability to detect more than one event within a specific time frame. Naturally, a

polyphonic SED system is more suitable for real-life applications since a recorded audio clip is more likely to contain multiple acoustic events which may overlap.

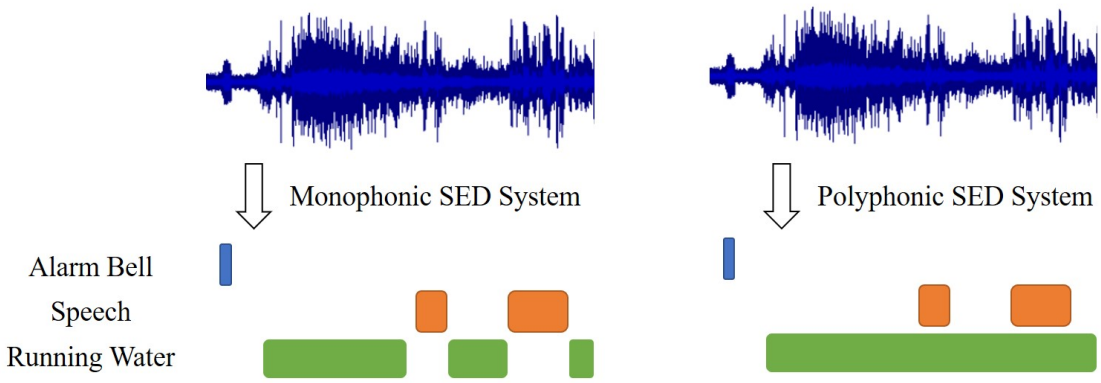


Figure 1.1 Difference between a monophonic (left) and polyphonic (right) SED system

Compared to an image, the information provided by a SED system may seem trivial at first thought; however, a SED system can have several advantages compared to a camera or video recorder. Firstly, a sound is not affected by the degree of illumination or occlusion, making it suitable for deployment in dark or areas with low visibility. Secondly, some events can only be detected by sounds such as a car horn or gunshot. Furthermore, sound can attract the immediate attention of an individual, while a video recording may not. For example, a baby wailing can capture the immediate attention of a parent or guardian, who can then remedy any issue or discomfort the baby faced. However, the video captured by a closed-circuit television may not show any symptoms or signs of the baby's discomfort, which can delay any necessary treatment. Finally, storing and processing an audio clip typically consumes lesser computation resources than a video recording.

Thus, a SED system can be promising in various domains such as medical telemonitoring (Nguyen and Tran, 2013), surveillance (Chaudhary et al., 2018; Clavel et al., 2005), equipment monitoring (Chan and Chin, 2019; Grollmisch et al., 2019), wildlife and biodiversity monitoring (Florentin et al., 2016; Zhao et al., 2017) .

1.2. Challenges for SED

As mentioned earlier, the environment surrounding us usually comprises several different sounds interleaving and overlapping in time and frequency (Virtanen et al., 2017). Thus, audio recorded in our daily lives is a complex array of acoustic information that can be difficult to decode through computational means.

As events may coincide, this may indicate that features extracted from the audio mixture may not match any features extracted from sounds in isolation (Parascandolo

et al., 2016; Schroder et al., 2017) . Moreover, it is not known a priori which events are overlapped and the number of events captured in an audio clip.

Subsequently, each event class may be made up of different sources with varying characteristics. For example, a barking sound produced by a chihuahua and a husky is different and can be affected by the dog's level of aggression at that moment (Parascandolo et al., 2016). Furthermore, background noise can complicate the identification of sound events within a particular time frame (Kong et al., 2019b) and is further aggravated if the Signal-to-Noise Ratio (SNR) is low.

System development can also be hindered by the need for a large amount of strongly labeled data, where the event tags and their corresponding onsets and offsets are known with certainty (as seen in Figure 1.2A). This is because collecting such data is difficult and time-consuming as it requires repeated listening and adjustments of events' boundaries on a visual interface (Kim and Pardo, 2019). Accuracy of onset and offset annotation is also ambiguous due to the fade in and fade out effect (McFee et al., 2018) and is subjected to the person labeling the event. Thus, the sizes of such data are often limited to minutes or a few hours (Kim and Pardo, 2019; Kong et al., 2019b; McFee et al., 2018).

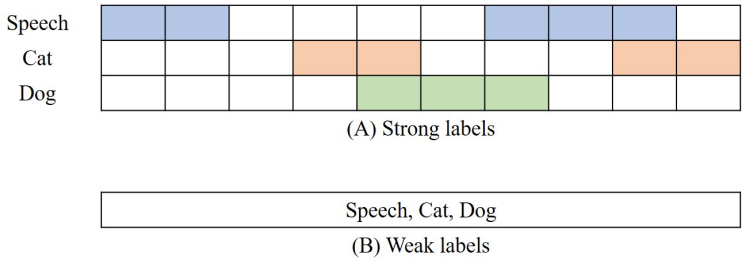


Figure 1.2 Difference between strong label and weak label

Research efforts then turn to weakly labeled data, where only the event tags are known with certainty for system development (as seen in Figure 1.2B). The use of such data alleviates the need for strongly labeled data and has shown promising results in various studies (Lin et al., 2019; Lu, 2018; Miyazaki et al., 2020). However, as seen in the Detection and Classification of Acoustic Scenes and Events (DCASE) 2017 challenge task 4, no system could win both tagging and localization subtasks when given only weakly labeled data. This may be due to the lack of strongly labeled data, and the development of a SED system may still require strongly labeled data to achieve maximal performance. Such a conclusion is derived from a study conducted by Turpault et al. (2020a), where they found that weakly labeled data can degrade the tagging performance slightly when using an end-to-end classifier trained in a discriminative manner. Subsequently, Hershey

et al. (2021) also found that the use of both strongly labeled and weakly labeled data can substantially improve the classifier's performance in the audio classification domain.

Among the different strategies used to train a SED system that excludes strongly labeled data, the most popular framework is the Mean-Teacher approach (Tarvainen and Valpola, 2017). The main idea is to train two identical models synchronously, in which one is known as the student model while the other is known as the teacher model. The student model is updated with a multi-label classification cost and a consistency cost, where the consistency cost enforces the predictions of the student model to be consistent with the predictions of the teacher model on the unlabeled data. On the other hand, the teacher model is updated using the exponential moving average of the student model's weight. An example of such an implementation is illustrated in Figure 1.3.

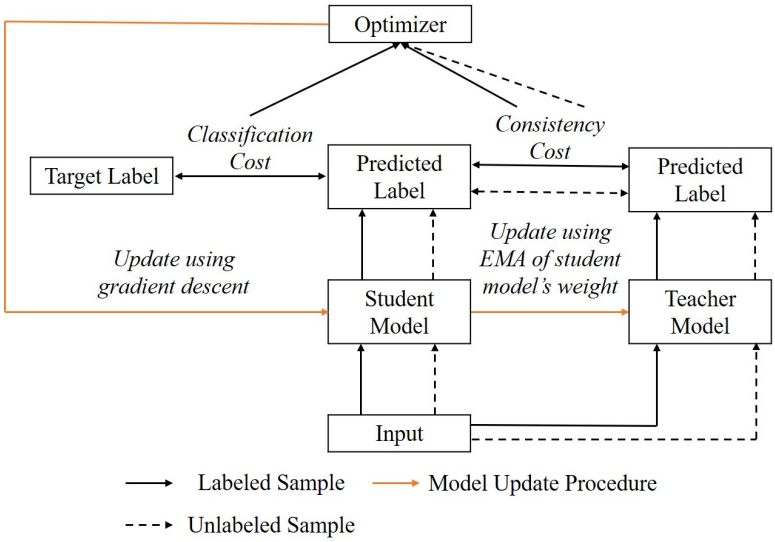


Figure 1.3 Example of a Mean-Teacher approach

However, the Mean-Teacher approach (Tarvainen and Valpola, 2017) can have two critical limitations. Firstly, it can be computationally expensive if a very deep model is designed due to a need to train two of them synchronously. Secondly, a model designed might only be optimal for either audio tagging or temporal localization but not both.

1.3. Contributions of This Thesis

This thesis proposes to address the lack of strongly labeled data by using pseudo strongly labeled data. A set of pseudo strongly labeled data can be defined as a set of data where the event tags are known with certainty, but the corresponding onsets and offsets are estimated.

In addition, we also propose to address the issues caused by using the Mean-Teacher approach. Using the pseudo strongly labeled data, we conducted different experiments and

studies to investigate various aspects of a SED system to maximize the system's accuracy at both subtasks (i.e., audio tagging and temporal localization). These lead to the following contributions:

- **A novel pseudo labeling approach using unsupervised Nonnegative Matrix Factorization (NMF).** In the SED domain, NMF (Lee and Seung, 1999) is commonly used as a decomposition method to extract spectral templates from isolated events to form a dictionary. Temporal localization is then done by applying a threshold on the activation matrix obtained from the decomposition of the test data using the event dictionary (i.e., consolidated spectral templates). In our study, instead of applying NMF in the conventional manner, we propose to apply NMF on each weakly labeled sound clip to derive the activation matrix. We then locate the activated frames in the activation matrix, which acts as the temporal labels for the weakly labeled clip. If the clip contains multiple events, those activated frames are deemed to contain all the sound events. A Convolutional Neural Network (CNN) is then trained using the pseudo strongly labeled data. Experiment results show that our proposed method can be a promising solution to train a SED system.

  - The entire framework is described in Chapter 3 and was published as a workshop paper; Chan, T. K., Chin, C. S., and Y. Li. (2019). Non-negative matrix factorization-convolution neural network (NMF-CNN) for sound event detection. In *Proceeding of the Workshop on Detection and Classification of Acoustic Scenes and Events*, pages 40-44, New York, NY, USA.
  - This study also won the DCASE 2019 challenge task 4 Judges' Award for the method considered by the judges to be the most interesting or innovative.

- **An improved pseudo labeling approach using supervised NMF and a combinative transfer learning and semi-supervised learning framework to train a SED system.** Our previous study assumed that each activated frame contains all the events present in an audio clip; such an assumption can introduce noise into the training data. As such, we propose an improved pseudo labeling method using supervised NMF. The idea is to approximate the temporal labels for each weakly labeled data using spectral templates extracted from synthetic strongly labeled data. We then proposed a combinative transfer learning and semi-supervised learning framework to train a SED system using the pseudo strongly labeled data. This combinative approach incorporates a cyclic learning scheme that improves convergence and a

novel student-teacher framework that allows the system to be adept at audio tagging and temporal localization. Subsequent analysis shows that our system is capable of producing SOTA (State-Of-The-Art) performance.

  - The entire framework is described in Chapter 3 and was summarized as journal paper; Chan, T. K., Chin, C. S., and Y. Li. (2021). Semi-supervised nmf-cnn for sound event detection. *IEEE Access*, 9:130529-130542.

- **Further improvement of the pseudo labeling quality using Convolutive Non-negative Matrix Factorization (CNMF) and a novel architecture combining CNN with Macaron Net encoding layer trained using curriculum consistency costs.** We propose using CNMF (Smaragdis, 2007) as our pseudo labeling method based on the hypothesis that if CNMF can better separate the audio mixtures than NMF, it should also improve the pseudo labeling quality. We then propose a novel architecture combining CNN with the Macaron Net encoding layer (Lu et al., 2019) that uses a new activation function known as Mish (Misra, 2019) rather than the conventional Rectified Linear Unit (ReLU) activation function in the entire architecture. Finally, to better leverage the large corpus of unlabeled data for semi-supervised learning, we propose two new consistency costs: curriculum consistency cost and curriculum interpolated consistency cost. The main idea is to vary the confidence threshold so that consistency costs that ensure consistency between the student and teacher models will not be calculated based on only highly confident predictions throughout the entire training process. Experiment results then show that CNMF is a better pseudo labeling tool, and a system trained using our proposed framework can be competitive to the SOTA.

  - The entire framework is described in Chapter 4 and was published as workshop paper; Chan, T. K., and Chin, C. S. (2021). Detecting sound events using convolutional macaron net with pseudo strong labels. In *Proceeding of the IEEE 23rd Workshop on Multimedia Signal Processing*, Tampere, Finland.

- **The proposal of an improved student-teacher framework that incorporates a triple instance-level pooling approach, an improved Macaron Net encoding layer, and an improved cyclic learning scheme that improve both audio tagging and temporalization.** Although our earlier student-teacher framework can be competitive to the SOTA, having to tune two completely different models can be time-consuming. As such, we propose a straightforward design where the only

differences lie in the number of convolutional layers, pooling size, and temporal pooling method. Subsequently, although the use of a Macaron Net encoding layer (Lu et al., 2019) can improve the accuracy of temporal localization, we found that the accuracy of audio tagging can be reduced. We then focused on improving the SED system as a whole rather than a single subtask. This leads to the proposal of several new ideas. Firstly, an improved cyclic learning scheme that incorporates a periodic increment and decrement of the learning rate. Secondly, a triple instance-level pooling approach that allows the model to learn unique characteristics from each temporal pooling approach. The incorporation of such a module in the teacher model not only raises the audio tagging accuracy but also forces the student model to learn from a more complex model, which leads to an improvement in temporal localization. Finally, we propose an improved Macaron Net encoding layer that adopts the pre-Layer Normalization (LN) arrangement with additional Fully Connected (FC) layers and shows that a single layer single head implementation is sufficient for SED. Extensive experiments were carried out to examine different aspects of our proposals. Based on our framework, our SED system can achieve an event-based F1-score of 48.5%, and by ensembling the top five models, the event-based F1-score can be increased to 50.4%. Such results allow our model to have a minimum margin of over 12% against the baseline system and be competitive against the other SOTA.

  - The entire framework is described in Chapter 4 and was summarized as journal paper; Chan, T. K., and Chin, C. S. (2021). Multi-branch convolutional macaron net for sound event detection. *IEEE/ACM Transactions on Audio, Speech and Language Processing*, 29:2972-2985.

- **The proposal of a noise-robust loss function with an improved depthwise separable convolution and an improved Conformer encoding layer.** As pseudo labels will inevitably contain a certain amount of noise, we propose a noise-robust loss function that extends the Binary Cross Entropy (BCE) by considering the Reverse Binary Cross Entropy (RBCE) to promote learning in the presence of label noise. Driven by the fact that deep learning models are often highly parametrized and may face deployment issues in many real-world applications, we propose using a lightweight convolution. This lightweight convolution is commonly known as the depthwise separable convolution (Chollet, 2017b), which factorizes a standard convolution into a depthwise and pointwise convolution. Unlike the standard depthwise separable convolution, which does not include the use of nonlinearities, we propose

the addition of Swish (Ramachandran et al., 2017) and Batch Normalization (BN), which was found to improve the system accuracy. Finally, we propose an improved Conformer encoding layer (Gulati et al., 2020) which utilizes a multibranch position-wise FeedForward (FF) module and a convolution module with lesser convolution operations. Through our experiments, we show that our lightweight system that utilizes approximately 509k parameters has the capability to outperform the SOTA.

  - The entire framework is described in Chapter 5 and was summarized as journal paper and is currently under review.

Other contribution includes **a comprehensive review on SED**, which was published as a journal paper and forms the Chapter 2 of this thesis.

- Chan, T. K., and Chin, C. S. (2020). A comprehensive review on polyphonic sound event detection. *IEEE Access*, 8:103339-103373.

1.4. Thesis Outline

The rest of the thesis is organized as follows:

In Chapter 2, a literature review for SED is provided. The chapter provides an in-depth discussion of different methodologies proposed by various authors, which comprises the features used, detection algorithms, and their corresponding accuracy and limitations.

In the earlier few sections of Chapter 3, the motivation of using pseudo labels is provided, followed by a proof of concept carried out to determine if NMF can be an effective tool for pseudo labeling and how pseudo strongly label data can affect the training of a CNN. These sections include a description of the pseudo labeling procedure and experimental results on the DCASE 2019 challenge task 4 dataset. The later sections of the chapter then present a supervised NMF pseudo labeling method and the proposed combinative transfer learning and semi-supervised learning framework. We first describe how NMF can provide temporal labels in a supervised manner, followed by the description of two different models that are designed to be adept at the respective subtasks (i.e., audio tagging and temporal localization). We then present how we trained the two models and examine the results of extensive experiments to better understand the effects of different hyperparameters.

In Chapter 4, we investigate the effectiveness of CNMF as a pseudo labeling tool and the incorporation of the Macaron Net encoding layer into our SED system, and finally, the use of curriculum consistency costs. The proposed ideas are then evaluated on the DCASE

2020 challenge task 4 dataset. We then attempt to improve a SED system as a whole rather than improving the accuracy on a single subtask. First, we demonstrate that two models can be designed in a simple way and yet adept at both subtasks (i.e., audio tagging and temporal localization). We then evaluate the use of 7 different pooling methods and their combinations and present a theoretical explanation of the results. We then examine other aspects of our system, such as the proposed cyclic learning scheme and improved Macaron Net encoding layer.

In Chapter 5, we attempt to address the noise that can be introduced through pseudo labeling and presents our framework to reduce the number of parameters in our system. Through extensive experiments, we demonstrate that a lightweight system can be competitive or even outperform the SOTA.

Finally, Chapter 6 concludes this thesis and discusses the future research directions.

Chapter 2. Literature Review

A SED system can be broadly classified into non-Neural Network (NN) based and NN-based methodologies. As seen in Figure 2.1, NN-based methods can be further classified into non-hybrid models, hybrid models, and models utilizing weakly labeled data. In this thesis, a non-hybrid model refers to a model that has not been modified or stacked with another model, whereas a hybrid model refers to models that are stacked together to become one model. As models trained using weakly label data generally have a big difference with their detection strategy, they are classified into a different subcategory instead of the non-hybrid or hybrid models subcategory. Thus this chapter proposed to discuss these groups of methodologies in the same manner. The discussion of methods in this section will be done in the following order, the non-NN based methods followed by the NN based methods.

The content of this chapter was published as a journal paper, Chan, T. K., and Chin, C. S. (2020). A Comprehensive Review on Polyphonic Sound Event Detection. *IEEE Access*, 8:103339-103373.

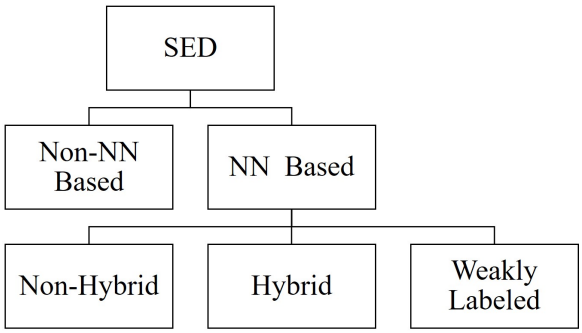


Figure 2.1 Different categories of a SED system

2.1. Non Neural Network Based Methodology

2.1.1. Gaussian Mixture Model-Hidden Markov Model

Perhaps one of the earliest SOTA for Automatic Speech Recognition (ASR) is the Hidden Markov Model (HMM) (Baker et al., 2009; Rabiner, 1989), where it provides the likelihood

of a set of acoustic vectors given a word sequence (Stuttle, 2003). HMM is a statistical model where each state is not directly observable (i.e., hidden) (Mayorga et al., 2015; Shi et al., 2018) and can be considered a natural speech recognition framework. This is because speech has a temporal structure and can be encoded as a sequence of spectral vectors spanning the audio frequency range (Gales and Young, 2007).

With the introduction of the Expectation-Maximization (EM) algorithm, Gaussian Mixture Model (GMM) can be used to model the probability distributions over vectors of input features associated with each state of an HMM (Hinton et al., 2012; Mayorga et al., 2015). Hinton et al. (2012) explained that GMM could have several benefits that make it suitable for such a task. Firstly, with enough components, they can model probability distributions to any required level of accuracy, and secondly, they are relatively easy to fit data using the EM algorithm. Such a combination made them so successful that it was difficult for any new method to outperform them for a long while (Hinton et al., 2012). With success in ASR, GMM-HMM becomes a natural choice for other sound-related tasks (Heck et al., 2016; Mayorga et al., 2015; Rajapakse and Wyse, 2005).

Mesaros et al. (2010) investigated the effectiveness of GMM-HMM on a large-scale audio database that consists of 61 event classes. In their study, Mel-Frequency Cepstral Coefficients (MFCCs), delta MFCCs, delta-delta MFCCs were used as the input features to train a three-state left-to-right HMM with 16 Gaussians per state for each event class. Subsequently, the models were connected into a network HMM having equal transition probabilities from one event model to another. Such implementation would allow an unrestricted sequence of 61 models where any model can follow any other with no limits to the number of detected events. Using the Viterbi algorithm (Forney, 1973), an optimal sequence of events can then be decoded, allowing the detection of the most prominent event at each given polyphonic segment and its timestamp.

However, such implementation only achieved an F1-score of 30.1% but came with an Error Rate (ER) of 84.1%. Despite the inclusion of event frequency of occurrence as the prior knowledge, the accuracy was not improved. Mesaros et al. (2010) explained that this might be due to the adding of events from different environments, which averaged out the differences in count between events specific to a particular environment. Although there was no limit to the number of detected events, such implementation only allowed the detection of the most prominent event in each segment, which did not reflect a real-life scenario where sound events can coincide. Moreover, it was found that the majority of the errors can be caused by overlapping sound events (Diment et al., 2013).

Heittola et al. (2013a) then proposed a two-stage detection methodology, which can be summarized in Figure 2.2, in an attempt to improve detection accuracy and allow polyphonic SED on the same dataset. In their framework, Heittola et al. (2013a) proposed determining the audio context before detecting the events in the audio. Context, in this case, refers to the audio background or environment. Heittola et al. (2013a) explained that this could reduce the search space for sound events as context information can provide rules for selecting a specific set of events.

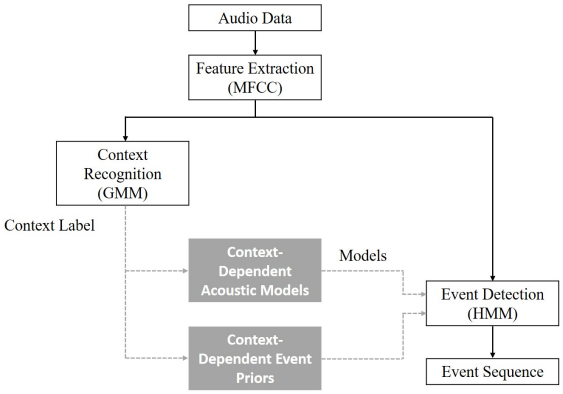


Figure 2.2 Proposed methodology by Heittola et al. (2013a)

Each context recognition system was trained using GMM with MFCCs extracted from the different contexts where each GMM consists of 32 Gaussian distributions. Whereas a SED system was trained using an HMM with Mel Frequency Cepstral Coefficients (MFCCs), delta MFCCs, and delta-delta MFCCs extracted from sound events belonging to different contexts. Similar to (Mesaros et al., 2010), each event class was modeled by an individual three-state left-to-right HMM with 16 Gaussians per state.

Frames with overlapping events were not discarded during training and were instead used as training samples for the respectively sound events. Heittola et al. (2013a) hypothesized that the variability caused by the overlapping sound events classes would be averaged out, and the model will still be able to learn a reliable representation of the target sound events. In order to allow polyphonic SED, consecutive passes of the Viterbi algorithm (Forney, 1973) were proposed, and each decoded pass must be different from the previously decoded one. Such restriction prevented the detection of a similar event, and in their study, the number of consecutive passes was fixed at 4 (Heittola et al., 2013a).

Test audio was first segmented into 4 seconds during the inference stage and classified by each context recognition model. Context label was then given based on the highest total log-likelihood accumulated over the audio. Based on the context label, events were then detected using the HMM models trained. Together with the use of event priors, such a

system can achieve a single-second segment-based F1-score of 19.5% and a 30-seconds segment-based F1-score of 29.4%.

Based on the results, such a system could not win a monophonic HMM-GMM system by a considerable margin and instead performed slightly worse in terms of the 30-seconds segment-based F1-score. The drawback of this system is the dependency on the context recognition accuracy. A wrongly recognized context can lead to wrong SED model selection and event priors. However, this complication was not fully reflected in their experiment as different contexts contain similar sound events, and some of the common events were correctly recognized. Also, the number of Viterbi (Forney, 1973) passes was estimated based on the average polyphony of the recorded materials. However, the number of events in test data is usually not known a priori.

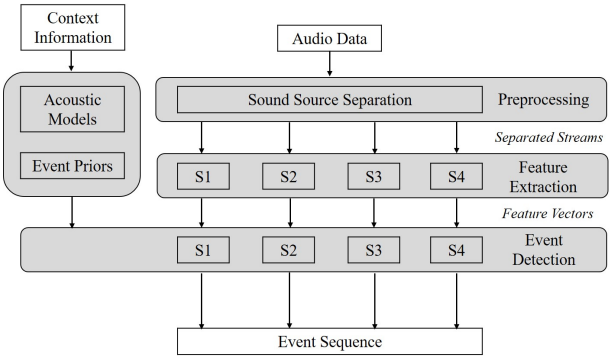


Figure 2.3 Proposed methodology by Heittola et al. (2013b)

Heittola et al. (2013b) then extended their work by including source separation before SED. In addition, the ground truth of the context was known with certainty and given to the SED system, which eliminates the need to train a context recognition system, unlike the earlier work (Heittola et al., 2013a).

As seen in Figure 2.3, the framework began with the source separation where NMF was utilized to decompose the spectrum into 4 different components. The number of components was determined based on the average amount of overlapping events in the evaluation dataset. Since each component may contain one or more sound sources, Heittola et al. (2013b) proposed using a Wiener filter on each component to separate the stream, which contained roughly homogenous spectral content and differs significantly from the others.

As the source separation was unsupervised, there was no knowledge of which event was separated into which stream. Thus, Heittola et al. (2013b) suggested two approaches to select the stream that contained the target event using the EM algorithm. The first approach was to choose the most prominent stream based on the highest likelihood. The second

approach was to perform iterative elimination of the least possible stream that contained the target event. However, it was found that results from both schemes were comparable, but the stream elimination scheme had the advantage of faster convergence and being straightforward. Separated streams from their respective event class then have features such as MFCCs, delta MFCCs and delta-delta MFCCs extracted where they were used to model a three-state left-to-right HMM with 16 Gaussians per state.

During the testing stage, test audio will be decomposed into 4 different streams, and the SED system will process each stream and combine the results into a single set of events. Based on the proposed methodology, Heittola et al. (2013b) reported a single-second segment-based F1-score of 44.9% and an average 30-seconds segment-based F1-score of 60.8%, which was approximately a double of their earlier work (Heittola et al., 2013a).

However, such a system requires the context of testing audio to be known a priori, which may limit its applicability. Similarly, the number of components to be separated was based on the average number of overlapping events in the evaluation dataset and thus cannot be known beforehand. Also, there may be a risk of selecting the wrong stream for event training.

Although GMM-HMM had seen its success in ASR, it does not seem to be very effective and accurate for SED. This may be due to the fact that GMM has a serious shortcoming where they cannot effectively exploit information embedded in a large window of frames (Hinton et al., 2012). Another major drawback of using GMM-HMM is the need to train an individual model to represent an event class.

Based on the results seen earlier, the use of MFCCs as the model input may also be unsuitable. Cakir et al. (2015b) suggested that the calculation of MFCCs may result in a loss of information as MFCCs are made up of the first few coefficients after the application of Discrete Cosine Transform (DCT). Moreover, the sum of MFCCs of different sound sources is not the same as the sum of MFCCs of the mixtures of these sources.

2.1.2. Nonnegative Matrix Factorization

Another popular methodology for SED is the use of NMF (Lee and Seung, 1999). NMF (Lee and Seung, 1999) is a matrix decomposition method where the objective is to decompose a nonnegative matrix $\ddot{\mathbf{V}} \in \mathbb{R}_{+}^{m \times n}$ into two nonnegative matrices $\ddot{\mathbf{W}} \in \mathbb{R}_{+}^{m \times r}$ and $\ddot{\mathbf{H}} \in \mathbb{R}_{+}^{r \times n}$. r, in this case, represents the number of components. The linear combination of $\ddot{\mathbf{W}}$ and $\ddot{\mathbf{H}}$ would then produce an estimated $\ddot{\mathbf{V}}$, which can be represented as $\tilde{\mathbf{V}}$. Mathematically, it can be defined as

$$\ddot{\mathbf{V}} \approx \ddot{\mathbf{W}}\mathbf{H} \tag{2.1}$$

To find an optimal $\tilde{\mathbf{V}}$, Lee and Seung (2000) proposed an efficient multiplicative update rule for $\ddot{\mathbf{W}}$ and $\ddot{\mathbf{H}}$, which can be defined as

$$\ddot{\mathbf{W}}_t = \ddot{\mathbf{W}}_{t-1} \otimes \frac{\frac{\ddot{\mathbf{V}}}{\ddot{\mathbf{W}}_{t-1}\ddot{\mathbf{H}}_{t-1}}\ddot{\mathbf{H}}_{t-1}^{\top}}{\mathbf{1}^n\ddot{\mathbf{H}}_{t-1}^{\top}} \tag{2.2}$$

$$\ddot{\mathbf{H}}_t = \ddot{\mathbf{H}}_{t-1} \otimes \frac{\ddot{\mathbf{W}}_{t-1}^{\top}\frac{\ddot{\mathbf{V}}}{\ddot{\mathbf{W}}_{t-1}\ddot{\mathbf{H}}_{t-1}}}{\ddot{\mathbf{W}}_{t-1}^{\top}\mathbf{1}^m} \tag{2.3}$$

where $\ddot{\mathbf{W}}_t$ and $\ddot{\mathbf{W}}_{t-1}$ represent $\ddot{\mathbf{W}}$ at time step t and $t-1$. $\ddot{\mathbf{H}}_t$ and $\ddot{\mathbf{H}}_{t-1}$ represent $\ddot{\mathbf{H}}$ at time step t and $t-1$. $\mathbf{1}^m$ and $\mathbf{1}^n$ represent an m and n dimensional vector of ones, respectively.

In practice, $\ddot{\mathbf{W}}$ and $\ddot{\mathbf{H}}$ are randomly initialized, and in each time step, both $\ddot{\mathbf{W}}$ and $\ddot{\mathbf{H}}$ are updated. In a regular coding environment, the updating procedure is sequential, meaning either $\ddot{\mathbf{W}}$ or $\ddot{\mathbf{H}}$ is updated first before updating the other matrix. If one chooses to update $\ddot{\mathbf{W}}$ first, then when $\ddot{\mathbf{H}}$ is updated, the calculation of the new $\ddot{\mathbf{H}}$ should utilize the updated $\ddot{\mathbf{W}}$. The updating procedure can be terminated when the cost function that measures the quality between $\ddot{\mathbf{V}}$ and $\tilde{\mathbf{V}}$ cannot be lowered any further or has reached the user's expectation, i.e., pre-defined threshold.

As discussed in the previous section, NMF was used by Heittola et al. (2013b) in an unsupervised manner where r is set as 4 for audio source separation before SED (for more information, please refer to the earlier section or (Heittola et al., 2013b)). On the other hand, NMF can also be used as a supervised approach for SED. In the SED domain, $\ddot{\mathbf{W}}$ can be considered as a dictionary that contains a collection of spectral templates. In contrast, $\ddot{\mathbf{H}}$ can be considered as the activation matrix which can be used to locate the activated frames (i.e., the occurrence of an event in an audio clip). Thus the first step to applying the conventional supervised NMF approach is to extract spectral templates from isolated events to form a dictionary (Bui et al., 2016; Smaragdis and Brown, 2003). Subsequently, the test data is decomposed using the consolidated dictionary based on Equation 2.1 while keeping $\ddot{\mathbf{W}}$ constant (i.e., $\ddot{\mathbf{W}}$ will not be updated using the multiplicative rule).

Although the algorithm's simplicity makes it attractive, it has difficulty detecting overlapping sounds (Smaragdis and Brown, 2003). While this problem can be overcome by modeling overlapping sound as an additional class (Smaragdis and Brown, 2003), the combination of different events would make the training intractable.

As such, research efforts then focused on improving the conventional supervised NMF approach for SED. Mesaros et al. (2015) proposed a coupled NMF where the idea was to learn two non-negative dictionary matrices jointly, $\ddot{\mathbf{W}}_1$ and $\ddot{\mathbf{W}}_2$ with a similar $\ddot{\mathbf{H}}$. Thus, the coupled NMF problem can be considered to minimize (Mesaros et al., 2015)

$$\varepsilon_1 D_1(\ddot{\mathbf{V}}_1||\ddot{\mathbf{W}}_1\ddot{\mathbf{H}}) + \varepsilon_2 D_2(\ddot{\mathbf{V}}_2||\ddot{\mathbf{W}}_2\ddot{\mathbf{H}}) \tag{2.4}$$

where ε_1 and ε_2 are the weights associated with divergences and are set as 1, respectively. $D(\ddot{\mathbf{V}}||\ddot{\mathbf{W}}\mathbf{H})$ is the divergence between $\ddot{\mathbf{V}}$ and $\ddot{\mathbf{W}}\mathbf{H}$. $\ddot{\mathbf{V}}_1$ and $\ddot{\mathbf{V}}_2$ represent the audio frequency spectrum and a frame-level one hot encoding matrix that contains the information about the events' occurrences. The proposed methodology by Mesaros et al. (2015) is illustrated in Figure 2.4.

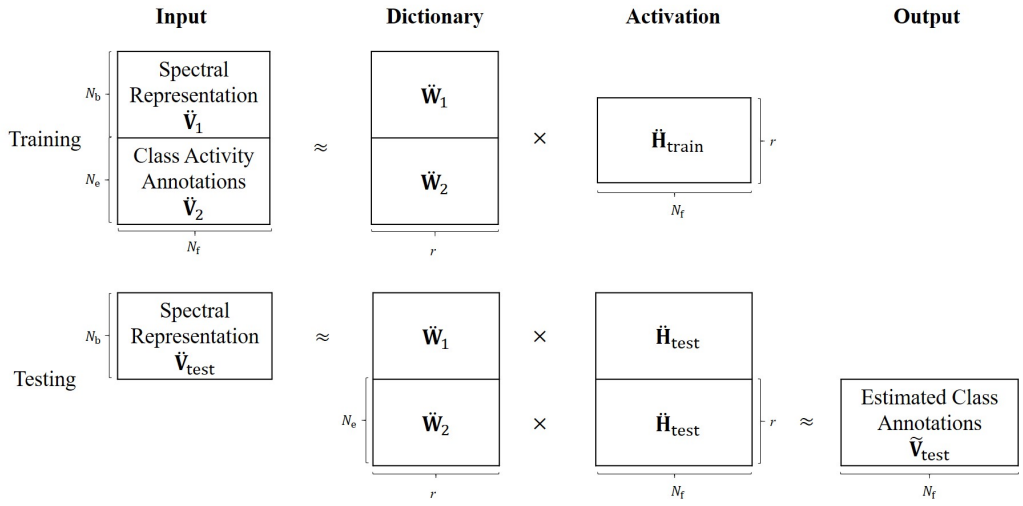


Figure 2.4 Proposed methodology by Mesaros et al. (2015)

In Figure 2.4, N_b and N_f represent the number of frequency bins and the number of frames of a spectral representation. N_e represents the number of events.

As the size of $\ddot{\mathbf{W}}_1$ is relative to the size of training data, this may not be computationally feasible. Thus, Mesaros et al. (2015) reduced the size of the dictionary through clustering of components and only allowed the centroid of clusters to form the dictionary, which allows computation efficiency to increase.

In order to detect the presence of a sound event in test audio, NMF was first applied using $\ddot{\mathbf{W}}_1$ to derive $\ddot{\mathbf{H}}_{\text{test}}$. $\ddot{\mathbf{H}}_{\text{test}}$ was then used to reconstruct the frame-level one hot encoding matrix using $\ddot{\mathbf{W}}_2$. The values in the encoding matrix were then converted to a binary form to determine the occurrence of an event. This is carried out using the mean value of the encoding matrix as the threshold. Values above the mean value were converted to 1 to indicate the occurrence of an event and 0 otherwise. To prevent noise and outliers,

detected sound events that did not correspond to a minimum of 200ms duration were discarded.

Based on such a setup, Mesaros et al. (2015) reported an average single-second segment-based F1 score of 57.8% on TUT-SED 2009. The most interesting result was that a dictionary formed using only cluster centroids could achieve a higher F1-score than the entire dictionary.

Although using cluster centroids can reduce the computation cost, there is a need to perform clustering whenever new training data is added. In addition, the optimal cluster number has to be derived through several trials. Finally, using the mean as a threshold may not be the best value to achieve maximal performance.

Bisot et al. (2017) proposed a methodology to learn a classifier and the NMF in a joint optimization problem referred to as nonnegative task-driven dictionary learning (TD-NMF). In essence, the methodology learns an optimized dictionary through NMF with beta divergence by minimizing the classification cost of a regularized linear logistic regression. Bisot et al. (2017) explained that this would allow a more discriminative dictionary of spectral templates to be learned.

Once the dictionary is learned, the test data can then be projected onto the learned dictionary. The projections from the test data were subsequently used as a feature for classification. To accommodate multi-label classification, the class probabilities of each test frame were thresholded by a value that was dependent on the acoustic scene (0.3 for the home environment and 0.35 for the residential area). Thus, overlapping events can be detected as long as the probabilities in a given frame exceed the fixed threshold.

Based on this methodology, Bisot et al. (2017) reported a single second segment based F1-Score of 49.5% with an ER of 69.5 when test is performed on the TUT-SED 2016 development dataset. While this methodology achieved the lowest ER compared to methods such as GMM, Random Forest (RF), Gated Recurrent Neural Network (GRNN), RNN, and NMF, it could not achieve the highest F1-score (RNN made a slightly higher F1-score of 49.8%). While such a methodology shows competitive results, it was only tested on a small dataset with a total duration of approximately 80 minutes, where the number of overlapping events is minimal. Finally, there is also a need to fine-tune the threshold to achieve maximal performance.

Ohishi et al. (2013) modeled the overlapping sound event using NMF and Bayesian nonparametric approaches (i.e., Markov Indian Buffet Process (mIBP) and Chinese Restaurant Process (CRP)). Such an approach removes the need to predefine the total number of

events that can be present and temporal labels can be estimated using the Bayesian logistic regression.

Such a method was subsequently tested on an English learning podcast, and Ohishi et al. (2013) reported an accuracy (in terms of Area Under Curve (AUC)) of 0.79, which outperforms a baseline GMM and three other variants of proposed methods.

However, the performance of this methodology appears to be inconclusive and biased. Firstly, it was only tested on a short English learning podcast (approximating 150s) prepared in a controlled environment. Secondly, the GMM may not be effective using only 100s of training data, and thus, the comparison does not appear to be fair. Thirdly, there was no training data for one of the event categories (i.e., Female C); it is unclear how the classification of this event was achieved. In addition, using AUC as the accuracy metric is not appropriate; almost the entire block of the testing signal was annotated as Female A and Female B, but the AUC calculated was 0.769 and 0.744, respectively. Lastly, it was mentioned by Ohishi et al. (2013) that the Poisson likelihood model used in their study suffers from theoretical issues where it is only applicable to discrete counts data (Hoffman, 2012).

2.1.3. Summary of Non-Neural Network Based Methodologies

The previous subsections showcased the different non-NN architectures proposed by various authors, and discussions on their methodologies and limitations were done.

In this section, details on the features used, architectures, performances, and limitations are summarized in the following tables. Table 2.1 to Table 2.3 provide the information on features used and their respective processing method. For non-NN based methodologies, there is a wide variety of features used, with MFCC forming the majority. However, the use of MFCC may result in the loss of information due to the application of DCT (Cakir et al., 2015b).

Table 2.4 provides the information on the architectures used. Among the different methodologies, some utilized context information to provide a set of rules for event detection. However, detection accuracy is highly dependent on the context recognition accuracy, which may limit such a strategy. Table 2.5 then showcases the results of their proposed methodology on different types of datasets. Although non-NN based methodologies may not require a large amount of strongly labeled data, as seen in Table 2.5, they do not perform very well. Table 2.6 then presents the summary of each methodology's limitations.

References	Features
Mesaros et al. (2010)	MFCCs Delta-MFCCs Delta-delta MFCCs
Heittola et al. (2013a)	MFCCs Delta-MFCCs Delta-delta MFCCs
Heittola et al. (2013b)	MFCCs Delta-MFCCs Delta-delta MFCCs
Mesaros et al. (2015)	Spectrogram
Bisot et al. (2017)	Mel spectrum
Ohishi et al. (2013)	Mel spectrum

Table 2.1 Features proposed for non-NN based methodologies

References	Window	Window length (ms)	Overlap (%)	Mel-filterbanks
Mesaros et al. (2010)	Hamming	20	50	40
Heittola et al. (2013a)	-	20	50	40
Heittola et al. (2013b)	-	20	50	-
Mesaros et al. (2015)	-	100	50	-
Bisot et al. (2017)	-	40	50	40
Ohishi et al. (2013)	Hanning	100	-	25

Table 2.2 Parameters used for feature calculation (non-NN based methodologies)

References	Additional processing steps
Mesaros et al. (2010)	-
Heittola et al. (2013a)	Frequency range to computer MFCCs is set to 30Hz to 22050Hz.
Heittola et al. (2013b)	-
Mesaros et al. (2015)	1024 bins for Fast Fourier Transform (FFT).
Bisot et al. (2017)	Min-max normalization before Short Time Fourier Transform (STFT)
Ohishi et al. (2013)	-

Table 2.3 Additional processing steps (non-NN-based methodologies)

References	Proposed algorithm	additional information
Mesaros et al. (2010)	GMM-HMM	• EM algorithm for training. • Viterbi algorithm for optimal sequence decoding. • Each event class model represented by three state left to right HMMs with 16 Gaussians per state.
Heittola et al. (2013a)	GMM-HMM	• GMM for context recognition. • EM algorithm for training. • Each event class model represented by three state left to right HMMs with 16 Gaussians per state.
Heittola et al. (2013b)	GMM-HMM	• NMF for source separation. • EM algorithm for training. • Viterbi algorithm for optimal sequence decoding. • Each event class model represented by three state left to right HMMs with 16 Gaussians per state.
Bisot et al. (2017)	NMF-Logistic Regression	• NMF for matrix decomposition. • Multinomial logistic regression for event detection.
Ohishi et al. (2013)	NMF-Bayesian Nonparametric	• NMF with mIBP and CRP for modeling overlapping audio events. • Bayesian logistic regression for event annotations.

Table 2.4 Non-NN based methodologies

References	Dataset	Segment-based F1-score (%)	Other metric
Mesaros et al. (2010)	TUT-SED 2009	-	F1-Score: 30.1% ER: 84.1
Heittola et al. (2013a)	TUT-SED 2009	1-Second: 19.5 30-Seconds: 29.4	-
Heittola et al. (2013b)	TUT-SED 2009	1-Second: 44.9 30-Seconds: 60.8	-
Mesaros et al. (2015)	TUT-SED 2009	1-Second: 57.8	-
Bisot et al. (2017)	TUT-SED 2016 Development Dataset	1-Second: 49.5	1-Second Segment Based ER: 69.5
Ohishi et al. (2013)	English Learning Podcast	-	AUC: 0.79

Table 2.5 Dataset used by different authors and reported accuracy using non-NN-based methodologies

2.2. Neural Network Based Methodology

2.2.1. Non Hybrid Models

A Feedforward Neural Network (FNN), Deep Neural Network (DNN), or Multilayer Perceptrons (MLP) can be considered as a quintessential deep learning model where the objective is to provide a non-linear mapping between an input vector to a category (Goodfellow et al., 2016).

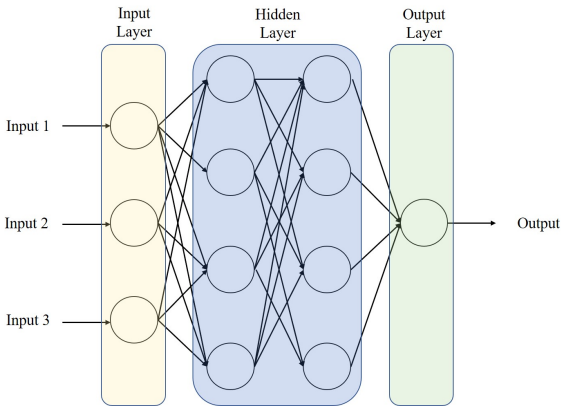


Figure 2.5 Example of an MLP

The term deep in deep learning does not represent a deeper understanding of the problem but rather the depth of successive layers of representations (Chollet, 2017a). As shown in Figure 2.5, a basic MLP would consist of three types of layers; 1) input layer, 2)

References	Limitations
Mesaros et al. (2010)	• Each event class has to be modeled by an individual HMM. • GMM cannot effectively exploit information embedded in large window of frames. • MFCC may not be a good feature. • Requires a large number of model to be trained to represent each event class. • Low accuracy with large error rate. • It can only detect the most prominent event at a time.
Heittola et al. (2013a)	• Each event class has to be modeled by an individual HMM. • GMM cannot effectively exploit information embedded in large window of frames. • MFCC may not be a good feature. • Requires a large number of model to be trained to represent each event class. • Accuracy of this system did not win a monophonic GMM-HMM system by a huge margin. • Accuracy is dependent on the context recognition accuracy. • Appropriate number of Viterbi passes was estimated based on data.
Heittola et al. (2013b)	• Each event class has to be modeled by an individual HMM. • GMM cannot effectively exploit information embedded in a large window of frames. • MFCC may not be a good feature. • Requires a large number of model to be trained to represent each event class. • Context of testing audio must be known a priori. • Appropriate number of components in each test stream is estimated based on evaluation dataset. • May have risk of selecting the wrong stream for event training.
Mesaros et al. (2015)	• Need to perform clustering for every addition of new training data. • Optimal cluster number need to be derive through several trials. • Requires careful tuning of threshold value.
Bisot et al. (2017)	• Training data cannot contain overlapping events. • Requires careful tuning of threshold value.
Ohishi et al. (2013)	• Only tested on a small dataset collected in a controlled environment. • Comparison and results are not conclusive and appear bias. • Poisson likelihood model used in the study suffers from theoretical issues.

Table 2.6 Limitations for different non-NN based methodologies

hidden layer, and 3) output layer, and the example shown would be considered an MLP with two hidden layers. The core fundamental of training a NN is through an updating procedure known as the backpropagation algorithm.

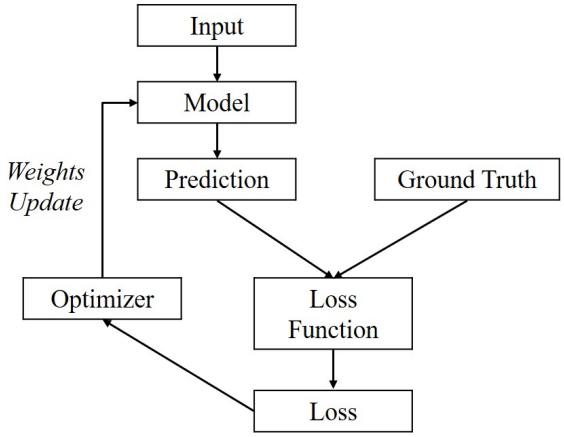


Figure 2.6 Backpropagation algorithm

As seen in Figure 2.6, the idea of backpropagation is to compare a model's prediction and the ground truth and using this difference to update the weights in the model. Due to the advances in machine learning algorithms and computer hardware, an MLP containing many hidden layers and a large output layer can now be trained efficiently (Hinton et al., 2012). This allows deep learning models to learn much better feature representations and appropriate classifiers (Takahashi et al., 2018), where successes can be seen in many sound-related domains such as ASR (Ferroni et al., 2015; Hinton et al., 2012; Snyder et al., 2015), sound event or environmental sound classification (Gencoglu et al., 2014; Piczak, 2015), and source separation (Du et al., 2014; Zhang and Wang, 2016)].

Such successes make an MLP a viable and attractive choice for SED. Moreover, an MLP architecture allows multi-label classification directly without additional training efforts. In contrast, non-NN models such as GMM-HMMs require additional effort to train individual models for each class or allowing multiple passing of the Viterbi algorithm (Forney, 1973) for polyphonic SED (Mesaros et al., 2019).

However, conventional NN-based methodologies only allow single-label classification (or multi-class classification) due to the softmax layer. Thus the easiest and the most common way to accommodate multi-label classification is to change the softmax layer to a sigmoid layer and threshold the event class probabilities given by the NNs. This would allow multiple sound events to be detected as long as the probabilities are above the predefined threshold.

In the SED domain, Cakir et al. (2015b) proposed using an FNN with maxout activation function (Note: Such network may also be referred to as a Maxout Networks (Goodfellow

et al., 2013)). The idea of a maxout unit is to facilitate optimization by dropout and improve the accuracy of dropout's fast approximate model averaging technique (Goodfellow et al., 2013). Given an input, **x**, a maxout unit implements the following function (Goodfellow et al., 2013),

$$h(\mathbf{x}) = \max(\mathbf{xW} + \mathbf{b}) \tag{2.5}$$

where **W** and **b** are learned parameters (i.e., learnable weights and bias). This function retrieves the max of input and can be interpreted as making a piecewise linear approximation to an arbitrary convex function (Goodfellow et al., 2013). Cakir et al. (2015b) explained that such function is not bounded, easy to optimize, and does not suffer from vanishing gradients. Most importantly, the maxout function shows superior results than the sigmoid function in speech-related tasks (Swietojanski et al., 2014; Swietojanski and Renals, 2014).

In order to model the dynamic properties of sounds, Cakir et al. (2015b) proposed the use of context windowing (window length of 5) where the audio frame of extracted feature vectors was concatenated with adjacent time frames to form a single training instance. In addition, the output from the FNN was smoothed by a median filter to remove noise. The training procedure can be illustrated in Figure 2.7.

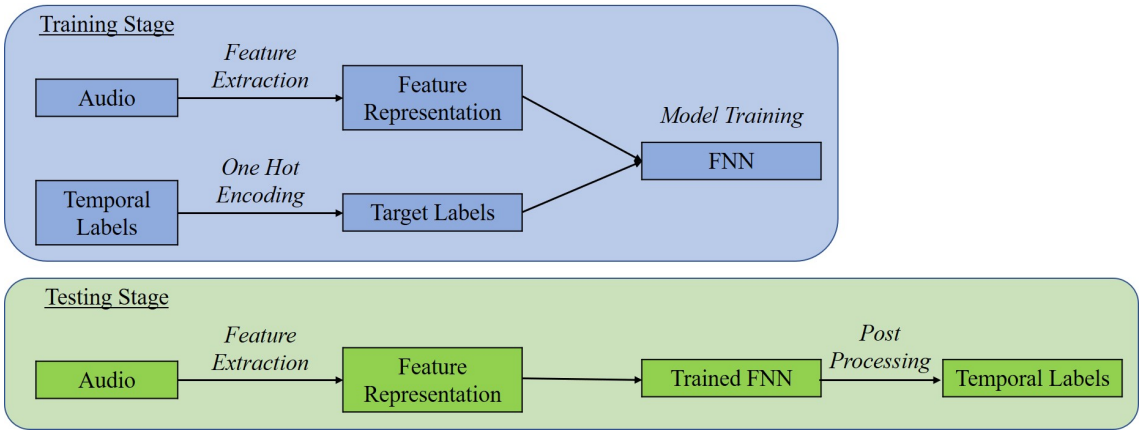


Figure 2.7 Proposed methodology by Cakir et al. (2015b)

Using a two layers FNN, Cakir et al. (2015b) reported a single-second segment-based F1-score of 63.8% and concluded that using log-mel band energies as an input feature was much better than using MFCC and mel-band energies.

Cakir et al. (2015a) then extended their work by decomposing the multi-label classifier into an ensemble of single-label classifiers. While this appears unnecessary, Cakir et al. (2015a) argued that the benefit of such a method is that it allows the dynamic inclusion of new labels since only a smaller classifier is trained for the new event class instead of

retraining the entire FNN. However, such implementation would require a test clip to be tested by multiple models to enable polyphonic SED. Such implementation is illustrated in Figure 2.8.

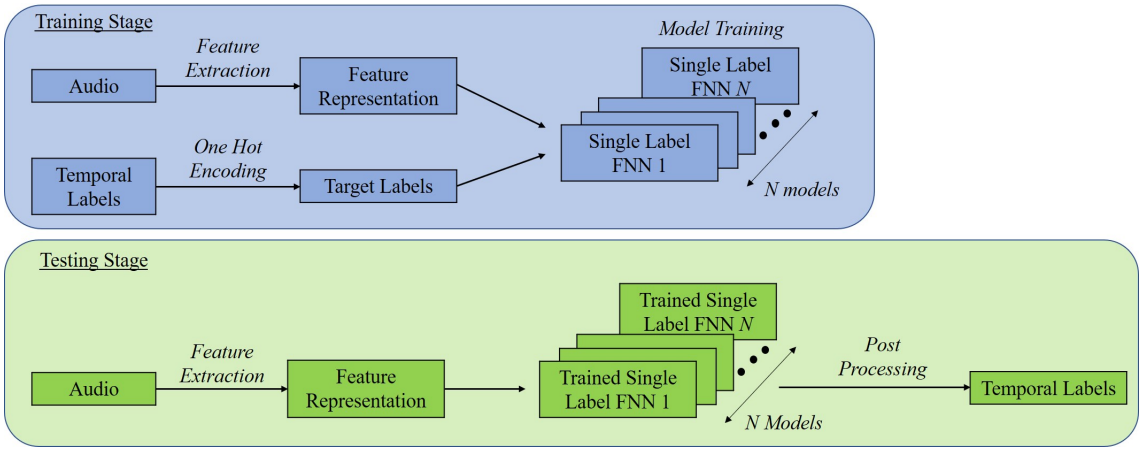


Figure 2.8 Proposed methodology by Cakir et al. (2015a)

Based on such an implementation, Cakir et al. (2015a) reported a single-second segment-based F1 score of 61.9%, which was slightly lower than a multi-label DNN accuracy. Although there is no need to retrain the entire model with new events, a single label classifier requires training of multiple classifiers that may need to be tuned separated to achieve maximal performance. In addition, the testing time for such architecture may be a few folds higher than a multi-label classifier.

In addition, both architectures that utilized the maxout functions may have several other disadvantages. Firstly, a maxout function doubles the number of parameters for every single unit, which may lead to a high parameter number (Castaneda et al., 2019). In addition, a maxout network is prone to overfitting (Cai et al., 2014). This is because the max function only propagates the gradient to the unit with the maximum value, and the remaining units do not get updated (Toth, 2015). Finally, an FNN with maxout function may not be comparable to a CNN with maxout function based on the result shown in (Renals and Swietojanski, 2014).

Parascandolo et al. (2016) proposed Bidirectional Long Short Term Memory (BLSTM) for polyphonic SED, which allows data processing in both directions by utilizing two separate hidden layers. These two layers are subsequently concatenated and fed to the same output layer (Graves et al., 2013). As such, it allows access to long-range context in both input directions, which can help in the classification, and Parascandolo et al. (2016) hypothesized that it might eliminate the need for tailored post-processing or smoothing steps.

In the preprocessing stage, audio recordings were normalized to a scale of -1 to 1 before calculating mel energies to account for audio recorded in different conditions. As an additional measure to reduce overfitting, Parascandolo et al. (2016) increased the dataset by 16 times using several data augmentation techniques such as time-stretching, sub-frame time-shifting, and blocks mixing. Parascandolo et al. (2016) also proposed adding Gaussian noise to the network weights. Such noise addition was found to 'simplify' RNN by reducing the amount of information required to transmit the parameters, which can improve generalization (Graves et al., 2013b).

Parascandolo et al. (2016) then compared against an FNN, vanilla LSTM, and a BLSTM without data augmentation on TUT-SED 2009. The final results showed that their method with data augmentation was the best performing architecture with a single-frame segment-based F1-score of 64.7% and a single-second segment-based F1-score of 65.5%. However, it was only marginally better than a BLSTM without data augmentation, which achieved a single frame segment F1-score of 64.0% and a single second F1-score of 64.6%. Such a result implies that simple data augmentation may not be helpful, given the considerable increase in preprocessing efforts and computational cost and yet negligible impact on model accuracy.

While Parascandolo et al. (2016) hypothesized that output from an RNN was already smoothed, but empirical results shown by (Hayashi et al., 2016) proved that post-processing is still necessary, especially for event-based evaluation. Although Gaussian noise injection can improve generalization, it was also found that such a method can also increase the training time and affect the performance of an LSTM (Greff et al., 2015). In addition, the LSTM has a relatively high model complexity, and parameter tuning for LSTMs is not always straightforward (Zeyer et al., 2017; Zohrer and Pernkopf, 2017). Finally, the sequential nature of LSTM prohibits parallelization, which may lead to a long training time (Martin and Cundy, 2018).

Adavanne et al. (2016) also proposed the use of LSTM but with additional input features such as the pitch and its periodicity and the Time Difference of Arrival (TDoA) in sub-bands. Pitch and periodicity were estimated using Librosa implementation of pitch tracking on thresholded parabolically interpolated STFT. On the other hand, TDoA was calculated using the generalized cross-correlation with phase-based weighting (GCC-PHAT). A median filter was then applied to the estimated TDoA to remove noise. Similar to (Parascandolo et al., 2016), block mixing was applied to increase the training data to reduce overfitting. Besides comparing with a GMM, Adavanne et al. (2016) also examined the performance difference between mono-channel and stereo-channel features.

Based on the ER, Adavanne et al. (2016) concluded that LSTM trained using log mel band energies and TDOA from the stereo channel was the best classifier with an ER of 0.91 and a single-second segment-based F1-score of 35.4%. If the results were based on the highest F1-score, LSTM trained using mel energies and pitch would be the best classifier that achieved a single-second segment-based F1-score of 35.7% with an ER of 0.92.

Although results showed that features extracted from the stereo channel were beneficial, additional features such as pitch and TDOA did not appear to provide many benefits. The conclusion derived from the fact that an LSTM trained using only log mel energies calculated from the stereo channel can already achieve a single second segment F1-score of 35.6% with an ER of 0.93. Moreover, the only system that won the baseline system in the DCASE 2016 challenge was only trained with mel energies from both channels and not with the proposed additional features. Besides the redundancy of extra features, LSTM also has some limitations, which we mentioned earlier.

Xia et al. (2018) proposed a regression-based CNN for SED. Xia et al. (2018) explained that multi-label classification using frame-wise labeling might not be accurate due to the annotation errors. Thus, Xia et al. (2018) proposed the soft labeling of events in each recording based on a confidence measure. In order to estimate the confidence measure, a parabolic function is used where the peak of the parabola was positioned at the center frame of the manually labeled event (i.e., the center frame has the highest confidence). This would allow a continuous representation for each acoustic event, and temporal labels will be represented as real positive numbers instead of discrete numbers.

Based on such implementation, Xia et al. (2018) reported a segment-based F1-score of 61.02% with an ER of 0.63 on the TUT-SED 2017 development dataset. On the other hand, a segment-based F1-score of 45.3% with an ER of 0.84 were reported on the evaluation dataset. Such results can be ranked fourth in the DCASE 2017 task 3 challenge (in terms of ER).

However, there are several limitations associated with their implementations. Firstly, the small number of layers and filters may not be sufficient to learn the complex structure of polyphonic audio. Secondly, CNN is unable to extract long temporal context information (Cakir et al., 2017). Thirdly, using a parabola as a confidence function may not be appropriate. This is because events with continuous output without too much fade-in fade-out effect, such as vacuum cleaner or blender, should have maximal confidence throughout the entire annotated frames instead of just the center frame. Finally, the hyperparameter for the parabolic function may require careful tuning to achieve maximal performance.

Vesperini et al. (2019) proposed Capsule Neural Network (CapsNet) for polyphonic SED. The introduction of CapsNet is to overcome some limitations of CNN, particularly the loss of information due to the application of the max pooling operator (Patrick et al., 2019).

A capsule can be thought of as a group of neurons whose output represents different properties of the same entities (Hinton et al., 2018). As explained by Hinton et al. (2018), NN typically uses simple non-linearities in which a non-linear function is applied to the scalar output of a linear filter. They may also use softmax non-linearities that convert a whole vector of logits into a vector of probabilities. On the other hand, capsules use a much more complicated non-linearity that converts the whole set of activation probabilities and poses of the capsules in one layer into the activation probabilities and poses of capsules in the next layer (Hinton et al., 2018).

A CapsNet consists of a convolutional layer for feature extraction and multiple capsule layers starting a primary capsule layer and ending with a class capsule layer. A primary capsule layer represents the lowest level of multi-dimensional entities and contains reshaping and squashing functions. Outputs from the primary capsule layer are then passed to the class capsule layer with one capsule per output class through a dynamic routing procedure. Event probabilities are then obtained by computing the Euclidean norm of the output of each capsule. The flowchart of a Capsnet used in (Vesperini et al., 2019) is illustrated in Figure 2.9.

Using CapsNet, Vesperini et al. (2019) reported an ER of 0.36 on the TUT-SED 2016 and TUT-SED 2017 development dataset using a binaural spectrogram as the input. On the other hand, the TUT-SED 2017 evaluation dataset results show that a CapsNet trained using log mel energies achieved the lowest ER of 0.58 instead of using a binaural spectrogram as input. Regardless of the inputs, CapsNet remains the best classifier (in terms of ER) compared to CNN, CRNN, and GMM. However, the results did not include the F1-score. Therefore, it is unclear how well such a classifier performs in terms of F1-score.

The drawback for CapsNet is that even for simple architecture, training CapsNet requires significant computational resources (Mukhometzianov and Carrillo, 2018) and training time can be much longer than a CNN (Jiang et al., 2018). The performance is also highly sensitive towards the hyperparameters (Vesperini et al., 2019). Moreover, CapsNet can have a more significant performance fluctuation during training, and if the fixed number of training epochs is suboptimal, CapsNet can be more prone to significant errors as compared to CNNs. In addition, a CapsNet also appears to show lesser generalization ability compared to a CNN (Vesperini et al., 2019). Finally, AdaDelta (Zeiler, 2012), which

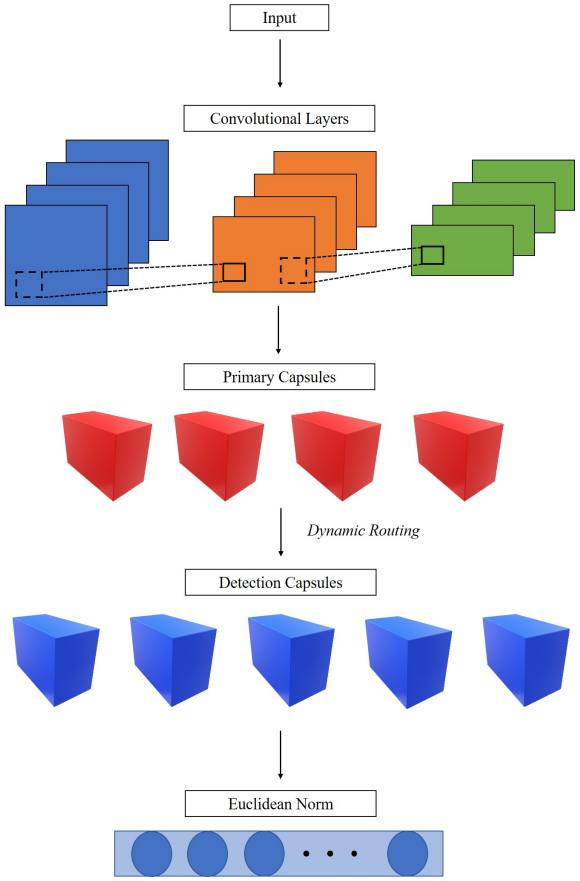


Figure 2.9 Flowchart of a CapsNet (Vesperini et al., 2019)

was used as the gradient optimizer in their architecture, can take a longer time to converge than Adam (Kingma and Ba, 2015) due to its iterative operation (Kim et al., 2018; Okewu et al., 2019).

2.2.2. Summary of Non Hybrid Models

The previous section showcased the different NN architectures proposed by various authors, and limitations on respective methodologies were discussed. In this section, details on the features used, architectures, accuracy, and limitations are summarized in the following tables.

Table 2.7 to Table 2.9 provides the information on features used and their respective processing method. It is evident that mel energies are the most popular input features, and there is not much difference in the way they are calculated. The MFCC could cause loss of information, and empirical results shown in (Cakir et al., 2015b) highlighted that using mel energies as a feature can increase the detection accuracy. The key difference in the input lies in the post-processing, where several authors proposed normalizing the energy bands.

Table 2.10 to Table 2.12 provides information on the architectures used and showcases the key configurations (if reported). In Table 2.10, there is a column named No. hidden

layers, {No. units / filters} with value given in the following format, 2, {800, 800}. This means that there are 2 hidden layers with 800 units in the first layer and 800 units in the second layer.

In Table 2.12, there is an entry known as the Early stop criterion. This is the number of epochs that the authors used to evaluate if there is any further performance improvement. Thus, for the Early stop criterion: 20 epochs, it means that the author proposed to stop the model's training if accuracy did not increase or did not have a significant increase after 20 epochs. Whereas, for Kernel size or Pooling size, it refers to the size of the filter and pooling operator, respectively. For example, if Pooling size: (1, 3) means the pooling operator is using a pooling size of 3 and if Pooling size: (1, 4), (1, 3), (1, 2) means the pooling operator has a pooling size of 1 by 4 in the first layer and 1 by 3 in the second layer and 1 by 2 in the third.

Table 2.13 presents the results of their proposed methodology tested on different datasets. Although results generally have shown that NN-based methodologies can perform better than non-NN based methodologies, there is still a large room for improvement that can be made. It should also be pointed out that all methodologies used segment-based evaluation metrics that place lesser emphasis on the onset and offset of events than event-based metrics.

Finally, Table 2.14 summarized the different limitations of each proposed methodology. It is essential to point out that all the NN-based methodologies require a large amount of strongly labeled training data to learn the mapping between features and event class which can be a significant limiting factor.

References	Features
Cakir et al. (2015b)	Log mel energies
Cakir et al. (2015a)	Mel energies
Parascandolo et al. (2016)	Log mel energies
Adavanne et al. (2016)	Log mel energies Pitch and its periodicity TDoA
Xia et al. (2018)	Log mel energies
Vesperini et al. (2019)	Log mel energies

Table 2.7 Features proposed for non-hybrid NN-based methodologies

References	Window length (ms)	Overlap (%)	Mel-filterbanks
Cakir et al. (2015b)	50	50	40
Cakir et al. (2015a)	50	50	40
Parascandolo et al. (2016)	50	50	40
Adavanne et al. (2016)	50	50	40
Xia et al. (2018)	-	-	-
Vesperini et al. (2019)	40	50	40

Table 2.8 Parameters used for feature calculation (non-hybrid NN-based methodologies)

References	Additional processing steps
Cakir et al. (2015b)	-
Cakir et al. (2015a)	• Min-max normalization before STFT.
Parascandolo et al. (2016)	• Min-max normalization before STFT. • Z-score normalization of energy band. • Data augmentation using time stretching, subframe time shifting and block mixing.
Adavanne et al. (2016)	• Z-score normalization of feature vectors. • Data augmentation using block mixing. • Periodicity extracted in 100Hz-4000Hz. • TDOA calculated using window length of 120ms, 240ms, 480ms with 20ms hop. • Post processed of TDOA with median filter (kernel of length three).
Xia et al. (2018)	-
Vesperini et al. (2019)	• Min-max normalization before STFT.

Table 2.9 Additional processing steps (non-hybrid NN-based methodologies)

References	Model	No. hidden layers, {No. units/filters}	Activation function
Cakir et al. (2015b)	FNN	2, {800,800}	• Maxout at hidden layer • Sigmoid for output layer
Cakir et al. (2015b)	FNN	2, {400,400}	• Maxout at hidden layer. • Sigmoid for output layer.
Parascandolo et al. (2016)	BLSTM	4, {200,200,200,200}	• Hyperbolic tangent for memory cell. • Sigmoid for input, forget, output gates and output layer.
Adavanne et al. (2016)	LSTM	2, {32,32}	• Sigmoid for output layer
Xia et al. (2018)	LSTM	2, {32,32}	• Rectified Linear Unit (ReLU) for convolution layers • Sigmoid for output layer
Vesperini et al. (2019)	CapsNet	TUT-SED 2016 Home: 3,{32,32,8} TUT-SED 2016 Residential and TUT-SED 2017: 4,{4,16,32,4}	• ReLU for convolution and capsule layers • Sigmoid for output layer

Table 2.10 Non-hybrid NN architecture

References	Loss function	Optimizer
Cakir et al. (2015b)	Kullback Leibler (KL) Divergence	Stochastic Gradient Descent (SGD)
Cakir et al. (2015a)	BCE	SGD
Parascandolo et al. (2016)	Root Mean Square Error (RMSE)	RMSProp
Adavanne et al. (2016)	BCE	Adam (Kingma and Ba, 2015)
Xia et al. (2018)	MSE	Adam (Kingma and Ba, 2015)
Vesperini et al. (2019)	Margin Loss	Adadelta (Zeiler, 2012)

Table 2.11 Loss functions and optimizers used for training non-hybrid NN-based architecture

2.2.3. Hybrid Models

One of the most popular hybrid models for SED is the Convolution Recurrent Neural Network (CRNN). In actual application, the combination of CRNN is pretty straightforward. As seen in Figure 2.10, stack a CNN over an RNN so that the features map extracted by the CNN can be passed directly to the RNN. Outputs from the RNN are then passed to a FC layer to reshape the dimension, and a sigmoid activation function is usually applied. A global or class-specific threshold can then be applied for polyphonic SED.

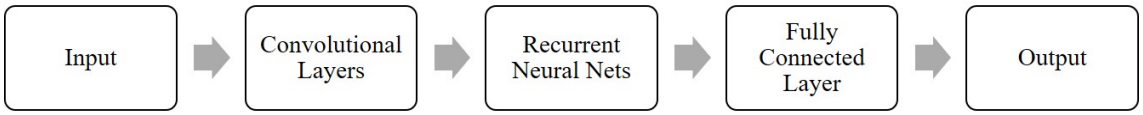


Figure 2.10 Example of a CRNN

The reason behind its popularity could be due to the combination of two can supplement each other. CNN can learn filters that are shifted in both time and frequency (Cakir et al., 2017) and can capture energy modulation patterns across time and frequency when applied to spectrogram-like inputs (Salamon and Bello, 2017). However, CNN lacks long temporal context information (Cakir et al., 2017). On the other hand, RNN can overcome this constraint by integrating information from an earlier time window, but it cannot capture the invariance in the frequency domain (Cakir et al., 2017). Thus, the combination of the two can overcome the shortcomings while providing the benefits of both approaches.

Cakir et al. (2017) proposed combining CNN with Gated Recurrent Unit (GRU) for polyphonic SED. The idea of GRU is similar to an LSTM, where its motivation is to overcome the vanishing or exploding gradient problems faced by a conventional RNN. A

References	Additional information
Cakir et al. (2015b)	• Initial weight range: +-0.001 • Learning rate: 0.02 • Mini-batch: 50 • Pooling operator: Max • Pooling size: (1, 3) • Context window: 5 frames • Median filter: 10 frames window
Cakir et al. (2015a)	• Learning rate: 0.02 • Context window: 2 frames • Median filter: 10 frames window
Parascandolo et al. (2016)	• Initial weight range: +-0.1 • Learning rate: 0.005 • Decay rate: 0.9 • Gaussian input noise: 0.2 • Early stop criterion: 20 epochs
Adavanne et al. (2016)	• Early stop criterion: 100 epochs
Xia et al. (2018)	• CNN Kernel size: (3, 3) • BN after each convolution layer • Pooling operator: Max • Pooling applied after every convolution layer
Vesperini et al. (2019)	TUT-SED 2016 Home • CNN Kernel size: (6, 6) • Pooling Operator: Max • Pooling size: (1, 4), (1, 3), (1, 2) • Primary capsule convolution capsule: 8 • Primary capsule kernel size: (4 , 4) • Routing iteration: 3 • Decay rate: 0.95 • Dropout rate: 0.2 • Mini-batch: 20 • Early stop criterion: 20
	TUT-SED 2016 Residential and TUT-SED 2017 • CNN kernel size: (4 , 4) • Pooling Operator: Max • Pooling size: (1, 2), (1, 2), (1, 2) • Pooling applied after every convolution layer • Primary capsule convolution capsule: 7 • Primary capsule kernel size: 3 by 3 • Routing iteration: 4 • Decay rate: 0.95 • Dropout rate: 0.2 • Mini-batch: 20 • Early stop criterion: 20

Table 2.12 Additional models information (non-hybrid NN-based architecture)

References	Dataset	Segment-based F1-score (%)	Segment-based ER	Other metric
Cakir et al. (2015b)	TUT-SED 2009	1-Second: 63.8		
Cakir et al. (2015a)	TUT-SED 2009	1-Second: 61.9		Hamming loss: 0.0325
Parascandolo et al. (2016)	TUT-SED 2009	1-Frame: 64.7 1-Second: 65.5		
Adavanne et al. (2016)	TUT-SED 2016 Development Dataset	1 Second: 35.4	1 Second: 0.91	
	TUT-SED 2016 Evaluation Dataset	1 Second: 47.8	1 Second: 0.80	
Xia et al. (2018)	TUT-SED 2017 Development Dataset	0.1 Second: 61.02	0.1 Second: 0.63	
Vesperini et al. (2019)	TUT-SED 2016 Development Dataset		1 Second: 0.36	
	TUT-SED 2016 Evaluation Dataset		1-Second: 0.69	
	TUT-SED 2017 Development Dataset		1-Second: 0.36	
	TUT-SED 2017 Evaluation Dataset		1-Second: 0.58	

Table 2.13 Dataset used by different authors and reported accuracy using non-hybrid NN-based methodologies

References	Limitations
Cakir et al. (2015b)	• Maxout function doubles the number of parameters for every single unit. • Maxout network is prone to overfitting. • FNN with maxout function may not perform as well as a CNN CNN with maxout function.
Cakir et al. (2015a)	• Single label classifier may need to be tuned separately to achieve maximal performance. • The testing time may be a few folds higher than a multi-label classifier. • Maxout function doubles the number of parameters for every single unit. • Maxout network is prone to overfitting. • FNN with maxout function may not perform as well as a CNN CNN with maxout function.
Parascandolo et al. (2016)	• Data augmentation techniques is not useful. • Lack of post processing which may results in a lower event-based evaluation metric. • Use of Gaussian noise as optimization can increase training time and decrease the accuracy. • LSTM has high model complexity and is not easy to tune. • Sequential nature of LSTM also inhibits parallelization.
Adavanne et al. (2016)	• Additional audio features provide little performance improvement. • LSTM has high model complexity and is not easy to tune. • Sequential nature of LSTM also inhibits parallelization.
Xia et al. (2018)	• Simple architecture may be the cause of low accuracy. • CNN lacks long temporal context information. • May require careful tuning of hyperparameters for the parabolic function. • Using a parabola as a confidence function may not be appropriate.
Vesperini et al. (2019)	• Unclear how well this classifier performs in terms of F1-score. • CapsNet requires significant computational resources. • Training of CapsNet is longer, can be more prone to large errors and also lower generalization ability as compared to CNNs. • Hyperparameters must be tuned correctly for maximal performance. • AdaDelta (Zeiler, 2012) has higher computational time as compared to Adam (Kingma and Ba, 2015)

Table 2.14 Limitations for different non-hybrid NN-based methodologies

gated recurrent unit (GRU) was proposed to make each recurrent unit capture dependencies of different time scales adaptively (Cho et al., 2014). It is similar to an LSTM, such that GRU also has gating units that modulate the flow of information inside the unit, but the difference with an LSTM is that GRU does not have separate memory cells (Chung et al., 2014). Such implementation results in a simpler model with much lesser parameters, and as evaluated by Cakir et al. (2017), performance between the two models was comparable in their application.

Based on the use of such a combination, Cakir et al. (2017) reported a single-frame segment-based F1-score of 69.7% with 0.45 single-frame ER and single-second segment-based F1-score of 69.3% with 0.48 single-frame ER on the TUT-SED 2009 dataset, which outperforms CNN, RNN, GMM, and FNN.

However, such architecture was not the best classifier when tested on TUT-SED 2016 dataset in terms of F1-score even though the architecture had the lowest single-frame and single-second ER. On the TUT-SED 2016 dataset, Cakir et al. (2017) reported a single-frame segment-based F1-score of 27.5% (lower than RNN with an F1-score of 27.6%) and a single-second segment-based F1-score of 30.3% (lower than GMM with an F1-score of 32.5%). This architecture was also not the best architecture when tested on the CHIME-Home evaluation dataset. CNN achieved the lowest Equal Error Rate (EER) of 10.7, while CRNN achieves an EER of 11.3.

On the other hand, Jung et al. (2019) proposed using BLSTM to stack with CNN. The architecture and training scheme remains largely similar as compared to (Cakir et al., 2017). However, the implementation by Jung et al. (2019) had more recurrent layers and utilized LN instead of BN on the recurrent layer. Since BLSTM produces two outputs that differ due to the input order, the outputs were concatenated and used as input to the subsequent layer. Jung et al. (2019) then proposed transfer learning in an attempt to increase the training accuracy. In their framework, transfer learning was carried out by first training the Convolutional Bidirectional LSTM (CBLSTM) using a set of synthetic data before transferring the adjusted weights to a new CBLSTM, which will be trained using the set of real training data.

Based on such implementation, Jung et al. (2019) achieved a single-frame segment-based F1-score of 49.9%, which was much higher than Cakir et al. (2017). Jung et al. (2019) reported an even higher single-frame segment-based F1-score of 55.9% with the application of transfer learning. Their system also had a lower single frame ER of 0.56 as compared to Cakir et al. (2017), which had a single frame ER of 0.98. However, due to the application of transfer learning, there is a need to generate synthetic data that requires

additional effort. Moreover, as mentioned earlier, LSTM has high model complexity and may be difficult to tune.

Adavanne et al. (2017) extended their work in (Adavanne et al., 2016) and proposed a CBLSTM with spatial features extracted from the different channels. In their proposal, input features comprised log mel energies, TDOA estimated using generalized cross-correlation with phase-based weighting (GCC-PHAT), GCC-PHAT, dominant frequency (dom-freq) and their amplitudes, and Autocorrelation (ACR), which was used to estimate pitch. Each feature from each channel was stacked over the other to form a volume, and by slicing the volume within a specific time frame, all multichannel features corresponding to the time frame can be extracted. Since there was a difference in features dimension, Adavanne et al. (2017) proposed to use separate CNN to learn local shift-invariant features and concatenated the feature maps from each CNN before passing them to the BLSTM.

Based on different feature combinations, Adavanne et al. (2017) concluded that using log mel energies with dom-freq was the best combination for TUT-SED 2009, which achieved a single-second segment-based F1-score of 71.7% with a single-second ER of 0.43. The best feature combination for TUT-SED 2016 was log mel energies with TDOA, which achieved a single-second segment-based F1 score of 35.8% with a single-second ER of 0.95.

While results showed that features extracted from the stereo channel could be beneficial, results did not indicate the usefulness of adding additional features. In their paper, a classifier trained using only log mel energies can achieve a single-second F1-score of 71.1% with 0.43 ER on the TUT-SED 2009, which means that the best classifier only performed marginally better.

Whereas in their earlier work, Adavanne et al. (2016) demonstrated that an LSTM trained using only log mel energies could achieve a single-second segment-based F1-score of 35.6% with an ER of 0.93 on the TUT-SED 2016. Thus, the proposed CBLSTM with additional features only marginally improved the F1-score by 0.2 but at the expense of ER.

Adavanne et al. (2018) further extended their work using a 3D CNN and only using log mel energies with GCC-PHAT. As compared to the earlier work (Adavanne et al., 2017), the other key differences are 1) the first layer of CNN used to extract features from log mel energies, and GCC-PHAT was a 3D CNN, 2) the bidirectional LSTM was replaced with a bidirectional GRU, 3) early stop was based on accuracy improvement over 100 epochs instead of 50.

Adavanne et al. (2018) then tested the architecture using a different combination of features on a synthetic dataset, and the results reiterated that GCC-PHAT as an additional

input feature was ineffectual in improving the accuracy. With this conclusion, Adavanne et al. (2018) then tested the architecture on TUT-SED 2017. They reported a single-second F1-score of 67.5% with a single-second ER of 0.35. Such results outperformed a similar architecture with 2D CNN, which achieved a single-second F1-score of 64.8% with a single-second ER of 0.37.

Xia et al. (2019) also extended his work (Xia et al., 2018) by introducing Auxiliary Classifier Generative Adversarial Network (AC-GAN) to balance the dataset between event classes and replaced the use of CNN with CRNN. In their framework, AC-GAN was utilized to generate virtual sound samples that are close to the real sound samples by considering additional conditions such as the sound event class information and the sound event localization information. Note that localization in this context refers to the location of the activated frames in the audio (i.e., temporal localization); it does not refer to the location of the sound source (i.e., spatial localization). Thus, generated samples would be more convincing than using simple data augmentation methods proposed in (Parascandolo et al., 2016), which were shown to produce limited success. Generated samples were then used together with the real samples to train a CRNN. The overall framework is illustrated in Figure 2.11.

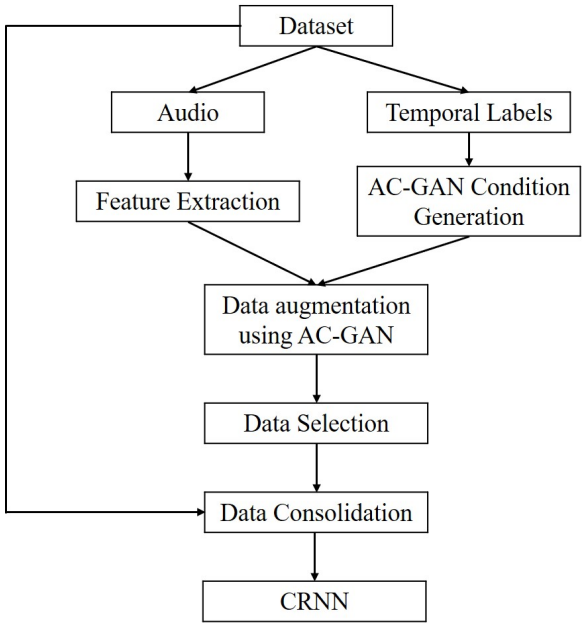


Figure 2.11 Proposed framework by Xia et al. (2019)

Based on such a proposal, Xia et al. (2019) reported a single-second segment-based F1-score of 31.1% with an ER of 0.58 on the TUT-SED 2016 development dataset (Home) but only achieved a single-second segment-based F1-score of 19.6% with an ER of 0.84 on the evaluation dataset. As results were only reported on the part of the dataset, it is

unclear how well it performs on the entire dataset and thus cannot be compared with other methodologies that tested their methods over the whole dataset.

When tested on TUT-SED 2017 development dataset, Xia et al. (2019) reported a single-second segment-based F1-score of 52.7% with an ER of 0.34. On the other hand, Xia et al. (2019) reported a single-second segment-based F1-score of 48.3% with an ER of 0.59 on the evaluation dataset. Although based on the evaluation results, Xia et al. (2019) can easily clinch the top spot in the DCASE 2017 challenge task 3, it had a 15% gap compared to (Adavanne et al., 2018) on the development dataset.

In the experiment, AC-GAN was used to balance five minority classes for SED-2016, while AC-GAN was used to balance only three minority classes for SED-2017. As seen from the results, the accuracy of such methodology appears to have a significant fluctuation when tested on different datasets. It could be due to the fact that AC-GAN tends to generate near-identical samples for most classes as the number of labels increases (Gong et al., 2019). Secondly, AC-GAN imposes perfect separability, which is disadvantageous when the supports of the class distributions have significant overlap (Gong et al., 2019). As the method for generating soft labels is similar to their earlier work (Xia et al., 2018), the limitations associated with the earlier work remain the same.

Ding and He (2020) proposed an adaptive multi-scale detection method that combined the idea of an hourglass network (Newell et al., 2016) with a Bidirectional GRU (BGR).

An hourglass network comprises several convolutional and max-pooling layers to process features down to a very low resolution. The network branches off at each max pooling step and applies more convolutions at the original pre-pooled resolution. After reaching the lowest resolution, the network begins the top-down sequence of upsampling and a combination of features across scales (Newell et al., 2016).

Since this was a hybrid model, the combination of features at each scale was sent to a bidirectional GRU. Outputs from the GRUs were then upsampled and multiplied with a set of weights to balance the contribution of each branch. The resulting values were then added up and sent to the output layer.

In their study, Ding and He (2020) proposed using a 4 layers hourglass network with 3 layers of bidirectional GRU at each scale. Based on such architecture, Ding and He (2020) reported a single-second F1-score of 48.7% with an ER of 0.78 on the TUT-SED 2016 evaluation dataset and a single-second F1-score of 43.6% with an ER of 0.77 on the TUT-SED 2017 evaluation dataset.

However, such results cannot outperform a conventional CRNN with data augmentation using Generative Adversarial Network (GAN) (Xia et al., 2019). Moreover, it requires

much higher computational resources than a CRNN due to the combination of features at different scales and may also overfit easily with a small number of samples.

2.2.4. Summary of Hybrid Models

In the previous section, different types of CRNN were reviewed in detail and discussed. Although there are different ways to construct a CRNN, the main difference lies in the choice of RNN. Although GRU and LSTM may be on par in terms of accuracy, GRU has much lesser parameters than an LSTM.

It should also be noted that although CRNN can further increase the detection accuracy, it still requires a large amount of strongly labeled training data to learn the mapping between features and event class. Moreover, stacking CNN with RNN will increase the computational complexity by several folds, resulting in a much longer training time. In addition, such architecture does not allow parallel computing due to its sequential nature (Kong et al., 2020). Lastly, CRNN also requires class-specific thresholds to determine the activation of sound events, which can affect the detection accuracy due to suboptimal tuning.

In this section, features used by different authors are presented in Table 2.15 to Table 2.17. Similarly, mel energies remain the most popular feature. The details of their architecture can be seen in Table 2.18 to Table 2.25. As an author can propose a different architecture based on different datasets or features, we split the architecture according to authors for better clarity.

The Filters and Units in Table 2.18 to Table 2.22 represent the number of filters or units in each layer. Example: 256, 256, 256 in the Filter column refers to 256 filters in the first layer, 256 filters in the second layer, and 256 filters in the third layer. Whereas the Pooling Size in Table 2.18 to Table 2.22 indicates the pooling size at each layer. Thus, a pooling size of (1, 5), (1, 4), (1, 2) represent that the Time-Frequency (TF) representation only has its frequency dimension reduced by 5 times in the first layer, further reduced by another 4 times in the second layer and another 2 times in the last layer. Using a 40 mel bands TF input as an example, the 40 bands become 1 band in 3 stages: $40 \rightarrow 8 \rightarrow 2 \rightarrow 1$.

Finally, results and limitations of all the CRNN architectures are presented in Table 2.26 and Table 2.27, respectively.

References	Features
Cakir et al. (2017)	Log mel energies
Jung et al. (2019)	Mel energies
Adavanne et al. (2017)	Log mel energies TDoA GCC-PHAT dom-freq ACR
Adavanne et al. (2018)	Log mel energies GCC-PHAT
Xia et al. (2019)	Log mel energies
Ding and He (2020)	Mel energies

Table 2.15 Features proposed for hybrid NN-based methodologies

References	Window	Window length (ms)	Overlap (%)	Mel-filterbanks
Cakir et al. (2017)	Hanning	40	50	40
Jung et al. (2019)	-	50	50	40
Adavanne et al. (2017)	Hamming	40	-	40
Adavanne et al. (2018)		40	50	40
Xia et al. (2019)	-	40	50	40
Ding and He (2020)		40	50	128

Table 2.16 Parameters used for feature calculation (hybrid NN-based methodologies)

References	Additional processing steps
Cakir et al. (2017)	• Z-score normalization of energy band.
Jung et al. (2019)	• Clip below -100db and min-max normalization of energy band.
Adavanne et al. (2017)	• dom-freq picked from 100 to 4000Hz. • ACR calculated in 40ms window and in the range of 107.5Hz to 4410 Hz.
Adavanne et al. (2018)	• GCC extracted in 120, 240, 480 ms window.
Xia et al. (2019)	• Data augmentation using AC-GAN.
Ding and He (2020)	• Z-score normalization of energy band.

Table 2.17 Additional processing steps (hybrid NN-based methodologies)

Model based on	CNN details	RNN details
TUT-SED2009	• Layers: 3 • Filters: 256, 256, 256 • Kernel size: (5, 5) • Pooling operator: Max • Pooling size: (1, 5), (1, 4), (1, 2) • Activation function: ReLU	• Type: GRU • Layer: 1 • Units: 256
TUT-SED 2016	• Layer: 3 • Filters: 256, 256, 256 • Kernel size: (5, 5) • Pooling operator: Max • Pooling size: (1, 2) , (1, 2), (1, 2) • Activation function: ReLU	• Type: GRU • Layers: 3 • Units: 96, 96, 96
CHIME-Home	• Layer: 4 • Filters: 256, 256, 256, 256 • Kernel size: (5, 5) • Pooling operator: Max • Pooling size: (1, 2), (1, 2), (1, 2), (1, 1) • Activation function: ReLU	• Type: GRU • Layer: 1 • Units: 256

Table 2.18 Hybrid NN architecture by Cakir et al. (2017)

CNN details	RNN details
• Layers: 3 • Filters: 256, 256, 256 • Kernel size: 3 • Pooling operator: Max • Pooling size: (1, 5), (1, 4), (1, 2) • Activation function: Sigmoid	• Type: BLSTM • Layers: 3 • Units: 100, 100, 100 • Activation function: Sigmoid

Table 2.19 Hybrid NN architecture by Jung et al. (2019)

Model based on	CNN details	RNN details
Log mel energies	• Layers: 3 • Filters: 100, 100, 100 • Kernel size: (3, 3) • Pooling operator: Max • Pooling size: (1, 2), (1, 2), (1, 2) • Activation function: ReLU	• Type: BLSTM • Layers: 2 • Units: 100, 100
GCC-PHAT	• Layers: 3 • Filters: 100, 100, 100 • Kernel size: (3, 3) • Pooling operator: Max • Pooling size: (1, 3), (1, 2), (1, 2) • Activation function: ReLU	
ACR	• Layers: 3 • Filters: 100, 100, 100 • Kernel size: (3, 3) • Pooling operator: Max • Pooling size: (1, 10), (1, 4), (1, 2) • Activation function: ReLU	
TDoA and dom-freq	• Layer: 1 • Filters: 100 • Kernel size: (3, 3) • Activation function: ReLU	

Table 2.20 Hybrid NN architecture by Adavanne et al. (2017)

Model based on	CNN details	RNN details
Log mel energies	• Layers: 3 • Filters: 64, 64, 64 • Kernel size: (3, 3)* • Pooling operator: Max • Pooling size: (1, 5), (1, 2), (1, 2) • Activation function: ReLU	• Type: GRU • Layers: 2 • Units: 64, 64 • Activation function: Hyperbolic Tangent
GCC-PHAT	• Layers: 3 • Filters: 64, 64, 64 • Kernel size: (3, 3)* • Pooling operator: Max • Pooling size: (1, 5), (1, 3), (1, 2) • Activation function: ReLU	

Table 2.21 Hybrid NN architecture by Adavanne et al. (2018)
*First layer is a 3D CNN

CNN details	RNN details
• Layers: 3 • Filters: 128, 128, 128 • Kernel size: (3, 3) • Pooling operator: Max • Pooling size: (1, 5), (1, 2) , (1, 2) • Activation function: ReLu	• Type: GRU • Layers: 2 • Units: 32, 32

Table 2.22 Hybrid NN architecture by Xia et al. (2019)

Hourglass network details	RNN details
• Layers: 4 • Activation function: ReLu	• Type: GRU • Layers: 3

Table 2.23 Hybrid NN architecture by Ding and He (2020)

References	Loss function	Optimizer
Cakir et al. (2017)	BCE	Adam (Kingma and Ba, 2015)
Jung et al. (2019)	BCE	Adam (Kingma and Ba, 2015)
Adavanne et al. (2017)	BCE	Adam (Kingma and Ba, 2015)
Adavanne et al. (2018)	BCE	Adam (Kingma and Ba, 2015)
Xia et al. (2019)	BCE	Adam (Kingma and Ba, 2015)
Ding and He (2020)	Weighted Average BCE	Adam (Kingma and Ba, 2015)

Table 2.24 Loss functions and optimizers used for training hybrid NN-based architecture

References	Additional information
Cakir et al. (2017)	• Dropout rate: 0.25 • Early stop criterion: 100 • BN after each convolutional layer or FC layer • Pooling applied after every convolution layer • Activation function for output layer: Sigmoid
Jung et al. (2019)	• Learning rate: 0.001 • Decay rate: Exponential • Dropout rate: 0.3 • BN for each convolution layer. • Pooling applied after every convolution layer • LN for each BLSTM layer • Activation function for output layer: Sigmoid
Adavanne et al. (2017)	• Mini-batch: 32 • Dropout rate: 0.5 • BN for each convolution layer • Early stop criterion: 50 epochs • Activation function for output layer: Sigmoid
Adavanne et al. (2018)	• Learning rate: 0.0001 • Mini-batch: 128 • Dropout rate: 0.2 • BN for each convolution layer • Pooling applied after every convolution layer • Early stop criterion: 100 epochs • Activation function for output layer: Sigmoid
Xia et al. (2018)	• Learning rate: 0.001 • BN for each convolution layer • Pooling applied after every convolution layer • Activation function for output layer: Sigmoid
Ding and He (2020)	• Learning rate: 0.001 • Mini-batch: 45 • BN for each convolution layer • Pooling applied after every convolution layer • Activation function for output layer: Sigmoid

Table 2.25 Additional models information (hybrid NN-based architecture)

References	Dataset	Segment-based F1-score (%)	Segment-based ER	Other metric
Cakir et al. (2017)	TUT-SED 2009	1-Frame: 69.7	1-Frame: 0.45	
		1-Frame: 69.3	1-Frame: 0.48	
	TUT-SED 2016 (Dev)	1-Frame: 27.6	1-Frame: 0.98	
		1-Frame: 30.3	1-Frame: 0.95	
	CHiME-Home (Dev)			EER: 13
	CHiME-Home (Eva)			EER: 11.3
Jung et al. (2019)	TUT-SED 2016 (Dev)	1-Frame: 55.9	1-Frame: 0.56	
Adavanne et al. (2017)	TUT-SED 2009	1-Frame: 71.7	1-Frame: 0.43	
	TUT-SED 2016 (Dev)	1-Frame: 35.8	1-Frame: 0.95	
Adavanne et al. (2018)	TUT-SED 2017 (Dev)	1-Frame: 67.5	1-Frame: 0.35	
Xia et al. (2019)	TUT-SED 2016-Home (Dev)	1-Frame: 37.1	1-Frame: 0.58	
	TUT-SED 2016-Home (Eva)	1-Frame: 19.6	1-Frame: 0.84	
	TUT-SED 2017 (Dev)	1-Frame: 52.7	1-Frame: 0.34	
	TUT-SED 2017 (Eva)	1-Frame: 48.3	1-Frame: 0.59	
Ding and He (2020)	TUT-SED 2016 (Eva)	1-Frame: 48.7	1-Frame: 0.78	
	TUT-SED 2017 (Eva)	1-Frame: 43.6	1-Frame: 0.77	

Table 2.26 Dataset used by different authors and reported accuracy using hybrid NN-based methodologies. Dev refers to development dataset. Eva refers to evaluation dataset.

References	Limitations
Cakir et al. (2017)	• Accuracy was not the best across different dataset.
Jung et al. (2019)	• Require additional effort to generate synthetic data for transfer learning. • LSTM has high model complexity and is not easy to tune.
Adavanne et al. (2017)	• Additional audio features provide little performance improvement. • LSTM has high model complexity and is not easy to tune.
Adavanne et al. (2018)	• Additional features did not help to increase accuracy.
Xia et al. (2019)	• May require careful tuning of hyperparameters for the parabolic function. • Using a parabola as a confidence function may not be appropriate. • AC-GAN produces sample with less diversity as classes increases. • AC-GAN imposes perfect separability which is disadvantageous.
Ding and He (2020)	• May requires much higher computational resource as compared to a CRNN. • Such network can easily overfit with a small amount of samples. • Cannot perform as well as a CRNN with data augmentation using GAN.

Table 2.27 Limitations for different hybrid NN-based methodologies

2.2.5. Models Utilizing Weakly Labeled Data

Due to the lack of strongly labeled data, research efforts then focus on the effective use of weakly labeled data. Lee et al. (2017) proposed an ensemble of CNNs for SED utilizing only weakly labeled data. The idea was to have two different CNN architectures trained with different inputs. In their framework, a Global Input Model utilized the entire TF representation as the training input and a Separated Input Model that took a smaller segment of TF representation as training input. These smaller segments are broken up from the original TF representation using a non-overlapping sliding window with a window size of 1 to 5 seconds and were considered to have the same label as the original TF representation. The final prediction was then given by combining the models' probabilities. The overall framework is illustrated in Figure 2.12.

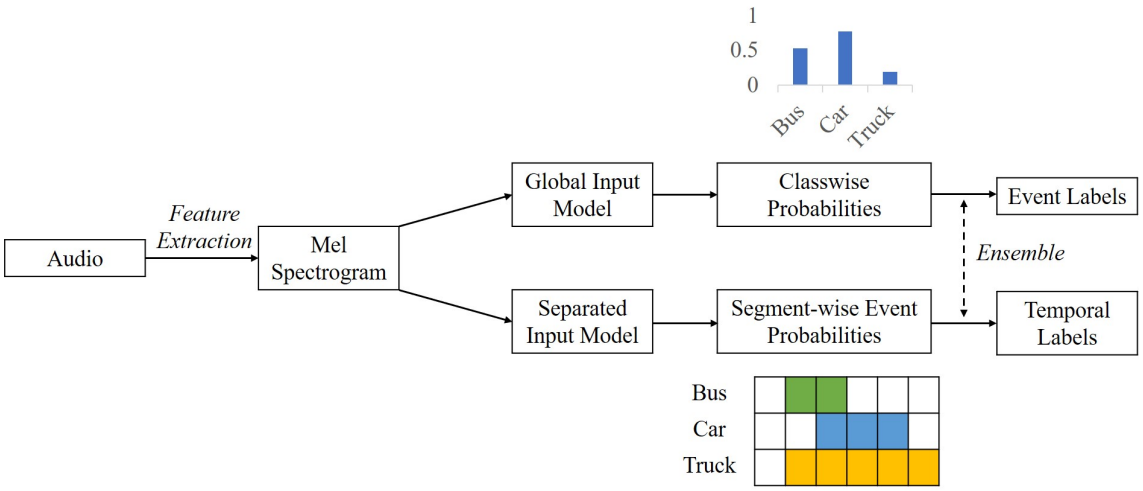


Figure 2.12 Proposed framework by Lee et al. (2017)

Lee et al. (2017) then proposed an iterative approach to form the ensemble based on accuracy measures. Thus, a model will only be included in the ensemble if it can raise the accuracy of the ensemble. An ensemble was formed from 12 models trained using the Global Input Model and Separated Input Model based on such an approach.

Using the ensemble, Lee et al. (2017) reported an F1-score of 52.1% with ER of 0.66 on the DCASE 2017challenge task 4 challenge development dataset and an F1-score of 55.5% with an ER of 0.66 on the evaluation dataset. Based on such results, Lee et al. (2017) were able to clinch first place in the challenge.

A significant drawback of such a methodology is the computational resource and time required to train many models and form an ensemble using the iterative selection approach. Subsequently, to eliminate the background noise from the audio, Lee et al. (2017) proposed a background subtraction method, which was carried out by subtracting the median from each mel band. However, this method can significantly degrade model performance that used the entire or half the TF representation. In addition, treating the smaller segments to have the same label as the original TF representation can induce noise to the training sample because the smaller samples may not contain any information regarding the label. The idea of disregarding the label position was also tested in (Chan et al., 2019), where results have shown that it can induce noise to the training examples, resulting in a lower segment-based F1-score.

Xu et al. (2018) proposed a CRNN with Gated Linear Unit (GLU) (Dauphin et al., 2018) and a temporal pooling approach known as attention pooling. Such a system was a joint system that produced the audio tags and the corresponding temporal labels. Such framework is illustrated in Figure 2.13.

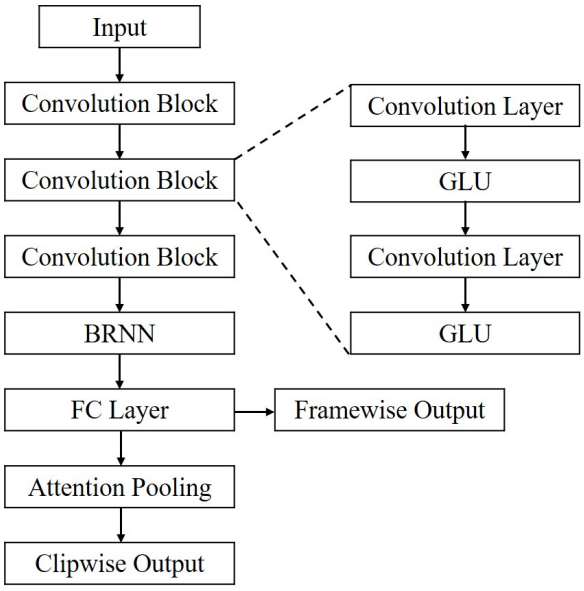


Figure 2.13 Proposed framework by Xu et al. (2018)

The GLU activation function can be regarded as a local attention scheme and is similar to other gating mechanisms in LSTM or GRU, which control the information flow to the next layer and is defined as (Dauphin et al., 2018)

$$\mathbf{Y} = (\mathbf{x}\mathbf{W}_1 + \mathbf{b}_1) \otimes \sigma(\mathbf{x}\mathbf{W}_2 + \mathbf{b}_2) \tag{2.6}$$

where $\mathbf{x}$ is the input and σ is the sigmoidal function. $\otimes$ is the hadamard product. $\mathbf{W}_1$, $\mathbf{W}_2$, $\mathbf{b}_1$ and $\mathbf{b}_2$ are learnable parameters (i.e., weights and bias). The benefit of GLU is that it reduces the gradient vanishing problem for the deep network by allowing a linear path for the gradients while keeping non-linear capabilities through the sigmoid operation (Dauphin et al., 2018).

Attention pooling is a temporal pooling approach that makes use of two FNNs with different activation functions. The first activation function in the FNN was the softmax activation function which was used to infer the temporal locations for each occurring class and attended to the most salient frames for each class. The second activation function was the sigmoid activation function which performed classification at each frame. The audio tags were then obtained through a two-step approach. The first step is to perform the sum of product between sigmoid and softmax activation outputs along the temporal axis. The second step is the division of the value obtained in step 1 by the sum of the output from the softmax activation along the temporal axis.

In order to alleviate the issue of an imbalanced dataset, Xu et al. (2018) proposed including samples from all classes in every training batch. The selection of samples would

follow the similar distribution ratio of each class but ensure that there is at least one sample from the minority classes. In contrast, the majority class would be restricted to be at most five times more than the minority class.

Xu et al. (2018) proposed fusing results generated during each training epoch to improve the detection accuracy. They also proposed an ensemble system where one system was trained using MFCCs while the other was trained using log mel energies. Outputs from both systems will then be averaged to retrieved the final posterior, which was then thresholded by an array of class-dependent thresholds to determine the occurrence of the sound event. Based on such implementation, Xu et al. (2018) achieved a single-second segment-based F1-score of 49.7% with ER of 0.72 on the DCASE 2017 challenge task 4 development dataset and a single-second segment-based F1-score of 51.8% with an ER of 0.73 on the evaluation dataset.

While such a system was among the top three implementations in the challenge, it was also found to perform rather poorly on event-based evaluations for a similar task in DCASE 2018 (Kothinti et al., 2019). In addition, experiments also found that such architecture cannot be simplified and will result in poor detection accuracy (Pellegrini and Cances, 2019). The use of GLU in their methodology can also increase the total number of parameters to be learned. Finally, it was also found that attention pooling may not be the best temporal pooling option for audio tags prediction (Wang et al., 2019a).

Kong et al. (2020) proposed a CNN-Transformer (CNNT) for SED. The motivation of their work was to reduce the training time of a CRNN by making use of a transformer to replace the RNN. As a CRNN has to be calculated sequentially, this can make it quite challenging to complete the training process in a short amount of time (Kong et al., 2020). On the other hand, the Transformer can take a long-time dependency into consideration for a system but comes with the benefit of allowing parallel computing, thus, making it an excellent alternative to RNN (Kong et al., 2020).

In their framework, Kong et al. (2020) made use of 3 different sets of thresholds. The first set would allow the audio to be tagged. The second set was a set of upper bound thresholds that would indicate the frames containing the respective sound event. However, this may result in several false negatives. Thus, the third set of the lower bound threshold was utilized to determine if the neighboring frames contain the same sound event. As there were many thresholds, manual tuning of these values can be complicated and inefficient. Thus, Kong et al. (2020) proposed an automatic threshold tuning method. Such a method would allow the thresholds to be tuned using backpropagation.

Based on such implementation, Kong et al. (2020) reported a segment-based F1-score of 52.4% with ER of 0.75 on the development dataset and a segment-based F1-score of 57.3% with an ER of 0.75 on the evaluation dataset.

However, based on both DCASE 2017 development and evaluation results, the CNNT could not outperform a CRNN with the attention layer. Although Kong et al. (2020) only applied the encoding layer of a Transformer, it should be pointed out that the overall training process can be slowed down if the decoding layer is used (Karita et al., 2019; Zhang et al., 2018a). Finally, as analyzed by Pellegrini and Masquelier (2021), the threshold optimizer proposed by Kong et al. (2020) may suffer from convergence issues because hyperparameters such as learning rate may not be optimal for each target class individually. It is also inefficient when dealing with a large number of classes and training examples.

Lu (2018) then proposed a modified version of the Xu et al. (2018) system. To lessen the number of parameters due to the use of GLU (Dauphin et al., 2018), Lu (2018) proposed using Context Gating (CG) (Miech et al., 2018) as the activation function for CNN, which can be defined as

$$\mathbf{Y} = \mathbf{x} \otimes \sigma(\mathbf{x}\mathbf{W}_1 + \mathbf{b}_1) \tag{2.7}$$

By comparing with Equation 2.6, CG only has one set of learnable parameters and thus has the benefit of improved efficiency compared to GLU. At the same time, it also allowing non-linear interactions among activations of the input representation (Miech et al., 2018).

Besides training the system with only weakly labeled data, Lu (2018) also utilized a large set of unlabeled data. Such an approach is known as semi-supervised learning, and Lu (2018) proposed using the Mean-Teacher semi-supervised learning scheme (Tarvainen and Valpola, 2017) in their framework.

The idea of the Mean-Teacher approach is to train two identical models (i.e., Teacher Model (TM) and Student Model (SM)) synchronously. Such a framework has two primary loss functions; a multi-class (or multi-label) loss function and a consistency loss function. The first loss function is to update the models according to the training input, whereas the second loss function is to enforce similar predictions between SM and TM (i.e., SM predictions should be close to TM predictions). In the training procedure, the weights of the SM are updated through gradient descent. In contrast, the weights of the TM are updated as an exponential moving average of the SM weights. Such a training scheme is based on the fact that using average model weights over training epochs instead of using the final weights directly can produce a more accurate model (Tarvainen and Valpola, 2017). An example of the Mean-Teacher approach is illustrated in Figure 1.3.

Similar to (Xu et al., 2018), Lu (2018) also proposed an ensembled system to produce a more accurate model, in which Lu (2018) proposed to combine the results from different systems using different consistency costs for the Mean Teacher model.

Based on this system, Lu (2018) reported an event-based F1-score of 34.4% with an ER of 1.16 on the development dataset and an event-based F1-score of 32.4% on the evaluation dataset, which allows Lu (2018) to clinch first place in the 2018 DCASE Task 4 challenge.

However, the subsequent analysis found that such a system performed poorly for sound events of short duration (Serizel and Turpault, 2019). Moreover, the major drawback of this model is the large number of models to be trained to form the ensemble. Due to the nature of the student-teacher model, each model with a different consistency cost requires two models (i.e., SM and TM) to be trained synchronously; this can result in a significant increase in computational cost and resource burdens. The use of attention pooling for audio tagging may also not be optimal (Wang et al., 2019a). Finally, consistency cost requires careful tuning to achieve maximal performance. Since Lu (2018) proposed an ensemble approach, the impact of consistency cost was alleviated.

Lin et al. (2020) also proposed a teacher-student frame for SED. However, instead of training two identical models synchronously, Lin et al. (2020) proposed training two different models synchronously. The TM was a model with higher model complexity and a larger pooling size along the temporal axis after each convolutional layer. It was hypothesized that such a model could integrate the audio contextual information and, thus, capable of producing a better clip level prediction. In contrast, the SM was a simpler model with no pooling along the temporal axis and was hypothesized to be better at frame-level prediction. The proposed teacher and student model can be seen in Figure 2.14. The entire training procedure can be broken down into two different stages where the transition from stage one to stage two is controlled by the training epochs defined by the user. The first stage can be considered the student learning stage, while the second stage can be considered the mutual learning stage. In the first stage, both TM and SM model parameters are updated based on a combined cost consisting of three components. The first two components were the classification cost of TM and SM on the labeled data, which can be given as

$$C_1 = J(\ddot{\mathbf{T}}_k, \mathbf{Z}_k) \tag{2.8}$$

$$C_2 = J(\ddot{\mathbf{S}}_k, \mathbf{Z}_k) \tag{2.9}$$

where $J(.)$ denotes the cost function and $\mathbf{Z}_k$ as the vector representing the clip-level ground truth of the k sample of the weakly labeled dataset. $\ddot{\mathbf{T}}_k$ and $\ddot{\mathbf{S}}_k$ represent the vectors

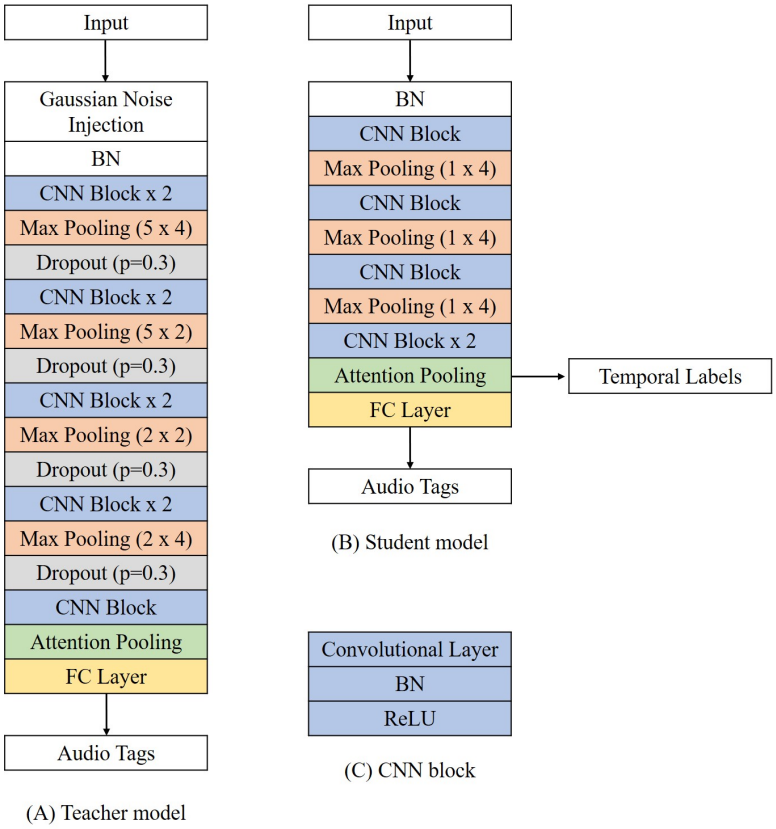


Figure 2.14 Proposed framework by Lin et al. (2020)

containing the probabilities of each event that occurred in weakly labeled sample k by the TM and SM, respectively. The third component was the classification cost of TM and SM on the unlabeled data, which can be defined as

$$C_3 = J(\ddot{\mathbf{S}}_k^{\mathrm{u}}, \ddot{\mathbf{T}}_k^{\mathrm{u}}) \tag{2.10}$$

where $\ddot{\mathbf{S}}_k^{\mathrm{u}}$ and $\ddot{\mathbf{T}}_k^{\mathrm{u}}$ represent the SM and TM prediction of the k sample of the unlabeled dataset. Thus, in the first stage, the learning procedure effectively forces the SM to learn from the TM. After a specific training epoch, the training procedure transits into the second stage, where the combined cost was added with another component, C_4, which can be defined as

$$C_4 = \varepsilon \times J(\ddot{\mathbf{T}}_k^{\mathrm{u}}, \ddot{\mathbf{S}}_k^{\mathrm{u}}) \tag{2.11}$$

where ε determines how much the TM should learn from the SM. Thus, such a learning scheme will allow the two CNN models to be trained simultaneously and forcing them to learn from each other, which Lin et al. (2020) hypothesized can increase the performance of the two models. As such, Lin et al. (2020) methodology can also be thought of as a variant of the distillation approach (Hinton et al., 2015) or the mutual learning (Zhang et al., 2018b) framework.

Based on such architecture, Lin et al. (2020) reported a higher event-based F1 score of 39.5% than (Lu, 2018) on the DCASE 2018 evaluation dataset.

Lin et al. (2019) then extended their work by creating a set of disentangled features that would mitigate the effects of overlapping sound events. Lin et al. (2019) then proposed using an adaptive median filter to smooth the SM's temporal outputs. The idea of an adaptive median filter is to smooth predicted temporal labels for an event class with an event-specific window which is determined through the average duration of event class in the synthetic dataset.

Finally, an ensemble of models with different ε was trained to produce the audio prediction. Based on these improvements, Lin et al. (2019) reported an event-based F1-score of 45.4% on the development dataset and an event-based F1-score of 42.7% on the evaluation dataset, which won first place in the DCASE 2019 task 4 challenge.

However, the major drawback of Lin et al. (2020) and Lin et al. (2019) is similar to Lu (2018); two models (i.e., SM and TM) have to be trained synchronously, which can result in a significant increase of computational cost and resource burdens. The use of ensemble in Lin et al. (2019) can further aggravate this problem.

Moreover, as seen in Figure 2.14, the two models are designed differently with no specific guidelines. This can result in a significant model tuning time. Besides, there are also several hyperparameters to be tuned. Firstly, how can the training epochs be determined effectively to allow proper transition from stage one to stage two? Secondly, how much should the TM learn from the SM? These two issues can only be answered through manual tuning, which further increases the model tuning time. Finally, the proposed adaptive median window size for each event may also not be optimal since it was derived based on the average event duration in the synthetic dataset.

Kothinti et al. (2019) also proposed two different setups for audio tagging and temporalization, which are illustrated in Figure 2.15. The temporal localization model was a combination of Restricted Boltzmann Machine (RBM), conditional RBM (cRBM), and Principle Component Analysis (PCA). In contrast, the audio tagging model was an ensemble of CRNNs. For the temporal localization model, a spectrogram was used as the input for the RBM to capture local-spectrotemporal dependencies. Outputs were subsequently passed to an array of cRBMs to generate the final high-dimensional representation of the acoustic signal. The dimensions of the signal representation were then reduced using PCA. The closest preceding sample at 25% of the maximal value in the reduced representation was then determined as the onset of an event. On the other hand, the offset was determined based on the threshold set on the short-term energies of the audio signal.

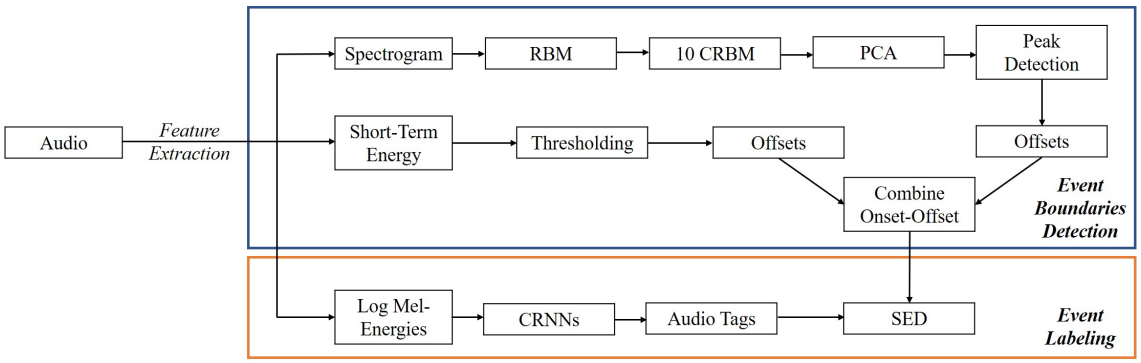


Figure 2.15 Proposed framework by Kothinti et al. (2019)

For the audio tagging model, log mel energies were used as the input for three different CRNNs. The first two CRNNs had the same architecture but were trained differently. The first system was trained using only weakly labeled data, while the second was trained using weakly labeled data and augmented data generated by mixing weakly labeled audios. The third system was the DCASE 2018 challenge task 4 baseline system trained using only weakly label audio. Besides the difference in training inputs used, the critical differences between the three systems were the number of filters and kernel size used in each convolutional layer. In their framework, the posterior probabilities from the three models were combined to give the final prediction through majority voting.

Kothinti et al. (2019) achieved an event-based F1-score of 30.05% with ER of 1.36 on the development dataset and an event-based F1-score of 25.4% with ER of 1.19 on the evaluation dataset.

Such results can only be considered mediocre despite such a complex design. In addition, it was found that the accuracy of the boundary detection system is lower for audio with overlapping events. The impact was only mitigated due to the error tolerance used for evaluation. Furthermore, it was found that the audio tagging system was poor at event labelings which deteriorate the overall performance significantly. Moreover, there also lies a possibility that the accuracy of boundary detection is affected by suboptimal thresholds.

In the framework, Contrastive Divergence (CD) is used to train a cRBM, but it was mentioned in (Mnih et al., 2011) that it might not be an excellent choice to use CD to train a cRBM for prediction purposes. It is unclear if it will also affect the representation learning, which in turn affects the boundary detection accuracy. However, theoretical analysis of CD is difficult (Carreira-Perpinan and Hinton, 2005), which can prohibit further investigation.

Pellegrini and Cances (2019) also proposed different model setups for SED, but the critical difference with the earlier proposals (Kothinti et al., 2019; Lin et al., 2019, 2020) is that the two setups shared the same feature extraction block. The overall framework is

illustrated in Figure 2.16. For the audio tagging model, the last feature extraction block

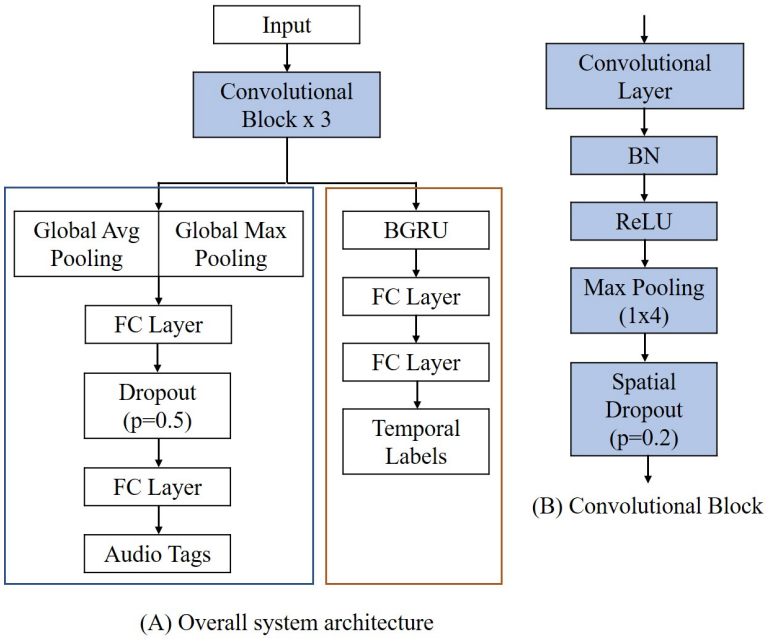


Figure 2.16 Proposed framework by Pellegrini and Cances (2019)

was followed by both global average pooling and global max pooling, then by a dense layer, and finally the output layer. The predictions from the audio tagging model were then thresholded by a set of class-dependent thresholds that were optimized by the Genetic Algorithm. For the temporal localization model, the last convolution block was followed by a BGRU, a dense layer, and finally, the output layer. Another distinct difference with the audio tagging model would be the cost function, where the temporal localization model took Cosine Similarity between classes into consideration, which was proposed to penalize overlapping events.

The output from this model was a set of temporal predictions for each class where Pellegrini and Cances (2019) proposed to discard all temporal predictions of classes detected as negative (i.e., event is predicted as not present) by the audio tagging model. The remaining temporal predictions were subsequently rescaled to 0 and 1 and smoothed with an average sliding window. Finally, the onsets and offsets of events were retrieved using a global threshold.

Based on such implementation, Pellegrini and Cances (2019) reported an event-based F1-score of 34.75% on the DCASE 2018 development dataset and an event-based F1-score of 26.2% on the evaluation dataset.

In their framework, several hyperparameters were coarsely determined. For example, the threshold used to detect the final onsets and offsets of events and the regularization weight used to determine the amount of contribution by Cosine Similarity to the loss

function. A coarsely determined threshold value can have an adverse effect on model accuracy. The overall accuracy is also dependent on the audio tagging model's accuracy. It was found that the temporal accuracy can have a 10% boost if all audio tags are correctly given.

Although Cosine Similarity was added to penalize overlapping events, it was not always helpful. In specific scenarios, it can help to decorrelate overlapping classes. However, in certain situations, it cannot provide decorrelation and can even cause one of the classes to be undetectable.

Finally, as GLU (Dauphin et al., 2018) was utilized, it can lead to a large parameter count due to a sophisticated model design.

2.2.6. Summary of Models Utilizing Weakly Labeled Data

In the previous section, different methodologies utilizing weakly labeled data with their limitations were discussed in detail. To leverage a large amount of weakly labeled data, most of the methodologies adopted the use of ensemble system or training of two different models where one is in charge of audio tagging while the other is in charge of boundary detection. However, this will require much more computational resources and time to train a different number of models. It is also important to point out that the model using weakly labeled data is not as effective as its counterpart using strongly labeled data. There are also studies that show audio tagging or classification accuracy may be reduced if trained with only weakly labeled data (Hershey et al., 2021; Turpault et al., 2020a). Thus, this remains an open research area as to how data can be utilized effectively. Finally, the use of threshold to determine the event activation remains a norm but Kong et al. (2020); Pellegrini and Cances (2019) presented some interesting ideas to tune it to an optimal value automatically and Kong et al. (2020) showed that accuracy did increase based on their optimization. However, there are still convergence issues associated with the tuning method proposed, and it may not work well with a large number of events classes or training samples (Pellegrini and Masquelier, 2021).

In this section, the features used by different authors are presented in Table 2.28 to Table 2.30. Similarly, mel energies remain the most popular as a feature and will most probably remain the gold standard for SED.

Table 2.31 to Table 2.39 that present the information of the architecture proposed. Due to significant differences in models' architecture, each model layout is presented in an individual table. Each column represents the different models used (either SM or TM or audio tagging model or event boundary model). In the case of (Kong et al., 2020; Xu

et al., 2018) , since they only utilized one type of hybrid architecture (i.e., CRNN and CNNT), the columns then showcase the details of the components used in their hybrid system. Since Lu (2018) used the mean-teacher training approach (Tarvainen and Valpola, 2017), both SM and TM are identical, so only one model is shown.

The table containing information on the Kothinti et al. (2019) system (which utilized 3 different CRNNs for audio tagging) is presented differently. The two systems that used a similar layout are classed together, so we only present one model architecture, and the information can be viewed under bullet point, Variant 1 and Variant 2. At the same time, the information on the third model can be viewed under bullet point, Variant 3.

Other naming convention, writing style (such as the number of filters, kernel size) remains the same as the previous section to ensure consistency.

Finally, results and limitations of all the CRNN architectures are presented in Table 2.40 and Table 2.41, respectively.

References	Features
Lee et al. (2017)	Log mel energies
Xu et al. (2018)	MFCC Log mel energies
Kong et al. (2020)	Log mel energies
Lu (2018)	Log mel energies
Lin et al. (2020)	Log mel energies
Lin et al. (2019)	Log mel energies
Kothinti et al. (2019)	Spectrogram Log mel energies
Pellegrini and Cances (2019)	Log mel energies

Table 2.28 Features proposed for methodologies utilizing weakly labled data

References	Window	Window length (ms)	Overlap (%)	Mel-filterbanks
Lee et al. (2017)	-	46	78	128
Xu et al. (2018)	Hamming	64	35	64
Kong et al. (2019b)	Hanning	32	69	64
Lu (2018)	-	93	83	128
Lin et al. (2020)	-	40	50	64
Lin et al. (2019)	-	40	50	64
Kothinti et al. (2019)	Unspecified for spectrogram	10	-	128*
Kothinti et al. (2019)	Hamming for log mel energies	40	50	64
Pellegrini and Cances (2019)	-	46	50	64

Table 2.29 Parameters used for feature calculation (models utilizing weakly labeled data). *Spectrogram channels

References	Additional processing steps
Lee et al. (2017)	• Audio was normalized prior to the calculation of mel energies. • All mel bands were then subtracted by the median value.
Xu et al. (2018)	-
Kong et al. (2020)	• Audio resampled to 32000Hz. • Frequency range to compute mel band is 50Hz to 14000Hz.
Lu (2018)	• Audio resampled to 22050Hz.
Lin et al. (2020)	• Normalization of energy bands.
Lin et al. (2019)	-
Kothinti et al. (2019)	-
Pellegrini and Cances (2019)	-

Table 2.30 Additional processing steps (models utilizing weakly labeled data)

Global input model	Separated input model
• Layers: 10 • Filters: 64, 64, 64, 64, 64, 64, 64, 64, 64, 64 • Kernel size: (3, 3) for all layers • Pooling Operator: Max • Pooling after every 2 convolution layers • Pooling size: (2, 4), (2, 4), (4, 4), (4, 4) • Activation function: ReLU	• Layers: 10 • Filters: 64, 64, 64, 64, 64, 64, 64, 64, 64, 64 • Kernel size: (3, 3) for all layers • Pooling Operator: Max • Pooling after every 2 convolution layers • Pooling size: (2, 2), (2, 2), (4, 3), (4, 4) • Activation function: ReLU

Table 2.31 Proposed architecture by Lee et al. (2017)

CNN details	RNN details
• Layers: 6 • Filters: 64 • Kernel size: (3, 3) for all layers • Pooling operator: Max • Pooling after every 2 convolution layers • Pooling size: (2, 2) through the whole model • Activation function: GLU	• Type: BGRU • Layer: 1 • Units: 128

Table 2.32 Proposed CRNN architecture by Xu et al. (2018)

CNN details	Transformer details
• Layers: 8 • Filters: 64, 64, 128, 128, 256, 256, 512, 512 • Kernel size: (3, 3) for all layers • Pooling operator: Average • Pooling after every 2 convolution layer • Pooling size: (4, 1) for the last layer, otherwise (2,2) • Activation function: ReLU	-

Table 2.33 Proposed CNNT architecture by Kong et al. (2020)

CNN details	RNN details
• Layers: 7 • Filters: 16, 32, 64, 128, 128, 128, 128 • Kernel size: (3, 3) for all layers • Pooling operator: Average • Pooling after every layer • Pooling size: (2, 2) for the first 2 layers, otherwise (1, 2)	• Type: BRNN • Layers: 1 • Units: 128

Table 2.34 Proposed CRNN architecture by Lu (2018). (Note: SM and TM are identical)

TM details	SM details
• Gaussian input noise: 0.15 • Layers: 9 • Filters: 16, 16, 32, 32, 64, 64, 12, 128, 256 • Kernel size: (1, 1) for the last layer otherwise (3, 3) • Pooling Operator: Max • Pooling after every 2 convolution layers • Pooling size: (5,4), (5,2), (2,2), (2,2) • Activation function: ReLU	• Layers: 3 • Filters: 160, 160, 160 • Kernel size: (3, 3) for the last layer otherwise (5, 5) • Pooling Operator: Max • Pooling after every convolution layer • Pooling size: (1,4), (1,4), (1,4) • Activation function: ReLU

Table 2.35 Proposed architecture by Lin et al. (2019, 2020)

Audio tagging model (CRNN) details		Event boundary model details
CNN details	RNN details	
• Variant 1 and Variant 2 – Layers: 3 – Filter: 128, 128, 192 – Kernel size: (1, 3) for all layers – Pooling after every layer – Pooling size: (1,8), (1, 4), (1,2) – Activation function: ReLU	• Variant 1 and Variant 2 – Layers: 1 – Units: 64	• RBM and cBRM • Units for RBM: 350 • Units for cRBM: 300
• Variant 3 – Layers: 3 – Filter: 64, 64, 64 – Kernel size: (3, 3) for all layers – Pooling operator: Max – Pooling after every layer – Pooling size: (1,4), (1, 4), (1,4) – Activation function: ReLU	• Variant 3 – Layers: 1 – Units: 64	

Table 2.36 Proposed architecture by Kothinti et al. (2019)

Shared block	Audio tagging model	Event boundary model
• Layers: 3 • Filters: 64 • Kernel size: (3, 3) for all layers • Pooling Operator: Max • Pooling applied after every convolution layer • Pooling size: (1, 4) for all layers • Activation function: GLU	• A global max and global avg pooling layer • FC units: 1024, 10	• RNN Type: BGRU – Layer: 1 – Units: 64 – Activation function: Hyperbolic Tangent • FC units: 64, 10

Table 2.37 Proposed architecture by Pellegrini and Cances (2019)

References	Loss function	Optimizer
Lee et al. (2017)	-	Adam (Kingma and Ba, 2015)
Xu et al. (2018)	BCE	Adam (Kingma and Ba, 2015)
Kong et al. (2020)	BCE	Adam (Kingma and Ba, 2015)
Lu (2018)	-	Adam (Kingma and Ba, 2015)
Lin et al. (2019, 2020)	BCE	Adam (Kingma and Ba, 2015)
Kothinti et al. (2019)	-	CD for RBM and cRBM
Pellegrini and Cances (2019)	BCE for CNN BCE with Cosine Similarity for CRNN	Adam (Kingma and Ba, 2015)

Table 2.38 Loss functions and optimizers used for models utilizing weakly labeled data

References	Additional Information
Lee et al. (2017)	• BN for each convolution layer • Global average pooling after the last convolution layer
Xu et al. (2018)	• Minibatch data balancing • Learning rate: 0.001 • Attention pooling
Kong et al. (2020)	• Learning rate: 0.001 • Mixup with alpha of 1 • BN after every convolution layer • Mel bins are averaged after the last convolution layer
Lu (2018)	• Dropout rate: 0.5 • Attention pooling
Lin et al. (2019, 2020)	• Learning rate: 0.0018 • Mini-batch: 64 • Decay rate: 0.8 • Dropout rate: 0.3 • BN for each convolution layer • Attention pooling • Early stop criterion: 20 epochs • Median filter applied on the output
Kothinti et al. (2019)	• PCA on the output of RBM+cRBM • Global average pooling after the last convolution layer for Variant 3 • Dropout rate: 0.3 for Variant 3 • Early stop criterion: 20 epochs for Variant 3 • Activation function for output layer: Sigmoid
Pellegrini and Cances (2019)	• Dropout rate: 0.2 for CNN layers, 0.5 for first dense layer in audio tagging model • BN after every convolution layer

Table 2.39 Additional models information (models utilizing weakly labeled data)

References	Dataset	Event-based F1-score (%)	Event based ER	Segment-based F1-score (%)	Segment-based ER
Lee et al. (2017)	DCASE 2017 Task 4 (Dev)	-	-	1-Frame: 52.1	1-Second: 0.66
	DCASE 2017 Task 4 (Eva)	-	-	1-Frame: 55.5	1-Second: 0.66
Xu et al. (2018)	DCASE 2017 Task 4 (Dev)	-	-	1-Frame: 49.7	1-Second: 0.72
	DCASE 2017 Task 4 (Eva)	-	-	1-Frame: 51.8	1-Second: 0.73
Kong et al. (2020)	DCASE 2017 Task 4 (Dev)	-	-	1-Frame: 52.4	1-Second: 0.75
	DCASE 2017 Task 4 (Eva)	-	-	1-Frame: 57.3	1-Second: 0.75
Lu (2018)	DCASE 2018 Task 4 (Dev)	34.4	1.12	-	-
	DCASE 2018 Task 4 (Eva)	32.4	-	-	-
Lin et al. (2020)	DCASE 2018 Task 4 (Eva)	39.5	-	-	-
Lin et al. (2019)	DCASE 2019 Task 4 (Dev)	45.4	-	1-Second: 69.02	-
	DCASE 2019 Task 4 (Eva)	42.7	-	1-Second: 71.4	-
Kothinti et al. (2019)	DCASE 2018 Task 4 (Dev)	30.1	1.36	-	-
	DCASE 2018 Task 4 (Eva)	25.4	1.19	-	-
Pellegrini and Cances (2019)	DCASE 2018 Task 4 (Dev)	34.8	-	-	-
	DCASE 2018 Task 4 (Eva)	26.2	-	-	-

Table 2.40 Dataset used by different authors and reported accuracy (models utilizing weakly labeled data). Dev refers to development dataset. Eva refers to evaluation dataset.

References	Limitations
Lee et al. (2017)	• Iterative selection method to form an ensemble is time-consuming. • Background subtraction method can significantly degrade model performance that used the entire or half the TF representation. • Pseudo labeling method can induce large amount of noise.
Xu et al. (2018)	• Accuracy was low in terms of event-based evaluations for a similar task in DCASE 2018. • Architecture cannot be simplified as it will result in poor detection accuracy • The use of GLU increases the total number of parameters to be learned. • Attention pooling may not work as well as other pooling function such as linear softmax pooling.
Kong et al. (2020)	• CNNT was not able to achieve better performance than a CRNN with attention layer. • Decoding process in a Transformer is slow (if decoding layer is added). • Threshold tuner has convergence issues and is inefficient when dealing with a large number of classes and training examples.
Lu (2018)	• Attention pooling may not work as well as other pooling function such as linear softmax. • Accuracy is low for sound events with a short duration. • Consistency cost requires careful tuning.
Lin et al. (2020)	• Different models are used which require various tuning. • Several hyperparameters have to be tuned carefully.
Lin et al. (2019)	• Different models are used which require various tuning. • Several hyperparameters have to be tuned carefully. • Window sizes of the median filters were derived based on the average event duration in the synthetic dataset which may not be perfect.
Kothinti et al. (2019)	• Complex design with mediocre accuracy. • Accuracy of boundary detection system is lower for audio with overlapping events which is caused by CD learning or suboptimal threshold values. • Audio tagging accuracy is poor.
Pellegrini and Cances (2019)	• Several hyperparameters have to be tuned carefully. • The overall accuracy is also dependent on the audio tagging model's accuracy. • The use of GLU increases the total number of parameters to be learned. • Cosine Similarity was not always helpful.

Table 2.41 Limitations of models utilizing weakly labeled data

Chapter 3. Training a SED System Using Pseudo Strongly Labeled Data

3.1. Motivation

In the previous chapter, a comprehensive review of the current SOTA for SED was presented. While there are many interesting strategies for training a SED system, a limited effort was made to alleviate the need for strongly labeled data through pseudo labels. Among the earlier methodologies reviewed, only Lee et al. (2017) attempted to use some form of pseudo labels to train a SED system. As mentioned earlier, Lee et al. (2017) methodology assumed all frames of a TF representation to contain an event based on the given audio tag (for example, if clip A is weakly labeled to contain Speech, then all frames are considered to contain Speech). Even though such an assumption can induce a significant amount of noise, their methodology was ranked first in the DCASE 2017 challenge task 4. Thus, how pseudo strongly labeled data can be an effective solution remains uncharted territory.

In this thesis, we propose to address the lack of strongly labeled data through the use of pseudo strongly labeled data. Rather than assuming all frames of a TF representation to contain an event based on the given audio tag, we propose using NMF to provide pseudo strong labels for the weakly labeled audio clip. The motivation for using NMF for pseudo labeling stems from the fact that the decomposition of a TF representation using NMF can still give a coarse temporal location of an event even though they do not work very well when applied directly for SED.

We first show an example of what we meant by NMF is capable of giving a coarse temporal location of an event. We propose using an audio clip that contains only a dog barking as an example. In this 10s audio clip, a dog was recorded barking 15 times. We first resampled the audio clip to 32000Hz, and a mel spectrogram was tabulated using the parameters shown in the following table.

NMF was then applied on the mel spectrogram to obtain $\ddot{\mathbf{H}}$ with r set as 1. The derived $\ddot{\mathbf{H}}$ was subsequently rescaled to a range of 0 to 1 and is illustrated in the Figure 3.1.

Variable	Value
FFT window size	1024
Hop length	500
Max frequency	16kHz
Min frequency	0Hz
Mel bins	64

Table 3.1 Parameters used to calculate a mel spectrogram

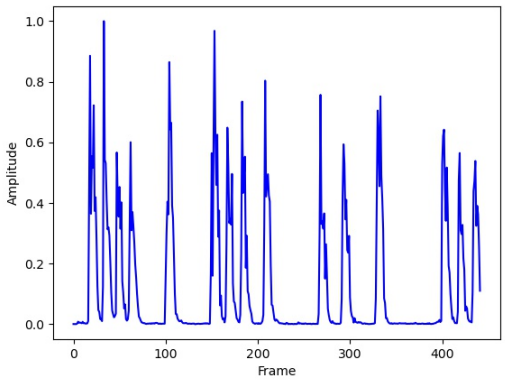


Figure 3.1 Activation matrix of an audio clip containing dog barking

As seen in Figure 3.1, there are exactly 15 spikes which correspond to the total number of barks in the audio clip. This example essential shows that NMF is capable of providing a coarse temporal location of an event. In the following subsection, we showcase the proof of concept where pseudo labels estimated using NMF can be effective when tested on a large-scale dataset.

3.2. Proof of Concept

The content of this section was published as a workshop paper; Chan, T. K., Chin, C. S., and Y. Li. (2019). Non-negative matrix factorization-convolution neural network (NMF-CNN) for sound event detection. In *Proceeding of the Workshop on Detection and Classification of Acoustic Scenes and Events*, pages 40-44, New York, NY, USA.

3.2.1. Dataset Used

In order to prove that NMF can be a useful pseudo labeling tool, we propose to test it on a large-scale dataset for SED in a domestic environment. The dataset used for the proof of concept is the DESED dataset (Turpault et al., 2019), a subset of Audioset (Gemmeke

et al., 2017). The dataset consists of five different subsets, 1) synthetic strongly labeled, 2) weakly labeled, 3) unlabeled, 4) validation, and 5) evaluation.

The synthetic strongly labeled set was generated by the organizers (Turpault et al., 2019) using Scaper (Salamon et al., 2017). Scaper (Salamon et al., 2017) is a tool that automatically generates soundscapes containing random mixtures of the provided events sampled from user-defined distributions, given a set of user-specified background and foreground sound event recordings.

Each subset of data is made up of ten event classes which are typically heard in a domestic environment. However, each subset of data contains a different number of audio clips, where each audio clip is 10s long and can contain multiple events, which may also overlap. Since each clip can contain multiple events, the total number of event occurrences is not equal to the total number of audio clips. Table 3.2 presents key information about the event classes, event distribution, and total occurrences and clips.

It is essential to point out that the evaluation data is not released to the public. We were able to obtain the evaluation results because we participated in the DCASE 2019 challenge task 4.

Event classes	Synthetic strongly labeled	Weakly labeled	Unlabeled	Validation	Evaluation 2019
	Event occurrences				
Speech	2132	550	-	1662	-
Dog	516	214	-	577	-
Cat	547	173	-	340	-
Alarm/bell/ringing	755	205	-	418	-
Dishes	814	184	-	492	-
Frying	137	171	-	91	-
Blender	540	134	-	96	-
Running water	157	343	-	230	-
Vacuum cleaner	204	167	-	92	-
Electric shaver/toothbrush	230	103	-	65	-
Total event occurrences	6032	2244	-	4063	-
Total audio clips	2244	1568	14412	1168	13205

Table 3.2 DCASE 2019 dataset

3.2.2. Unsupervised NMF for Pseudo Labeling

As mentioned in the earlier chapter (Chapter 2.1.2), NMF can be applied directly for SED. The conventional NMF SED approach consolidates the extracted spectral templates from isolated events to form a dictionary for a particular event class. This dictionary is then used to derive $\ddot{\mathbf{H}}$ for a test audio clip, and activated frames in $\ddot{\mathbf{H}}$ are considered to contain the specific event.

Instead of applying the conventional method, we propose to decompose the TF representation of a weakly labeled audio clip using NMF and locate the activated frames in $\ddot{\mathbf{H}}$. As we are only interested in the activated frames, r can be set as 1. If r is set above 1, it can be considered the unsupervised audio source separation technique (Heittola et al., 2013b), which is not the focus of this application. Moreover, there is no indication of which separated stream belongs to which audio event.

Thus, in our application, r is set as 1, and the occurrence of an event is determined by locating the activated frames. Since each audio clip can contain multiple events which can overlap, the activated frames are assumed to contain all events based on the given audio tags. $\ddot{\mathbf{H}}$ is then binarized and used as the pseudo strong labels of a weakly labeled audio clip. The process can be illustrated in Figure 3.2.

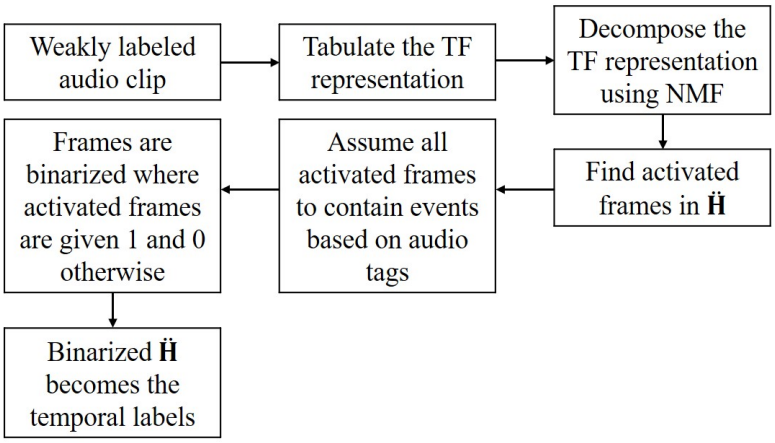
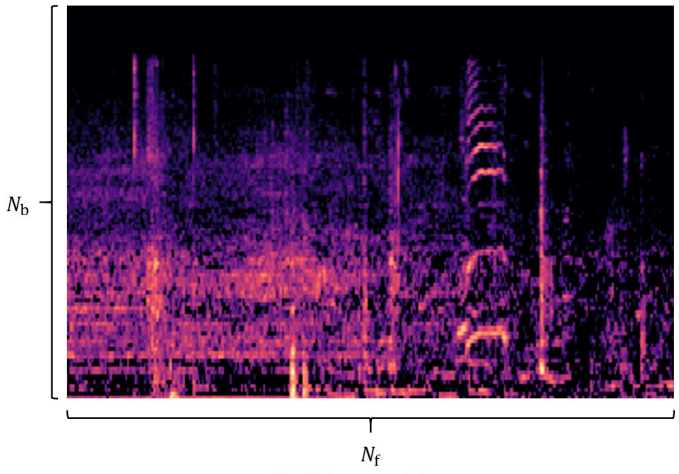


Figure 3.2 Flowchart of proposed unsupervised NMF pseudo labeling method

For better clarity on why a binarized $\ddot{\mathbf{H}}$ can be used as the pseudo strong labels of a weakly labeled audio clip. We began with the description of the model input. A typical TF representation has N_f number of frames and N_b number of frequency bin. Naturally, for a strongly labeled audio clip, one would have the timestamps of an event's occurrence, which can be translated into a one-hot-encoding matrix where 1 indicates the event's occurrence and 0 otherwise.

Given a 10s audio clip with a dog event from 0s to 1s, the calculated TF representation with 100 frames will only have frames 1-10 labeled as having a dog event. Since each frame can contain multiple events, the occurrence of any events is represented in a one-hot-encoding matrix of size N_f by N_e. Using Figure 3.3 as an example, assuming Event A is present in the TF representation from frames 2-4, they are annotated as 1 from frames 2-4 under the event A row in the strong label matrix. Likewise, if Event C is present from frames 1 to 10, they are annotated as 1 from frames 1 to 10 under the event C row in the strong label matrix.



(A) TF representation

A	0	1	1	0	1	0	1	0	1	1	0	1
B	0	0	0	0	0	0	0	0	0	0	0	0
C	1	1	1	1	1	1	1	1	1	1	0	0
D	0	0	0	0	0	0	0	0	0	0	0	0
E	0	0	0	0	0	0	0	0	0	0	0	0

N_e

N_f

(B) Strong labels

Figure 3.3 Comparison between a TF representation and the strong labels of the TF representation

If NMF is used to decompose a TF representation, $\mathbf{TF} \in \mathbb{R}_+^{N_b \times N_f}$, one would obtain $\ddot{\mathbf{W}} \in \mathbb{R}_+^{N_b \times r}$ and $\ddot{\mathbf{H}} \in \mathbb{R}_+^{r \times N_f}$. With r is set as 1, $\ddot{\mathbf{H}}$ is a single row nonnegative matrix (or vector), which after binarization, represents the temporal labels. In our proposed method, $\ddot{\mathbf{H}}$ is then transposed and used to replace the column of a strong label matrix representing the event present in an audio clip.

For the rest of this thesis, the weakly labeled data which has been given pseudo strong labels will be termed pseudo strongly labeled data.

3.2.3. Proposed SED Model

The proposed model used for proof of concept is a simple CNN based on the Kong et al. (2019a) DCASE 2019 cross-task model. As seen in Fig. 3.4, the main difference between the two models is that our proposed model used a slightly larger kernel size and has lesser convolution layers. In our model, the clip-level prediction is obtained by applying a temporal max pooling on the frame-level prediction. In contrast, in (Kong et al., 2019a), the clip-level prediction is obtained by applying another FC layer after the application of temporal max pooling on the frame-level prediction.

Proposed	Kong et al (2019)
Input: Log-mel spectrogram	
Convolution Layer Filters: 64 Kernel Size: (5, 5) Stride: (1, 1) BN ReLU	Convolution Layer x 2 Filter: 64 Kernel Size: (3, 3) Stride: (1, 1) BN ReLU
Max pooling: (2, 2)	
Convolution Layer Filters: 128 Kernel Size: (5, 5) Stride: (1, 1) BN ReLU	Convolution Layer x 2 Filter: 128 Kernel Size: (3, 3) Stride: (1, 1) BN ReLU
Max pooling: (2, 2)	
Convolution Layer Filters: 256 Kernel Size: (5, 5) Stride: (1, 1) BN ReLU	Convolution Layer x 2 Filter: 256 Kernel Size: (3, 3) Stride: (1, 1) BN ReLU
Max pooling: (2, 2)	
Convolution Layer Filters: 512 Kernel Size: (5, 5) Stride: (1, 1) BN ReLU	Convolution Layer x 2 Filter: 512 Kernel Size: (3, 3) Stride: (1, 1) BN ReLU
Global Max Pooling Along Frequency Axis	
FC Layer	FC Layer
Interpolate	Interpolate
Output: Frame-Level Prediction	
Global Max Pooling Along Time Axis	
Output: Clip-Level Prediction	FC Layer
	Output: Clip-Level Prediction

Figure 3.4 Difference between proposed model and Kong et al. (2019a) model

3.2.4. Experiment Setup

As segments containing higher frequency may not be helpful for event detection in daily life (Lu, 2018), in this experiment, audio clips were first resampled to 32 kHz since the frequency range was suggested to contain the most energies (Kong et al., 2019a). The spectrogram is then tabulated using STFT with a window size of 1024 and a hop length of 500. Mel spectrogram is then tabulated using 64 mel bins within a cut-off frequency of 50Hz to 14 kHz. Once all mel spectrograms of weakly labeled data have been tabulated, we provide the pseudo strong labels for each clip based on the unsupervised NMF pseudo labeling approach described in Section 3.2.2. Once we obtain the pseudo strong labels, mel spectrograms are converted to log-scale by applying a logarithm operator.

We began our experiment by training our proposed model using a batch size of 16 made up of only pseudo strongly labeled data. The frame-level loss can be defined as

$$l_{\mathrm{f}} = \frac{1}{N_{\mathrm{e}} \times N_{\mathrm{f}}} \sum_{i=1}^{N_{\mathrm{e}}} \sum_{j=1}^{N_{\mathrm{f}}} [g_{i,j} \log(M_{i,j}) + (1 - g_{i,j}) \log(1 - M_{i,j})] \tag{3.1}$$

where $g_{i,j}$ represents the ground truth for event i at frame j. $M_{i,j}$ represents the predicted probability for event i at frame j. Note that for Kong et al. (2019a) model, they are training it using clip-level BCE. Based on this loss, the proposed model is updated using Adam (Kingma and Ba, 2015). The learning rate began at 0.001 and is annealed by a factor of 0.1 every 50 iterations.

The post-processing method follows (Kong et al., 2019a), where each frame is considered activated (i.e., containing an event) if the probability exceeds 0.5. Neighboring frames of an activated frame are also considered activated if they exceed a lower bound threshold of 0.1. Subsequently, similar events are joined together if the difference between the first event offset and the second event onset is less than 0.2s. Finally, an event with a duration shorter than 0.2s is removed as they are considered noise.

In this section, all experiments were conducted on a system using an Intel Processor i7-6700HQ with a base frequency of 2.6GHz, 16GB ram and a GTX1060 GPU.

3.2.5. Evaluation Metric

Accuracy is measured using event-based and segment-based F1-score (Mesaros et al., 2016). Event-based metrics compare system output and corresponding reference by the event, whereas segment-based metrics compare system output and reference in short time segments (Mesaros et al., 2016).

Based on the definitions given in (Mesaros et al., 2016), the intermediate statistics for event-based metric are defined as follows

- True Positive (TP): a predicted event with its onset and offset tally with the actual annotation. A margin of error, also known as the collar, is usually given for the predicted onset and offset.
- False Positive (FP): a predicted event with its onset and offset do not tally the actual annotation.
- False Negative (FN): a misdetection, the system did not predict any event even though there is an event occurrence.

The intermediate statistics for the segment-based metric consider the accuracy in segments rather than considering the entire event. The intermediate statistics for segment-based metric are defined as follows

- TP: a predicted event tally with the actual annotation in that segment.
- FP: a predicted event does not tally with the actual annotation in that segment.
- FN: a misdetection, the system did not predict any event in the segment even though there is an event occurrence.

Based on the intermediate statistics, the event-based or segment-based F1-score can be calculated using the following equation

$$F1-score = \frac{2Precision \times Recall}{Precision + Recall} \tag{3.2}$$

$$Precision = \frac{TP}{TP+FP} \tag{3.3}$$

$$Recall = \frac{TP}{TP+FN} \tag{3.4}$$

As recommended in (Mesaros et al., 2016), the event-based metric should be used as the primary measure to evaluate the performance and capabilities of any models proposed. This is because event-based metrics better illustrate the ability to locate and label longer blocks of audio correctly.

Note that the official challenge metric (i.e., metric used to determine ranking) is the event-based F1-score. In the challenge, the allowable collar for an event onset is 0.2s, and

the allowable collar for an event offset is 20% of event duration. On the other hand, the segment length used for calculating the segment-based F1-score is set as 0.2s.

3.2.6. Results and Discussion

Method	F1-score (%)	
	Event-based	Segment-based
Proposed	**29.7**	55.8
Kong et al. (2019a)	24.1	**63.0**
Baseline (Turpault et al., 2019)	23.7	55.2

Table 3.3 Comparison against Kong et al. (2019a) and baseline system on the validation dataset

Based on the experiment setup, we tested our system against the system proposed by Kong et al. (2019a) and the baseline system (Turpault et al., 2019) on the DCASE 2019 validation dataset. As seen in Table 3.3, our system trained using the pseudo strongly labeled data can obtain an event-based F1-score of 29.7%, which is 5.6% higher than the system proposed by Kong et al. (2019a). At the same thing, we also observed a higher event-based accuracy of 6% compared to the baseline system.

However, while our system's event-based F1-score is higher than the system proposed by Kong et al. (2019a), the segment-based F1-score is lower. We hypothesized that such a phenomenon is due to how we provide the pseudo strong labels to the weakly labeled audio clips. As mentioned earlier, if each clip contains multiple events, each activated frame is assumed to contain all events; this may cause a lower segment-based F1-score.

As mentioned in Section 3.2.4. Experiment Setup, the training data only consists of the pseudo strongly labeled data. We then experimented with different combinations of training data. Each row in Table 3.4 represents the results of our system trained using a different set of training data, and the set of training data is represented by C1 to C7. The representation for C1 to C7 is given in the table caption. Therefore, Proposed-C1, where C1 is denoted as pseudo strongly labeled data in the table caption, means that our system is trained using only pseudo strongly labeled data. Each set of training data can be derived using the flow chart given in Figure 3.5.

As seen in Table 3.4, training with only synthetic strongly labeled data can result in a poor detection accuracy which may be due to the mismatch in domains. However, detection accuracy can be improved by training our proposed model with both pseudo strongly labeled data and synthetic strongly labeled data.

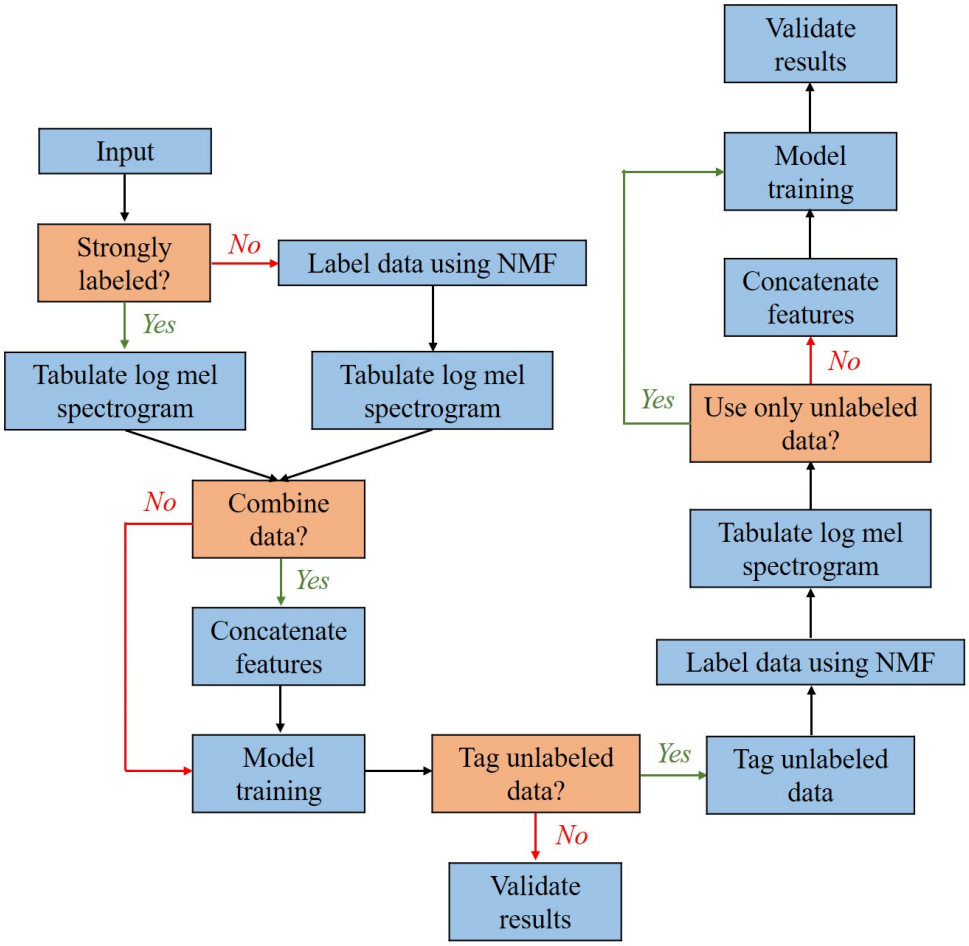


Figure 3.5 Flowchart for data combination

Method	F1-score (%)	
	Event-based	Segment-based
♣ Proposed-C1	29.7	55.8
♠ Proposed-C2	15.3	43.6
♢ Proposed-C3	**30.4**	**57.7**
♡ Proposed-C4	25.5	45.9
★ Proposed-C5	27.2	48.5
□ Proposed-C6	26.6	47.1
△ Proposed-C7	27.8	50.9

Table 3.4 F1-Score on validation dataset using different types of data. ♣ C1-Pseudo strongly labeled data. ♠ C2- Synthetic strongly labeled data. ♢ C3- Pseudo strongly labeled data and synthetic strongly labeled data. ♡ C4- Unlabeled data (labeled using ♣ Proposed-C1). ★ C5- Pseudo strongly labeled data and unlabeled data (labeled using ♣ Proposed-C1). □ C6- Unlabeled data (labeled using ♢ Proposed-C3). △ C7- Pseudo strongly labeled data, synthetic strongly labeled data and unlabeled data (labeled using ♢ Proposed-C3).

In order to incorporate a large amount of unlabeled data for model training, we propose to tag all the unlabeled data using different systems and subsequently provide each unlabeled clip with pseudo strong labels. By tag, we meant that we only provide the audio tags for each unlabeled clip. The reason for not providing the temporal labels for each unlabeled clip using the trained system is that the event-based F1-score of our system is still not considered high compared to the audio tagging accuracy. By providing only the audio tag, we can reduce the amount of noise that may be given compared to providing temporal labels directly.

As seen in Table 3.4, when trained with only unlabeled data (Proposed-C4 and Proposed-C6), the system accuracy is lower than using the pseudo strongly labeled data. This is because the audio tagging accuracy is not perfect, which can still induce a certain amount of noise. While including the use of pseudo strongly labeled data and synthetic strongly labeled data can increase the accuracy (Proposed-C5 and Proposed-C7), the improvement is still insufficient to outperform our system trained using only pseudo strongly labeled data. This is because the amount of unlabeled data overweighs the combined amount of pseudo strongly labeled data and synthetic strongly labeled data by several folds. Thus, the noise present in the training data can still cause a significant impact on the detection accuracy.

Method	F1-score (%)	
	Event-based	Segment-based
♣ Proposed-C1	29.7	55.6
♢ Proposed-C3	**31.0**	58.2
★ Proposed-C5	26.9	48.7
△ Proposed-C7	27.7	50.5
Kong et al. (2019a)	22.3	**59.4**
Baseline (Turpault et al., 2019)	25.8	53.7

Table 3.5 F1-Score on evaluation 2019 dataset using different types of data. ♣ C1-Pseudo strongly labeled data. ♢ C3- Pseudo strongly labeled data and synthetic strongly labeled data. ★ C5- Pseudo strongly labeled data and unlabeled data (labeled using ♣ Proposed-C1). △ C7- Pseudo strongly labeled data, synthetic strongly labeled data and unlabeled data (labeled using ♢ Proposed-C3).

We then submitted the top 4 systems (Proposed-C1, Proposed-C3, Proposed-C5, Proposed-C7) to the DCASE 2019 challenge task 4. As seen in Table 3.5, our systems demonstrated a level of robustness where the largest drop in event-based F1-score is only 0.3%. Our best system (Proposed-C3) can even obtain a higher event-based and segment-based F1-score on the evaluation 2019 dataset.

In contrast, the system proposed by Kong et al. (2019a) observed a more considerable drop in accuracy for both event-based and segment-based F1-score (1.8% decrease in event-based accuracy and 3.6% decrease in segment-based accuracy). Based on the event-based F1-score, our best system (Proposed-C3) outperforms the system proposed by Kong et al. (2019a) by 8.7%. At the same time, our best system (Proposed-C3) outperforms the baseline system by 5.2%. Such results showcase the effectiveness of using pseudo strongly labeled data and indicate that it can be a valuable alternative to train a SED system.

3.3. Semi-supervised NMF-CNN For SED

With the proof of concept as shown in the previous section, we continued with the use of pseudo strongly labeled data to train a SED system. As an improvement, we propose a novel methodology to label the weakly labeled data using NMF (Lee and Seung, 1999) in a supervised manner. The main idea is to extract the spectral templates from the synthetic data and derive the activation matrix from the weakly labeled data. Activated frames would then serve as the pseudo strong label.

In order to better leverage both the synthetic and unlabeled data, we then train our SED system in a combinative transfer learning and semi-supervised learning framework. Rather than using the Mean-Teacher approach (Tarvainen and Valpola, 2017), we propose using two different CNNs where they are trained synchronously to pursue different targets. In the inference stage, one will produce the clip-level prediction, while the other will produce the frame-level prediction.

By comparing against the top 3 submissions from the DCASE 2019 challenge task 4, our single system can even outperform the ensemble system of the first-place submission. As our system is also submitted to the DCASE 2020 challenge task 4, results show that our proposed methodology can have a minimum winning margin of 7% against the baseline system and is competitive against the other SOTA. A post-challenge analysis was also conducted and revealed that our system is much more duration robust than the top-place submission in DCASE 2020 challenge task 4. In the following subsections, the description of the proposed framework is discussed in detail.

The content of this chapter was summarized into a journal paper and is currently under review.

3.3.1. Dataset Used

We continued using the DESED dataset (Turpault et al., 2019) for the subsequent experiments and evaluations; however, it should be noted that the number of synthetic strongly labeled data and evaluation data is different. The synthetic dataset is slightly larger, and the maximum polyphony (i.e., the maximum number of events in each clip) is set as 2. In addition, the evaluation dataset may contain clips that may have a duration of 10s or 5 mins. The distribution of the data can be seen in Table 3.6. Thus, to prevent confusion, we termed this set of data as DCASE 2020 dataset.

It is important to emphasize that the evaluation dataset is not released to the public, and we were able to obtain the results because we participated in the DCASE 2020 challenge task 4.

Event classes	Synthetic strongly labeled	Weakly labeled	Unlabeled	Validation	Evaluation 2020
	Event occurrences				
Speech	2625	550	-	1662	-
Dog	858	214	-	577	-
Cat	783	173	-	340	-
Alarm/bell/ringing	532	205	-	418	-
Dishes	1115	184	-	492	-
Frying	201	171	-	91	-
Blender	414	134	-	96	-
Running water	271	343	-	230	-
Vacuum cleaner	432	167	-	92	-
Electric shaver/toothbrush	334	103	-	65	-
Total event occurrences	7475	2244	-	4063	-
Total audio clips	2595	1568	14412	1168	12566

Table 3.6 DESED 2020 dataset

3.3.2. Supervised NMF for Pseudo Labeling

As shown earlier, NMF (Lee and Seung, 1999) can approximate strong labels for the weakly labeled data in an unsupervised manner. However, assuming all frames to contain events based on the audio clip may not be valid for all scenarios. As a follow-up, we improve on the pseudo labeling strategy by estimating the pseudo strong labels using NMF in a supervised manner.

The first step is to extract the event template from the synthetic audio clips to form a dictionary for different event classes. Since a synthetic sound clip can contain multiple events, temporal masking is applied to the mel spectrogram using the given temporal annotations. Templates of each event class are retrieved from the masked mel spectrogram using NMF by allowing r to be set as 1. For example, if synthetic clip A has Speech and Cat occurring at frame 1 to 100 and 100 to 110 respectively, all frames from 101 onwards are masked to extract the Speech template followed by masking all frames except frames 100 to 110 to extract the Cat template. Note that for events that are overlapping, we process them in the same manner.

As weakly labeled data possessed the audio tags, we apply the corresponding dictionary on the audio clip to derive the activation matrix. Activated frames are assumed to contain the event class. For example, if Clip B contains Speech and Dog, we first apply NMF to decompose Clip B using Speech dictionary and with r set as 1 to derive the $\ddot{\mathbf{H}}$. Frames that are over a threshold are assumed to contain only Speech. Based on this condition, $\ddot{\mathbf{H}}$ is binarized and becomes the temporal label for Speech event, where 1 represents the occurrence of Speech and 0 otherwise. A similar procedure is applied to derive the temporal annotation for Dog by using the Dog dictionary. The entire labeling process is illustrated in Figure 3.6.

As each weakly labeled clip has 10 events, this would indicate there are 10 different temporal labels. For events that do not exist in the weakly labeled clip, the temporal labels for those non-existing events are simply vectors of 0. Thus the columns of these events in the strong label matrix are simply 0. In contrast, the binarized $\ddot{\mathbf{H}}$ is used to replace the column of a strong label matrix, which represents the event that is present in an audio clip. (Note that the formation of a strong label is similar to the procedure as discussed in Section 3.2.2)

Once the labeling process is completed, the mel spectrogram is converted to log-scale, which will be used as model input.

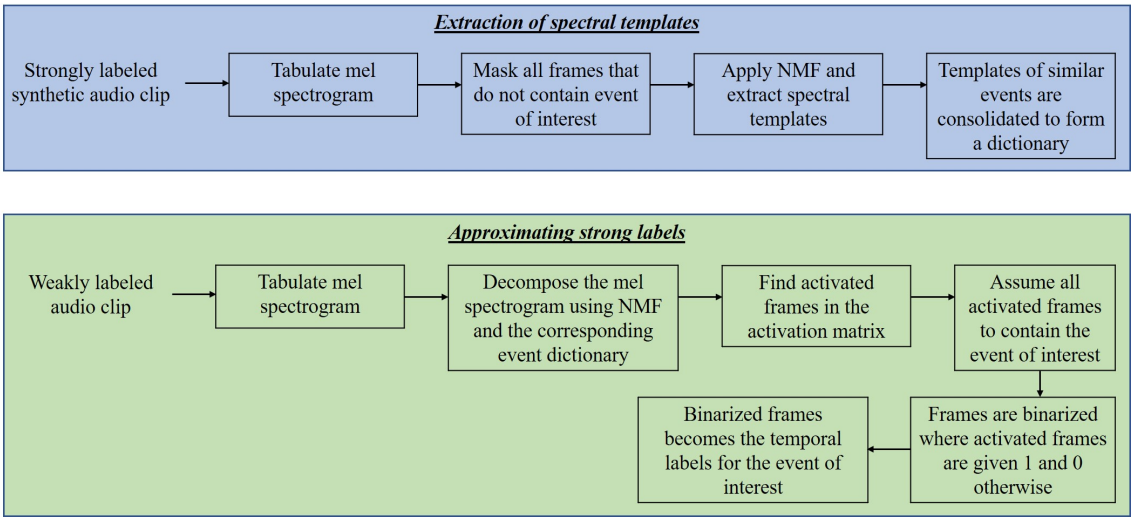


Figure 3.6 Proposed supervised NMF labeling method

3.3.3. Proposed Semi-supervised Learning Framework

Lin et al. (2019) mentioned that there could be a trade-off in SED performance due to the pooling operation. The accuracy of clip-level detection can be improved with higher temporal compression (pooling along the time axis). In comparison, the accuracy of the frame-level prediction can be improved with lower temporal compression (Lin et al., 2019).

Therefore, we propose a Student Model (SM) with no temporal compression for frame-level prediction and a Teacher Model (TM) with temporal compression for clip-level prediction. In addition to the difference in pooling size, SM has fewer convolutional layers, adopted context gating (Miech et al., 2018) as the activation function instead of ReLU, and has a slightly higher dropout rate. The details of SM and TM can be found in Figure 3.7 and Figure 3.8, respectively.

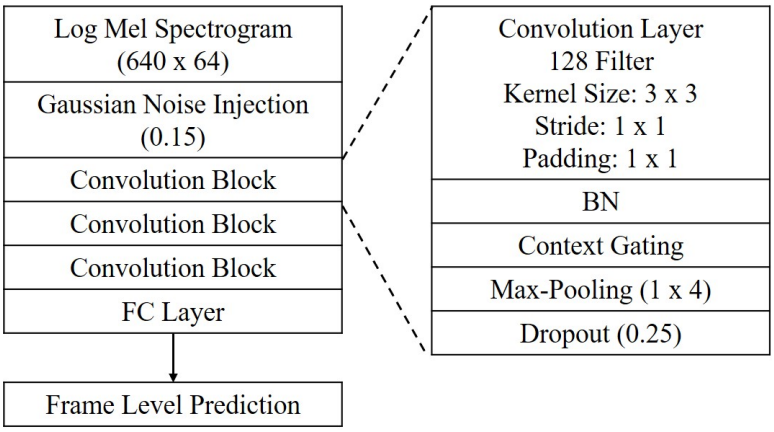


Figure 3.7 SM for frame-level prediction

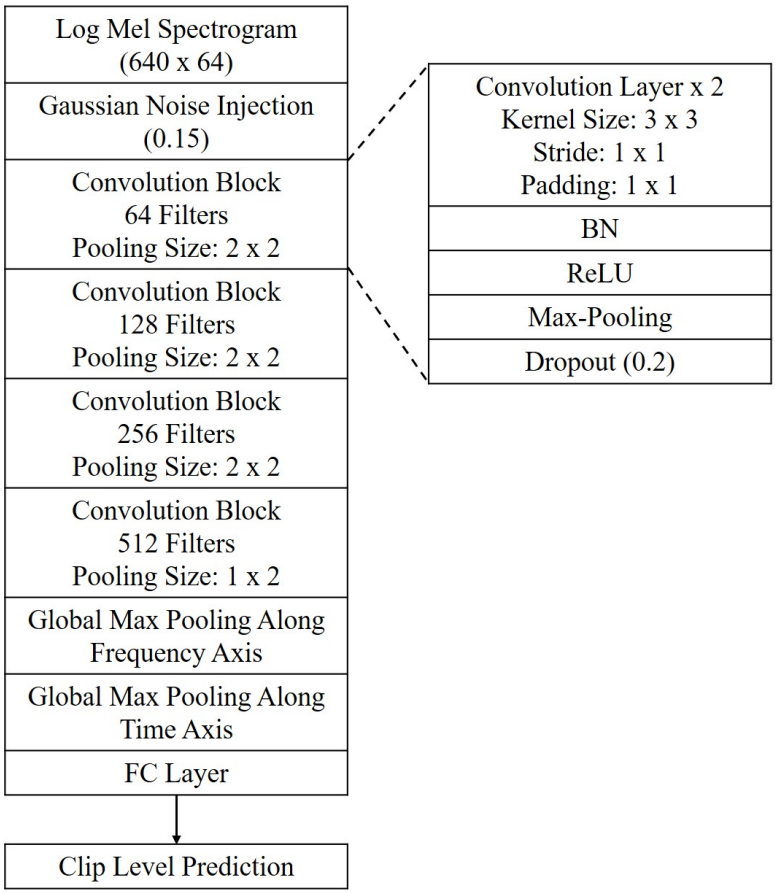


Figure 3.8 TM for clip-level prediction

Given that $S_{i,j}$ represents the SM's predicted probablity of event i at frame j. $g_{i,j}$ represents the ground truth of event i at frame, j. The frame-level loss, l_{f}, is defined as

$$l_{\mathrm{f}} = \frac{1}{N_{\mathrm{e}} \times N_{\mathrm{f}}} \sum_{i=1}^{N_{\mathrm{e}}} \sum_{j=1}^{N_{\mathrm{f}}} [g_{i,j} \log(S_{i,j}) + (1 - g_{i,j}) \log(1 - S_{i,j})] \tag{3.5}$$

On the other hand, the clip-level loss, l_{c}, is defined as

$$l_{\mathrm{c}} = \frac{1}{N_{\mathrm{e}}} \sum_{i=1}^{N_{\mathrm{e}}} [z_i \log(T_i) + (1 - z_i) \log(1 - T_i)] \tag{3.6}$$

where T_i represents the TM's predicted probability of event i in a labeled sample and z_{i} represents the grouth truth of event i in the same sample. We hypothesize that enforcing the prediction of SM to be consistent with TM, it could produce a better frame-level prediction. As the prediction output of SM is in frame level, we apply a global max pooling on the time axis of SM's prediction to obtain the clip level prediction. Instead of using BCE, MSE is used as the consistency loss function as it was evaluated to be a better consistency loss function (Laine and Aila, 2017). However, the consistency loss, l_{con}, will only be

calculated if TM is confident with its prediction. Thus, l_{con} is given as

$$l_{\text{con}} = \begin{cases} \frac{1}{N_{\text{e}}} \sum_{i=1}^{N_{\text{e}}} (S_i^{\text{c}} - T_i)^2 & \text{if } \max(\mathbf{T}) > \lambda \\ 0 & \text{otherwise} \end{cases} \tag{3.7}$$

where S_i^{c} represents the SM's predicted probability of event i in a labeled sample and $\mathbf{T}$ represents the vector containing the TM's predicted probabilities of all events present in a sample. λ is the confidence level used to regulate the loss.

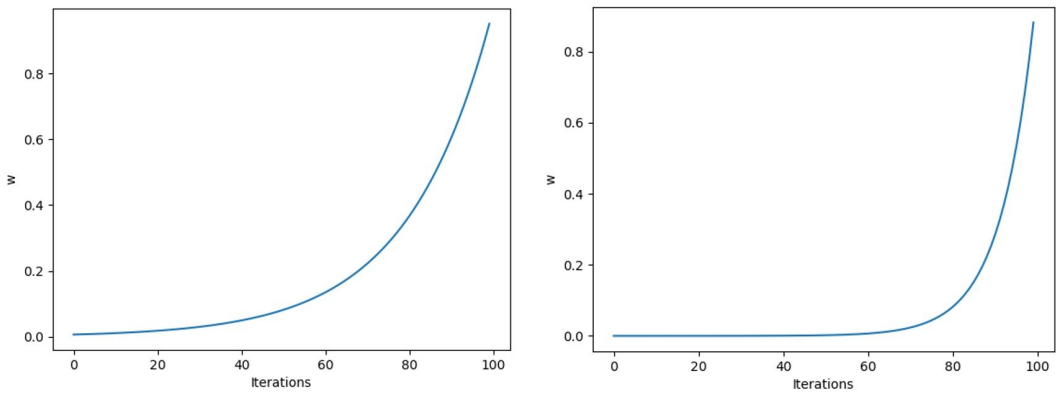
In order to leverage the large corpus of unlabeled data for semi-supervised learning, we also enforce the consistency of prediction on the unlabeled data. Thus the consistency cost on the unlabeled data, l_{unlabel}, can be defined as

$$l_{\text{unlabel}} = \begin{cases} \frac{w}{N_{\text{e}}} \sum_{i=1}^{N_{\text{e}}} (\tilde{S}_i^{\text{c}} - \tilde{T}_i)^2 & \text{if } \max(\tilde{\mathbf{T}}) > \lambda \\ 0 & \text{otherwise} \end{cases} \tag{3.8}$$

where $\tilde{S}_i^{\text{c}}$ and $\tilde{T}_i$ represent the SM's and TM's predicted probability of event i in an unlabeled sample, respectively. $\tilde{\mathbf{T}}$ represents the vector containing the TM's predicted probabilities of all events present in an unlabeled sample. w is a weighing parameter defined as (Laine and Aila, 2017)

$$w = \exp(-5(1-P)^2) \tag{3.9}$$

where P is a positive value representing the training progression and will be discussed in more detail in the next section. The purpose of w is to further regulate l_{unlabel} as it was found that if consistency loss is given too much weightage in the early training stage, it can lead to a suboptimal solution (Laine and Aila, 2017).



(A) Scaling factor = 5 (B) Scaling factor = 12.5

Figure 3.9 Effects of different scaling factor

The integer 5 in Equation 3.9 is a scaling factor that controls how steep the transition is from 0 to 1. As seen in Figure 3.9, the larger the value, the steeper is the transition. With a larger scaling factor, w will be close to 0 for most iterations and sharply transit to a larger value at the later iterations. Thus a scaling factor of 5 can be seen as a suitable scaling factor that has a smoother transition (Note: the use of 5 is also proposed in (Laine and Aila, 2017)).

In the next section, we then present the experimental setup on how the models are trained.

3.3.4. Experiment Setup

In the preprocessing step, all audio clips that are longer or shorter than 10s are truncated or padded to have an equal duration of 10s. As for the 5 mins audio clips, they are split into clips with a duration of 10s. Processed clips are resampled at 22,050 Hz, and spectrograms are tabulated using an FFT window size of 2048 with a hop length of 345. Mel-spectrograms are then tabulated using 64 mel filter banks. Based on such a setting, a tabulated mel spectrogram would have a size of 640 by 64, which is seen in Figure 3.7 and Figure 3.8.

The experimental setup consists of two stages, 1) the model training stage and 2) the model inference stage. Based on our proposed methodology, models are trained in a two-phase transfer learning framework in the model training stage. The first phase is referred to as the transfer learning phase, while the second phase is referred to as the adaptation phase.

Transfer learning is a technique that aims to improve the predictive capability of a model on the target domain using the knowledge learned in the source domain (Pan and Yang, 2010). In other words, a model that is trained and updated using a dataset that may or may not be related to the target domain is used as a starting point for the target domain and is trained and updated using the target domain dataset. Thus, in the transfer learning phase, the model is trained using a related dataset, and in the adaptation phase, the model is then trained using the target domain dataset. The entire training procedure can be seen in Figure 3.10.

The first phase, which lasts 5 epochs, utilizes only the synthetic data to train the models with a batch size of 64. In this phase, the learning rate begins at 0.0012 and is reduced by a factor of 0.1 every 300 iterations. Since the unlabeled data are not utilized in this phase, the loss components only consist of l_{f}, l_{c} and l_{con}. λ which is used to calculate l_{con} is set

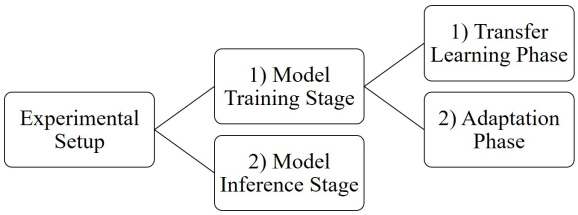


Figure 3.10 Flowchart of experiment setup

as 0.9. Based on the calculated losses, models are updated using Adam (Kingma and Ba, 2015).

In the second phase (i.e., 6th epoch onwards), the training data is first replaced with the real data, consisting of both the pseudo strongly labeled and unlabeled data. Batch size remains at 64 but the data is split equally into pseudo strongly labeled clips and unlabeled audio clips. At the start of phase 2, the learning rate is reset to 0.0012 and is annealed according to the cosine function given as (Loshchilov and Hutter, 2017)

$$LR_{\text{curr}} = LR_{\text{min}} + 0.5(LR_{\text{max}} - LR_{\text{min}})(1 + \cos(\frac{P_{\text{curr}}}{P_{\text{tot}}}\pi)) \tag{3.10}$$

where LR_{max} represents the maximum learning rate and is set as 0.0012. LR_{min} represents the minimum learning rate, which is set as 1e-6. P_{curr} represents the current training iteration and P_{tot} represents the maximum training iterations before a learning rate reset. The way Equation 3.10 is defined, the larger the value of P_{curr}, the smaller is LR_{curr}. Based on the framework of learning rate reset (Loshchilov and Hutter, 2017), the learning rate will be reset when P_{curr} equals P_{tot} and P_{curr} will revert to 0 while P_{tot} is multiplied with an integer, P_{mult}, which can delay the next restart if P_{mult} is larger than 1. Loshchilov and Hutter (2017) explained that such a learning scheme benefits the performance of a deep NN by allowing faster convergence and deriving a better solution. In this chapter, we begin our experiment by setting P_{tot} as 1 epoch and P_{mult} as 2. Based on such settings, the learning rate will be reset after 1 epoch in the first reset, and the learning rate will only be reset after 2 epochs in the second reset, while in the third reset, the learning rate will only be reset after 4 epochs.

In this phase, the loss components would include l_{f}, l_{c}, l_{con}. as well as l_{unlabel} with λ remains as 0.9. As mentioned earlier, w is required to regularize the contribution of l_{unlabel} where the calculation of w is affected by P which represents the training progression. With the inclusion of the learning rate reset scheme, we propose to define P according to the

learning rate reset schedule. Thus, P can be written as

$$P = \frac{P_{curr}}{P_{tot}} \tag{3.11}$$

Therefore, w will also be reset to the minimal value whenever the learning rate is reset. Similarly, models are updated using Adam (Kingma and Ba, 2015) based on the calculated losses. The total epoch used to train the models in phase 2 is set as 100 epochs. As such, the total training epoch consisting of both phases is 105 epochs.

In the inference stage, the trained models are used for audio tagging and temporal annotation. The tagging threshold is set as 0.5, which determines an event's presence if the predicted probability by TM exceeds 0.5. The frame-level prediction given by SM for the detected event will then be used to determine the onset and offset. This frame-level prediction is first subjected to further processing before the temporal labels are given. The frame-level prediction is first smoothed using two passes of the median filter with event-specific window size for this framework. All frames with probabilities that exceed the event-specific threshold are considered activated (i.e., the occurrence of an event). Following (Kong et al., 2019a), neighboring frames are also considered activated if they exceeded a lower bound threshold of 0.08. In addition, detected events with a duration of shorter than 0.1s are removed as they are considered as noise. Subsequently, two similar events are concatenated together if the difference between the first event offset and the second event onset is shorter than 0.2s. As for the 5 mins audio clips that were split in the preprocessing stage, we combine the detection results across the split clips.

Based on the information given in the previous sections, the entire framework can be summarized and illustrated in Figure 3.11.

In this section, all experiments were conducted on a system using an Intel Processor i7-6700HQ with a base frequency of 2.6GHz, 16GB ram and a GTX1060 GPU.

3.3.5. Results and Discussion

In this section, several different experiments are conducted on the validation dataset to investigate the importance of synthetic data, consistency loss functions, and several different hyperparameters, post-processing methods that can affect the detection accuracy. For subsequent experiments, the accuracy of a system is measured using only the event-based F1-score (Mesaros et al., 2016).

We began the experiment by testing the importance of using synthetic data. A system trained based on the experiment setup described earlier in Section 3.3.4 is referred to as

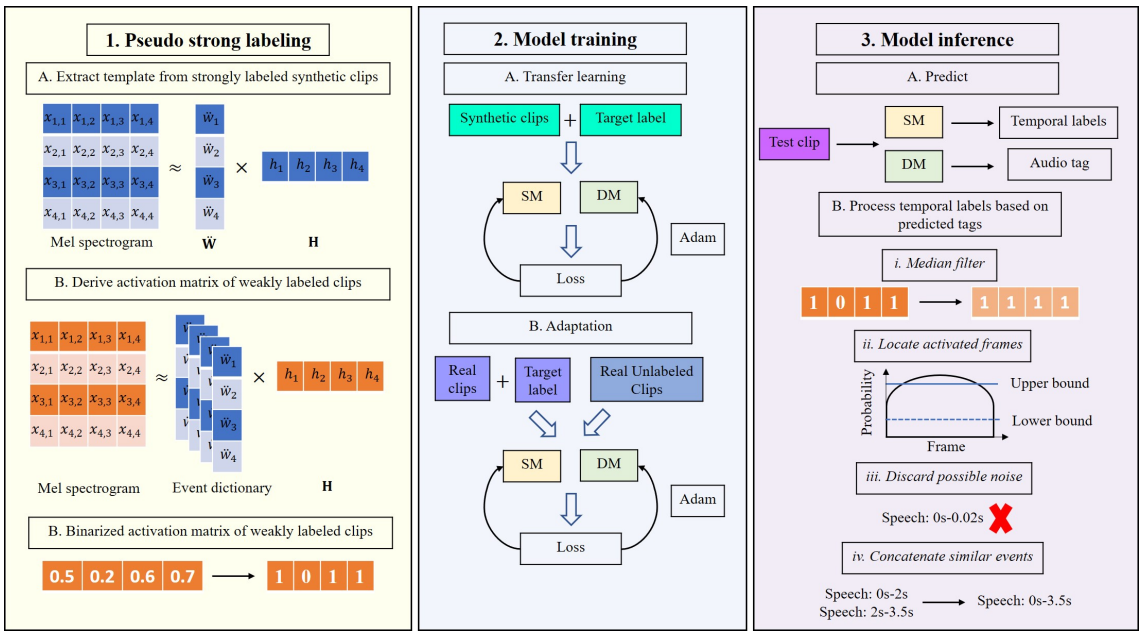


Figure 3.11 Flowchart of experiment setup

Methodology	Event-based F1-score (%)
System 1 (using only real data in adaptation phase)	45.2
System 1 (without transfer learning)	43.4
System 2 (using both real and synthetic data in adaptation phase)	**45.7**

Table 3.7 Importance of synthetic data

System 1 for the rest of this chapter. To determine the importance of synthetic data, we first eliminate the transfer learning phase in the model training stage. As seen in Table 3.7, model accuracy can reduce by 1.8% without the transfer learning phase. This performance degradation is also consistent with the findings in (Lin et al., 2019). Thus, results showing that the use of synthetic data for transfer learning can help improve the performance of the models.

A different transition from the transfer learning phase to the adaptation phase is done in the following experiment. Instead of replacing the training data from synthetic to real data, models were made to train with both synthetic and real data in the adaptation phase. A batch size of 64 is used where half of the batch consists of real unlabeled data while the other half consists of a mixture of synthetic data and pseudo strongly labeled data. Such setup is referred to as the combinative adaptation phase, and the system trained using this setup is referred to as System 2 in Table 3.7. As seen in Table 3.7, using a combinative adaptation phase can allow the detection accuracy to increase. We hypothesize that strongly

λ	Event-based F1-score (%)
0.1	44.3
0.3	44.4
0.5	44.6
0.7	43.8
0.9	**45.7**

Table 3.8 Sensitivity of λ

P_{tot}	P_{mult}	Event-based F1-score (%)	Peak accuracy at epoch
No learning rate reset		44.7	92
1	1	44.1	102
5	1	45.4	33
10	1	44.7	70
1	2	**45.7**	32
5	2	45.3	69
10	2	44.7	66

Table 3.9 Effects of using different P_{tot} and P_{mult}

labeled synthetic clips in the adaptation phase alleviate the complications caused by noisy strong labels approximated using NMF, thus improving the overall detection accuracy.

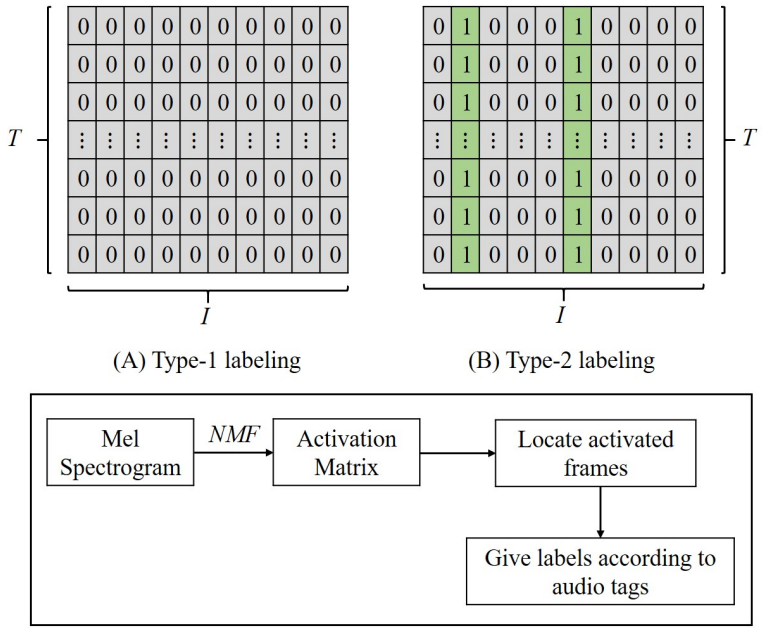
Since it was found that the combinative adaptation phase can yield better detection accuracy, we proceed with this setup for the rest of our experiment. In the subsequent experiment, different λ values were tested to examine the sensitivity of λ. As seen in Table 3.8, allowing lower confidence predictions to contribute to l_{con} and l_{unlabel}, system accuracy can be negatively affected. We hypothesize that at the earlier learning stage, the TM has yet to achieve optimal detection capability (i.e., low audio tagging accuracy at the early training stage). Thus, by enforcing SM to be consistent with TM, SM may be force to produce incorrect predictions, which leads to performance degradation. Therefore, a high λ value of 0.9 is required to prevent a suboptimal solution.

In the proposed framework, P_{tot} controls how fast the learning rate will reduce from LR_{max} to LR_{min}. In our experiment if P_{tot} is smaller than 5 epochs, P_{mult} must be at least 2 to prevent the large fluctuation of learning rate throughout the training process. If P_{tot} is larger than 5 epochs, P_{mult} can be set as 1 as the learning rate transition from LR_{max} to LR_{min} can be considered slow and steady. As seen in Table 3.9, having a large fluctuation

of learning rate during the training process (i.e., setting P_{tot} and P_{mult} as 1) can result in a worse solution than a system trained without learning rate reset.

Subsequently, we found that it is not a guarantee that a better solution can be found following a learning rate reset. However, the peak accuracy of a SED system can be found much earlier for a system trained using learning rate reset than a system trained without learning rate reset. As seen in Table 3.9, the accuracy of a system trained without the use of learning rate reset is at 44.7%, which is similar to a system trained using learning rate reset with P_{tot} set as 10 and P_{mult} set as 1 as well as a system trained using learning rate reset with P_{tot} set as 10 and P_{mult} set as 2. However, the peak accuracy of the systems trained using learning rate reset can be found much earlier (i.e., up to 26 epochs) than the system trained without learning rate reset. Such improvement is also consistent with the results shown in (Loshchilov and Hutter, 2017). Thus, with appropriate P_{tot} and P_{mult}, the overall performance of the system can benefit from learning rate reset.

We then investigate the effectiveness of our proposed NMF labeling method. As a comparison, we experiment with three additional types of labeling methods. 1) each remaining weakly labeled audio clip is given a zero matrix of 640 by 10, indicating that there is no presence of any event. 2) each remaining weakly labeled audio clip is given a binary matrix of 640 by 10 where a specific column is filled with ones based on the given audio tags. This will indicate the presence of an event throughout an entire clip. 3) Unsupervised NMF labeling method as proposed in Section 3.2.2. An example of the three labels method is illustrated in Figure 3.12.



(A) Type-1 labeling

(B) Type-2 labeling

(C) Unsupervised NMF labeling

Figure 3.12 Difference in labeling method

Pseudo labeling approach	Event-based F1-score (%)
Proposed NMF labeling	**45.7**
Type-1 labeling	30.7
Type-2 labeling	34.1
Unsupervised NMF labeling	44.5

Table 3.10 Comparison against different labeling method

As seen in Table 3.10, our proposed NMF labeling method can significantly increase the frame-level prediction. A system trained with pseudo strongly labeled data using type-1 and type-2 labeling methods can only achieve a detection accuracy of 30.7% and 34.1%, respectively. In contrast, our proposed NMF labeling method can obtain a detection accuracy of 45.7%, which translates to an increment of up to 15% compared to the two simple labeling methods. Compared to the unsupervised labeling method, our supervised labeling approach also has a higher detection accuracy of 1.2%. Since assuming activated frames to contain all events can induce noise, one may wonder why the difference in accuracy is not as significant as the type-1 and type-2 labeling methods. This is because we propose using two models for SED, one for clip-level prediction and one for frame-level prediction; this alleviates the impact of the assumption.

In order to investigate how much weakly labeled data should be labeled using our proposed NMF labeling method, we varied the amount of weakly labeled data being labeled. It is essential to point out that all remaining weakly labeled data are still being used in the training stage. Since all remaining weakly labeled data are used, they are either labeled using the type-1 or type-2 labeling method. The left column in Table 3.11 and Table 3.12 indicates the percentage of weakly labeled data labeled using our proposed NMF labeling method, while the center column indicates the percentage of weakly labeled data labeled using the type-1 or type-2 labeling method. Naturally, the percentages of the two columns at each row add up to 100%.

Based on the results shown in Table 3.11 and Table 3.12, accuracy tends to increase when more pseudo strongly labeled data labeled using our proposed NMF method were used for training. Thus, one should always use pseudo strong labels for all weakly labeled audio clips to achieve the best possible accuracy.

We then performed an ablation study to investigate the impact of ablating l_{con} and l_{unlabel} on the detection accuracy with and without the use of cyclic learning. As seen in Table 3.13, the ablation of l_{unlabel} can cause the detection accuracy to reduce to 45.1%, and by further ablating l_{con}, the accuracy can further reduce to 43.7%. As seen in Table 3.13,

Amount of proposed NMF labeled data (%)	Amount of type-1 labeled data (%)	Event-based F1-score (%)
0	100	30.7
25	75	40.5
50	50	43.3
75	25	44.6
100	0	**45.7**

Table 3.11 Model trained with pseudo strongly labeled data and weak labeled data labeled with type-1 labeling

Amount of proposed NMF labeled data (%)	Amount of type-2 labeled data (%)	Event-based F1-score (%)
0	100	34.1
25	75	36.8
50	50	40.4
75	25	42.8
100	0	**45.7**

Table 3.12 Model trained with pseudo strongly labeled data and weak labeled data labeled with type-2 labeling

Loss Included	Event-based F1-score (%)	
	With LR Reset	Without LR Reset
l_{f}, l_{c}, l_{con}, l_{unlabel}	**45.7**	44.7
l_{f}, l_{c}, l_{con}	45.1	44.1
l_{f}, l_{c}	43.7	42.3

Table 3.13 Ablation of l_{con} and l_{unlabel}

the ablation of l_{unlabel} resulted in a smaller performance drop compared to the ablation of l_{con}. This is due to w, which is used to regularize l_{unlabel}. Throughout the entire training procedure, w does not stay constant as 1 and varies between 0 to 1 in most training iterations. Thus, the contribution of l_{unlabel} is less significant as l_{con}, and therefore the ablation of l_{unlabel} results in a less significant performance degradation as compared to ablating l_{con}. As seen in Table 3.13, the impact of ablating l_{con} is much higher without the use of learning rate reset. Such results reinforce that learning rate reset can help derive a better solution as it alleviates the impact of removing these two losses. Based on the ablation study, the two losses are considered essential, which helps to raise our system accuracy.

Pooling method in SM	Event-based F1-score (%)
Max	**45.7**
Average	41.3
Linear Softmax	44.3
Exponential Softmax	42.6
Auto (McFee et al., 2018)	43.2
Attention	43.0

Table 3.14 Effects of different pooling approach in SM

We then experimented with different temporal pooling methods in our proposed SM. Based on the results shown in Table 3.14, the max-pooling operator is considered the most optimal approach, while the average pooling is the worst pooling approach. The reason for poor accuracy when using average pooling is due to the manner of gradient propagation.

As explained in (McFee et al., 2018; Wang et al., 2019a), gradient propagation for average pooling will assign equal weightage to all frames. Due to the presence of an event, all frames get updated equally, leading to a large number of false positives, which reduces the detection accuracy (Wang et al., 2019a).

As mentioned in Section 3.3.4 Experiment Setup, we propose using event-specific median filter window sizes and thresholds for post-processing and event detection, which can help increase overall detection accuracy. The optimal value of these window sizes or thresholds can be found by tuning them against the validation dataset. However, the optimal set of event-specific median filter window sizes or thresholds can differ across different systems. We termed the set of parameters that maximize the overall detection accuracy for only one system as the optimal local set of parameters.

In this section, instead of using an optimal local set of parameters, we propose using an optimal global set of parameters. An optimal global set of parameters refers to a set of parameters found to increase (but not maximize) the overall accuracy of multiple systems. In this section, five different parameters were used for post-processing or event detection: 1) event-specific median filter window size used in the first pass, 2) event-specific median filter window size used in the second pass, 3) event-specific frame threshold, 4) lower-bound threshold.

The event-specific median filter window sizes were found using a random search where the constraint set was that window sizes used in the second pass must be larger than the window sizes used in the first pass. We found that using a smaller window size in the first pass of filtering and larger window size in the second pass of filtering usually produced

Event label	Event-based F1-score (%)
Speech	52.6
Dog	25.5
Cat	37.1
Alarm/bell ringing	46.1
Dishes	20.8
Frying	50.9
Blender	48.3
Running water	41.2
Vacuum cleaner	73.8
Electric shaver/toothbrush	60.5

Table 3.15 Classwise event-based F1-score (%) of System 2

higher accuracy. The event-specific frame thresholds were found using a grid search in a range of 0.1 to 1.0 with a step size of 0.1. In contrast, the lower bound threshold was found using a grid search in a range of 0.0 to 0.1 with a step size of 0.01. The benefits of using an optimal global set of parameters are that it avoids the need to perform parameter tuning every time a new system is trained and can avoid overfitting on a specific dataset.

The post-processing method in (Kong et al., 2019a) began with joining similar events before removing noise. However, in our experiment, we found that accuracy can be higher if the noise is removed before concatenating similar events.

One weaker aspect of our framework is the detection accuracy of the dishes. As seen in Table 3.15, the detection accuracy of dishes is only marginally above 20%. As mentioned in (Pellegrini and Cances, 2019), most of the training data for dishes contain other event classes. This can make it difficult for the system to learn the unique characteristic of this event, resulting in low accuracy.

3.3.6. Comparison against SOTA

We proposed using a non-ensembled system and two ensembled systems as a form of comparison against the other SOTAs. The non-ensembled system is chosen as System 2, the best system analyzed in the previous section. The first ensembled system is the ensemble of System 1 and System 2, which are combined by averaging the posterior outputs of both systems while using the same optimal global set of parameters. The second ensembled system uses the same system combination as the first ensembled system. However, median

filter window sizes were tuned further while using the same event-specific frame threshold and lower bound threshold.

We first compared our systems against the top 3 submissions in DCASE 2019 on the validation dataset. Since we are also comparing the accuracy of the ensemble system, the column which indicates "No of systems" refers to how many systems are combined to form an ensemble system. As mentioned earlier, two proposed ensembled systems will be compared against the SOTA; as such, the accuracies of two ensembled systems are separated by a centerline in the following tables under the Ensembled column.

As seen in Table 3.16, even though our non-ensemble system has 2M fewer parameters than the first place ensembled system in DCASE 2019, we can still win them by 0.3%. On the other hand, our ensembled system can have a winning margin of up to 3.2% against the first place ensembled system in DCASE 2019. However, we note that the number of parameters is 3M higher in this case.

Method	Validation		No. of parameters (M)		No. of systems
	Non-ensembled	Ensembled	Non-ensembled	Ensembled	
	Event-based F1-score (%)				
Proposed	45.7	48.0 | 48.6*	5	10	2
DCASE 2019 1st place (Lin et al., 2019)	44.5	45.4	-	7	5
DCASE 2019 2nd place (Delphin-Poulat et al., 2020)	43.6	-	1	-	-
DCASE 2019 3rd place (Shi et al., 2019)	-	42.5	-	6	6

Table 3.16 Comparison against the top 3 submissions from 2019 on the validation dataset. (Note: *Ensembled system with median filter window sizes tuned).

Method	Validation		Evaluation 2020		No. of parameters (M)		No. of systems
	Non-ensembled	Ensembled	Non-ensembled	Ensembled	Non-ensembled	Ensembled	
	Event-based F1-score (%)						
Proposed	45.7	48.0 | 48.6*	44.4	45.8 | 46.3*	5	10	2
DCASE 2020 Baseline (Turpault and Serizel, 2020)	34.8	-	34.9	-	1	-	-
DCASE 2020 Baseline with source separation (Turpault et al., 2020b)	35.6	-	36.5	-	1	-	-
DCASE 2020 1st Place (Miyazaki et al., 2020)	46.0	50.6	-	51.1	2	17	8

Table 3.17 Comparison against the baseline systems and top submission from 2020 on the validation dataset and evaluation 2020 dataset. (Note: *Ensembled system with median filter window sizes tuned).

Method	Augmentation used	Event-based F1-score (%)
Proposed (global obtimal set of parameters)	Gaussian noise injection	45.7
Proposed (local obtimal set of parameters)	Gaussian noise injection	**47.5**
DCASE 2020 1st Place (Miyazaki et al., 2020)	Gaussian noise injection	41.4
DCASE 2020 1st Place (Miyazaki et al., 2020)	Time shift and mixup (Zhang et al., 2017)	46.0

Table 3.18 Augmentations used (non-ensembled system)

Since our systems (ensembled and non-ensembled) were submitted to the DCASE 2020 challenge task 4, we then compare against the top submission in the DCASE 2020 challenge task 4 and the baseline system in the DCASE 2020 challenge task 4 on the validation dataset. As seen in Table 3.17, our system outperforms the baseline system by a considerable margin. A winning margin of over 10% was observed on the validation dataset. In addition, a winning margin of over 7% was observed on the evaluation 2020 dataset.

Compared with the top submission of DCASE 2020, the difference between the non-ensemble system is only 0.3% on the validation set. In contrast, the difference between the ensembled system is 2%. As the non-ensembled system of Miyazaki et al. (2020) was not submitted to the challenge, the comparison on the challenge evaluation dataset can only be made between the ensembled systems. As seen in Table 3.17, a difference of 4.8% can be observed between our proposed ensembled system and the first place ensembled system. It is essential to point out that our system maintained a level of competitiveness across different datasets despite using a much smaller ensemble with fewer parameters (lower number of systems, 2 compared to 8, lower number of parameters 10M compared to 17M).

One interesting detail shown in Table 3.17 is that our proposed ensembled system with median filter tuned does not seem to exhibit overfitting. Instead, it performs better than the other ensembled system, which used the optimal global set of parameters. As the use of a median filter helps to smooth the predicted probabilities, having a non-tuned median filter may not achieve the desired smoothing effect. Thus, it is still necessary to tune the median filter to achieve the best possible results.

We then proceed with the post-challenge analysis. We first investigated the cause of performance differences. Firstly, the leading cause of the difference is the use of an optimal

Ensembled system	Event-based F1-score (%)
Proposed	**39.9**
DCASE 2020 1st Place (Miyazaki et al., 2020)	2.9

Table 3.19 Accuracy on long-duration (60s) dataset (Turpault et al., 2021)

global set of parameters instead of an optimal local set of parameters. As mentioned earlier, one benefit of using an optimal global set of parameters is to avoid overfitting. However, as shown in Table 3.17, no sign of performance degradation can be seen on the proposed ensembled system with median filter window sizes tuned. Thus, using the optimal local set of parameters does not appear to cause any issue of overfitting. We then tuned our proposed non-ensembled system using the validation dataset and obtained the optimal local set of parameters. As seen in Table 3.18, with the optimal local set of parameters, our system can be improved by 2.2% and became 1.5% higher than the first place submission.

The second cause of performance difference is the augmentation techniques used. In our framework, we only attempted the Gaussian noise injection, and from the information showed in (Miyazaki et al., 2020), they used more augmentations such as time-shift and mixup (Zhang et al., 2017). Based on the information shown in Table 3.18, if Miyazaki et al. (2020) only used Gaussian noise injection, our system can perform much better than their system (4.3% to 6.1%).

Finally, we compare the duration robustness of our system against the DCASE 2020 challenge task 4 first-place submission (Miyazaki et al., 2020). Turpault et al. (2021) explained that such a test could be meaningful because, in real-life scenarios, a SED system is more likely to operate on sound segments that are longer than 10s. In addition, when operating in real-life scenarios, a SED system is more likely to face scenarios where the sound event density is much lower than the sound event density in the YouTube videos (AudioSet), which are generally recorded because something is actually happening.

Based on the results seen in Table 3.19, the first place system (Miyazaki et al., 2020) is not duration robust and had an abysmal performance when tested on a subset of audio clips which are 60s long. Miyazaki et al. (2020) system only had an accuracy of 2.1%, while our system had an accuracy of 39.9%, which is 37.9% higher than the first place system. It is essential to point out that the results are obtained from (Turpault et al., 2021) as challenge evaluation data is not released. Such difference indicates that our system is much more robust on long-duration audio clips.

3.3.7. Summary

This chapter discussed the motivation of using pseudo strong labels to train a SED system. The proof of concept indicated that pseudo strong labels are a viable and effective alternative to train a SED system. Although pseudo strong labels estimated using unsupervised NMF can contain a certain amount of noise. However, results indicated that our proposed CNN trained using such data could easily outperform the baseline system. Nevertheless, such an issue can be alleviated by using our proposed student-teacher framework, where the SM will provide the frame-level prediction based on the clip-level prediction provided by the TM.

As our proposed student-teacher framework consists of several different variables and hyperparameters, we have extensively studied different aspects of our proposed framework and established the most optimal variables and hyperparameters to use. Experiments include the investigation of transfer learning, cyclic learning, the effectiveness of the proposed pseudo labeling method, ablation of consistency losses, optimal pooling method for SM.

Through extensive comparison and evaluation, we established that our proposed framework could be competitive with the SOTA. However, one notable weakness is that the classwise accuracy of impulse/short sounds such as Dishes is relatively low compared to the other event classes. In the next chapter, we explore the possibility of integrating the Transformer encoding layer (Vaswani et al., 2017) into the architecture to examine if the detection accuracy can be improved.

Chapter 4. Improved Pseudo Labeling Approach and Integration of Macaron Net

4.1. Supervised CNMF for Pseudo Labeling

In the previous chapter, we established that NMF could be a helpful tool for estimating pseudo strong labels for each weakly labeled clip. In this chapter, we attempt to further improve the quality of the pseudo strong labels through the use of CNMF.

Smaragdis (2007) explained that while NMF is useful for analyzing data, it ignores the potential dependencies across successive columns of its input. The fact that there is a sequence would not be apparent by examining the bases but would only be discovered by careful analysis of the basis weights (Smaragdis, 2007). Thus, CNMF was proposed to resolve this issue, and 2.1 is extended as follow

$$\ddot{\mathbf{V}} \approx \sum_{k=0}^{K-1} \ddot{\mathbf{W}}_k \overset{k\rightarrow}{\ddot{\mathbf{H}}} \tag{4.1}$$

Where $\ddot{\mathbf{V}}$ represents a nonnegative matrix. $\ddot{\mathbf{W}}$ and $\ddot{\mathbf{H}}$ represent the basis and activation matrix, respectively. $\ddot{\mathbf{W}}_k$ represents $\ddot{\mathbf{W}}$ at k-step shift. $\overset{k\rightarrow}{\ddot{\mathbf{H}}}$ represents $\ddot{\mathbf{H}}$ which is shifted k steps. $\rightarrow$ is the shift operator in the right direction. Thus, the operator $\overset{k\rightarrow}{(.)}$ indicates that the column of the argument is shifted to the right by k steps. For each shift, the input argument is padded with vectors of 0 to maintain the same size. For clarity, a simple example is given as follows.

$$\mathbf{x} = \begin{bmatrix} 0 & 1 & 2 & 3 \\ 0 & 1 & 2 & 3 \\ 0 & 1 & 2 & 3 \\ 0 & 1 & 2 & 3 \end{bmatrix}, \overset{0\rightarrow}{\mathbf{x}} = \begin{bmatrix} 0 & 1 & 2 & 3 \\ 0 & 1 & 2 & 3 \\ 0 & 1 & 2 & 3 \\ 0 & 1 & 2 & 3 \end{bmatrix} \tag{4.2}$$

$$\overset{1\rightarrow}{\mathbf{x}} = \begin{bmatrix} 0 & 0 & 1 & 2 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 1 & 2 \end{bmatrix}, \overset{2\rightarrow}{\mathbf{x}} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.3}$$

Whereas for an operator $\overset{\leftarrow k}{(.)}$, it represents that an input argument is shifted to the left by k steps which is the opposite of $\overset{k\rightarrow}{(.)}$. For clarity, a simple example is given as follows.

$$\mathbf{x} = \begin{bmatrix} 0 & 1 & 2 & 3 \\ 0 & 1 & 2 & 3 \\ 0 & 1 & 2 & 3 \\ 0 & 1 & 2 & 3 \end{bmatrix}, \overset{\leftarrow 0}{\mathbf{x}} = \begin{bmatrix} 0 & 1 & 2 & 3 \\ 0 & 1 & 2 & 3 \\ 0 & 1 & 2 & 3 \\ 0 & 1 & 2 & 3 \end{bmatrix} \tag{4.4}$$

$$\overset{\leftarrow 1}{\mathbf{x}} = \begin{bmatrix} 1 & 2 & 3 & 0 \\ 1 & 2 & 3 & 0 \\ 1 & 2 & 3 & 0 \\ 1 & 2 & 3 & 0 \end{bmatrix}, \overset{\leftarrow 2}{\mathbf{x}} = \begin{bmatrix} 2 & 3 & 0 & 0 \\ 2 & 3 & 0 & 0 \\ 2 & 3 & 0 & 0 \\ 2 & 3 & 0 & 0 \end{bmatrix} \tag{4.5}$$

Based on Equation 4.1, the use of CNMF would result in k number of $\ddot{\mathbf{W}}$ but with only one $\ddot{\mathbf{H}}$. Smaragdis (2007) then proposed to update $\ddot{\mathbf{W}}$ and $\ddot{\mathbf{H}}$ as follow

$$\ddot{\mathbf{W}}_{k,t} = \ddot{\mathbf{W}}_{k,t-1} \otimes \frac{\frac{\ddot{\mathbf{V}}}{\tilde{\ddot{\mathbf{V}}}_{t-1}}}{\mathbf{1}^n(\overset{k\rightarrow}{\ddot{\mathbf{H}}}_{t-1}^{\top})} \tag{4.6}$$

$$\ddot{\mathbf{H}}_t = \ddot{\mathbf{H}}_{t-1} \otimes \frac{\ddot{\mathbf{W}}_{k,t-1}^{\top}(\frac{\overset{\leftarrow k}{\ddot{\mathbf{V}}}}{\tilde{\ddot{\mathbf{V}}}_{t-1}})}{\ddot{\mathbf{W}}_{k,t-1}^{\top}\mathbf{1}^m} \tag{4.7}$$

where $\ddot{\mathbf{W}}_{k,t}$ represents $\ddot{\mathbf{W}}$ at k-step shift and time step t. $\ddot{\mathbf{W}}_{k,t-1}$ represents $\ddot{\mathbf{W}}$ at k-step shift and time step $t-1$. $\overset{k\rightarrow}{\ddot{\mathbf{H}}}_t$ represents $\ddot{\mathbf{H}}$ which is shifted k steps at time step, t. $\overset{k\rightarrow}{\ddot{\mathbf{H}}}_{t-1}$ represents $\ddot{\mathbf{H}}$ which is shifted k steps at time step, $t-1$. $\tilde{\ddot{\mathbf{V}}}_{t-1}$ represents an estimated $\ddot{\mathbf{V}}$ at time step $t-1$. $\mathbf{1}^m$ and $\mathbf{1}^n$ are m and n dimensional vectors of one, respectively.

As mentioned earlier, CNMF only produces one $\ddot{\mathbf{H}}$ (i.e., $\ddot{\mathbf{H}}$ is shared across all k), while it is possible to update $\ddot{\mathbf{W}}_k$ and $\ddot{\mathbf{H}}$ at each k, this would result in a biased estimate of $\ddot{\mathbf{H}}$ with the update at $K-1$ dominating over the other updates (Smaragdis, 2007). Thus, in practice it would be better to use the average of $\ddot{\mathbf{H}}$ at all k (Smaragdis, 2004).

Smaragdis (2007) demonstrated that such an extension could allow a better separation of audio mixtures. Therefore, it should also allow a better and more accurate basis matrix to be extracted, which improves the quality of pseudo labeling.

As shown in the earlier chapter, a SED model trained using pseudo strongly labeled data estimated in a supervised manner could be more accurate than pseudo strongly labeled data estimated in an unsupervised manner. We continue to provide pseudo labels in a

supervised manner. Thus, the pseudo labeling process remains similar to the labeling process as described in Chapter 3.3.2, where the only change is in the algorithm used (i.e., instead of NMF, in this chapter, we propose the use of CNMF).

4.2. Macaron Net

4.2.1. Motivation

As discussed in Chapter 2, the most popular hybrid architecture used for SED is the CRNN which can learn filters that are shifted in time and frequency and consider long-term temporal context information (Cakir et al., 2017). However, the sequential nature of RNN can make it difficult for parallel computing (Kong et al., 2020; Vaswani et al., 2017). One possible solution is to replace RNN with a newer SOTA known as the Transformer (Vaswani et al., 2017). A transformer is an architecture that can also consider long-term dependency of sequences and was found to outperform RNN in various domains such as language translation (Vaswani et al., 2017) and speech recognition (Karita et al., 2019).

Such a combination was demonstrated in (Kong et al., 2020); however, it could not outperform CRNN in the SED subtask. On the other hand, the combination of CNN with a variant of Transformer known as the Conformer (Gulati et al., 2020), proposed by Miyazaki et al. (2020), could obtain first-place in the DCASE 2020 challenge task 4. Such results lead us to believe that Transformer variants can be better than vanilla Transformer, and in this chapter, we propose combining CNN with Macaron Net (Lu et al., 2019).

4.2.2. Preliminaries of Transformer and Macaron Net

Before describing Macaron Net, we first provide the preliminaries of Transformer (Vaswani et al., 2017). The entire architecture of a Transformer is illustrated in Figure 4.1. As seen in Figure 4.1, there are two shaded boxes; the enclosed components in the Blue box made up a module known as the encoding layer, whereas the enclosed components in the Gray box made up another module known as the decoding layer. Depending on the application, a Transformer can be made up of several encoding layers and several decoding layers.

As seen in Figure 4.1, both the encoding and decoding layers require an input or output. We use a straightforward translation task to illustrate why both the encoding and decoding layers require an input. Given that the task is to translate an English sentence into a Japanese sentence, the encoding layer would take the text in the original language (i.e., English). In contrast, the decoder would take the text in the target language (i.e., Japanese). Thus, given “Thank You” as the input, the output should be “Arigato” (refers to “Thank

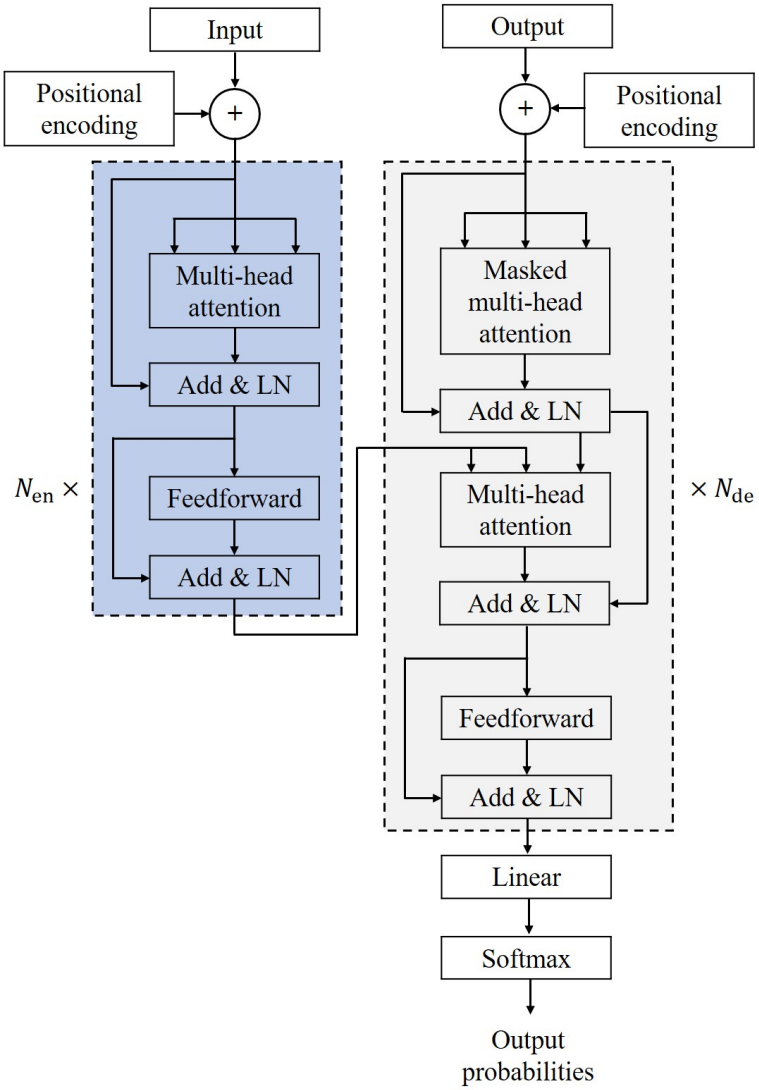


Figure 4.1 Architecture of a Transformer (Vaswani et al., 2017)

You" in Japanese). The output probabilities from the decoding layer would then contain the probabilities of different Japanese words given "Thank You". Ideally, "Arigato" should have a probability of 1.

However, in the SED domain, usually, only the encoding layer is utilized (Kong et al., 2020; Miyazaki et al., 2020). Therefore, as seen in Figure 4.1, one only needs to be concerned with the left side of the Transformer.

If a CNN is stacked over the encoding layer, the input of the encoding layer would naturally represent the output from the last convolutional layer. As Transformer contains no recurrent and convolutional operations, it does not contain the sequence information of an input (Vaswani et al., 2017). Thus, a positional encoding module is used to inject sequence information of the input. Given that $\mathbf{x}_{\text{FM}}$ represents the feature maps from the last convolutional layer and $\hat{\mathbf{x}}$ represents the resulting output after the injection of positional information. $\hat{\mathbf{x}}$ can be represented as

$$\hat{\mathbf{x}} = \mathbf{x}_{\text{FM}} + \mathbf{PE} \tag{4.8}$$

where **PE** represents the positional encoding and is defined as (Vaswani et al., 2017)

$$\mathbf{PE}_{pos,d} = \begin{cases} \sin(pos/10000^{2d/d_{\text{model}}}) & \text{if } d \text{ is even} \\ \cos(pos/10000^{2d/d_{\text{model}}}) & \text{if } d \text{ is odd} \end{cases} \tag{4.9}$$

Where pos is the position and d is the dimension. d_{model} represents the dimension of model and is equal to the output dimension of the last convolutional layer. Based on Equation 4.9, each dimension of the **PE** would correspond to a sinusoid (Vaswani et al., 2017).

For better clarity, we use a simple example of a 5 by 4 **PE**. Based on the image shown in Figure 4.2, the first row would correspond to the first dimension while the first column would corresponding to the first position. Thus, the odd dimension of the **PE** (highlighted in Blue) would use the first equation in Equation 4.9 to calculate the position information. In contrast, the even dimension of the **PE** (highlighted in Orange) would use the second equation in Equation 4.9 to calculate the position information. By referring to Figure 4.1, after obtaining $\hat{\mathbf{x}}$, the next step is to pass $\hat{\mathbf{x}}$ into the encoding layer. Given that $\mathbf{Y}_{\text{en}}$ represents the output from the encoding layer, the operations conducted in the encoding layer to derive $\mathbf{Y}_{\text{en}}$ can be mathematically expressed using the following equations.

$$\bar{\mathbf{x}} = \text{LN}(\hat{\mathbf{x}} + \text{MHA}(\hat{\mathbf{x}})) \tag{4.10}$$

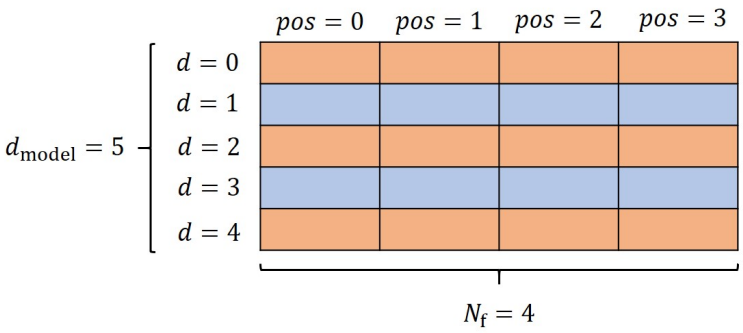


Figure 4.2 Example of a **PE**

$$\mathbf{Y}_{\text{en}} = \text{LN}(\bar{\mathbf{x}} + \text{FF}(\bar{\mathbf{x}})) \tag{4.11}$$

MHA represents the multi-head attention module which contains the following operations.

$$\text{Multihead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Concat}(\mathbf{H}_1, \cdots, \mathbf{H}_{N_\text{h}})\mathbf{W}^\text{O} \tag{4.12}$$

$$\mathbf{H}_i = \text{Attention}(\mathbf{Q}\mathbf{W}_i^\text{Q}, \mathbf{K}\mathbf{W}_i^\text{K}, \mathbf{V}\mathbf{W}_i^\text{V}) \tag{4.13}$$

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_\text{k}}})\mathbf{V} \tag{4.14}$$

where $\mathbf{Q}$, $\mathbf{K}$, $\mathbf{V}$ represent query, key and value which in this application, all three represent $\hat{\mathbf{x}}$. N_h represents the number of heads and $\mathbf{H}_i$ represents the i head. $\mathbf{W}_i^\text{Q}$, $\mathbf{W}_i^\text{K}$ and $\mathbf{W}_i^\text{V}$ represent the parameter matrices for the i-th head and $\mathbf{W}^\text{O}$ represents the final parameter matrix applied on the concatentated feature vector. Finally, d_k represents the dimension of the key.

On the other hand, FF represents the positionwise feedforward module and performs the following operation.

$$\text{FF}(\hat{\mathbf{x}}) = \mathbf{W}_2(f(\mathbf{W}_1\hat{\mathbf{x}} + \mathbf{b}_1)) + \mathbf{b}_2 \tag{4.15}$$

where $\mathbf{W}_1$, $\mathbf{b}_1$, $\mathbf{W}_2$, $\mathbf{b}_2$ represent the learnable parameters of the first and second FFN. $f(.)$ represents the activation function. In the SED domain, $\mathbf{Y}_\text{en}$ is then subsequently passed to other modules to obtain the frame-level probabilities. However, Lu et al. (2019) suggested that the use of only one FF module in the encoding layer can bring bias and lead to higher local truncation error. Nevertheless, this can be mitigated if two FF modules with half-step residual connection are used, thus leading to the proposal of the Macaron Net encoding layer. The difference between the two encoding layers can be seen in Figure 4.3. For a

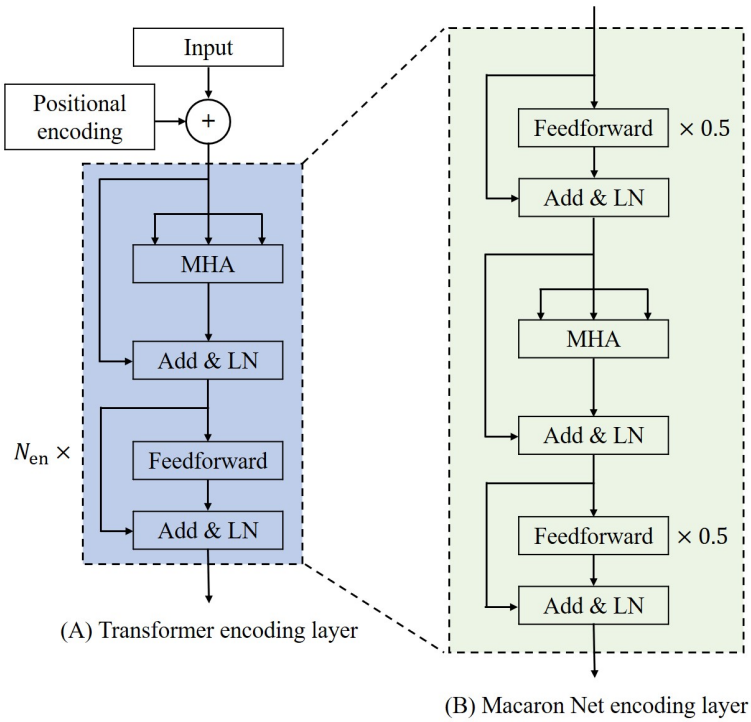


Figure 4.3 Difference between a Transformer encoding layer (Vaswani et al., 2017) and a Macaron Net encoding layer (Lu et al., 2019)

Macaron Net encoding layer, $\mathbf{Y}_{\text{en}}$ can be obtained through the following equations.

$$\tilde{\mathbf{x}} = \text{LN}(\hat{\mathbf{x}} + \text{FF}(\hat{\mathbf{x}})) \tag{4.16}$$

$$\bar{\mathbf{x}} = \text{LN}(\tilde{\mathbf{x}} + \text{MHA}(\tilde{\mathbf{x}})) \tag{4.17}$$

$$\mathbf{Y}_{\text{en}} = \text{LN}(\tilde{\mathbf{x}} + \text{FF}(\tilde{\mathbf{x}})) \tag{4.18}$$

Subsequent experiments conducted by Lu et al. (2019) then indicated that the Macaron Net could achieve higher accuracy than the Transformer on different tasks, which motivates us to use it in the SED domain.

4.3. Experimentation using CNMF and Macaron Net

In this section, experimentation using CNMF and Macaron Net encoding layer is conducted. The primary focus is to identify if CNMF can provide better pseudo labels and how will the system accuracy be affected if a Macaron Net encoding layer is added to the SM and TM. The secondary focus of the experimentation is to investigate ways to improve the utilization of unlabeled data, to which we propose the use of curriculum consistency cost and interpolated consistency training. The subsequent subsections will describe the relevant components in detail.

The experimentation results were published as workshop paper; Chan, T. K., and Chin, C. S. (2021). Detecting Sound Events Using Convolutional Macaron Net With Pseudo Strong Labels. In *Proceeding of the IEEE 23rd Workshop on Multimedia Signal Processing*, Tampere, Finland.

4.3.1. Proposed Semi-Supervised Learning Framework

Based on the models described in Section 3.3.3, the SM will provide the frame-level prediction, and the TM will provide the clip-level prediction. The positional encoding and Macaron Net encoding layer are added to the last convolutional layer in both models. Details of the models can be seen in Figure 4.4 and Figure 4.5. In addition, we propose

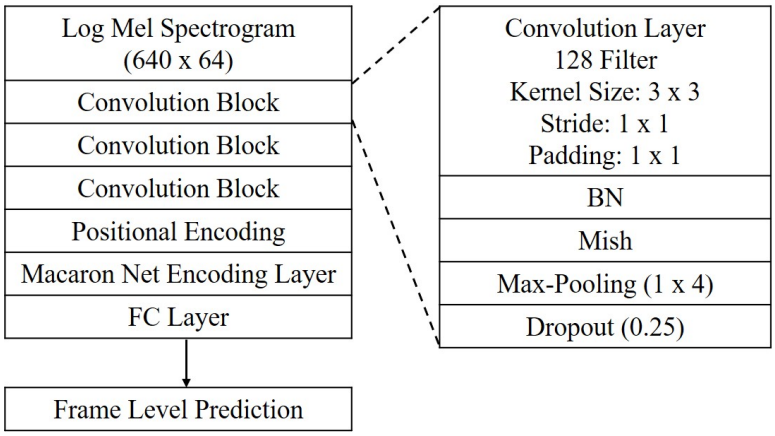


Figure 4.4 SM with Macaron Net (Lu et al., 2019) for frame-level prediction

the use of Mish (Misra, 2019) as the activation function for our models. Unlike ReLU, Mish is continuously differentiable. Such a characteristic is preferable because it avoids singularities and undesired side effects when performing gradient-based optimization. Moreover, Mish was shown to be better in terms of performance and stability than the other activations functions across different image-related tasks (Misra, 2019). Mish can be defined as (Misra, 2019)

$$f(\mathbf{x}) = \mathbf{x} \cdot \tanh(\text{softplus}(\mathbf{x})) \tag{4.19}$$

where tanh represents hyperbolic tangent and softplus is defined as

$$\text{softplus}(\mathbf{x}) = \ln(1 + e^{\mathbf{x}}) \tag{4.20}$$

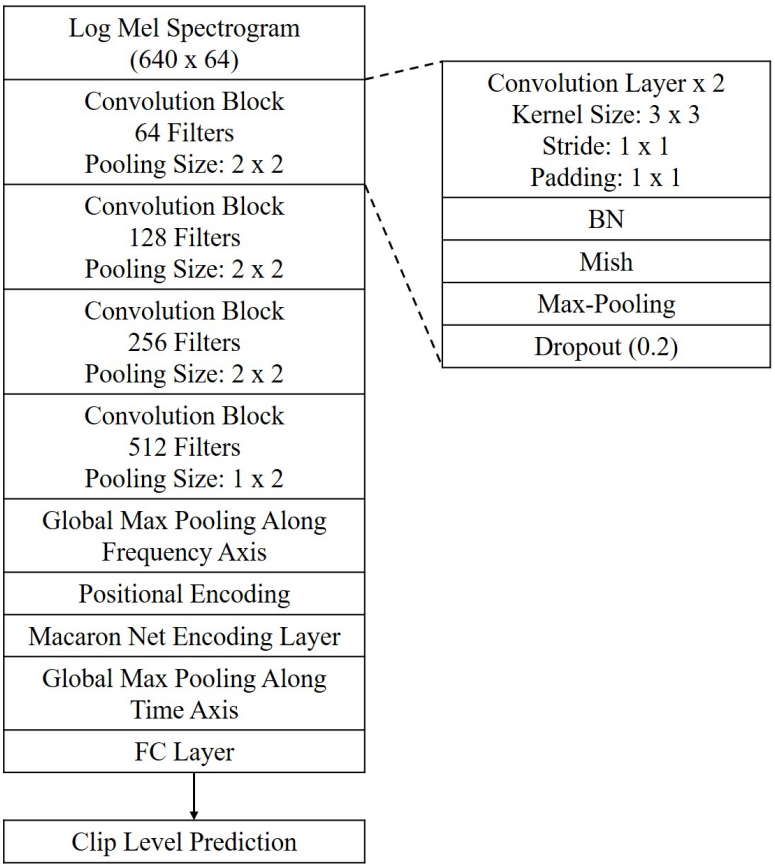


Figure 4.5 TM with Macaron Net (Lu et al., 2019) for clip-level prediction

Based on the student-teacher framework proposed in Section 3.3.3, four loss components are introduced. The first loss component is the frame-level loss, l_{f}, defined as

$$l_{\mathrm{f}} = \frac{1}{N_{\mathrm{e}} \times N_{\mathrm{f}}} \sum_{i=1}^{N_{\mathrm{e}}} \sum_{j=1}^{N_{\mathrm{f}}} [g_{i,j} \log(S_{i,j}) + (1 - g_{i,j}) \log(1 - S_{i,j})] \tag{4.21}$$

where $S_{i,j}$ represents the SM's predicted probability of event i at frame j and $y_{i,j}$ represents the ground truth of event i at frame j. The second loss component is clip-level loss, l_c, defined as

$$l_{\mathrm{c}} = \frac{1}{N_{\mathrm{e}}} \sum_{i=1}^{N_{\mathrm{e}}} [z_i \log(T_i) + (1 - z_i) \log(1 - T_i)] \tag{4.22}$$

where T_{i} represents the TM's predicted probability of event i in an audio clip and z_{i} represent the grouth truth of event i in the same audio clip.

Similar to the earlier framework, we enforce the prediction of SM to be consistent with TM. As observed in Chapter 3.3.5, if TM has yet to achieve optimal detection capability (i.e., low audio tagging accuracy at the early training stage), SM may be forced to produce incorrect predictions at the early stage if the threshold is low. Such enforcement can lead to performance degradation. However, setting a high threshold throughout the training phase may restrict the consistency loss to be calculated on a small number of training examples.

Therefore, instead of using a constant threshold to regulate the consistency loss, we propose a curriculum consistency loss where the confidence threshold will be adjusted according to the learning stage. More specifically, we propose to adjust the confidence threshold from a high value to a lower value. Such implementation is based on the observation in (Mangalam and Prabhu, 2019) where DNN begins learning the easier example before moving to a more difficult example.

Consider that the predicted probability of an event occurring in an easier example should be higher than the predicted probability of an event occurring in a difficult example. Thus, adjusting the confidence threshold from a high value to a lower value restricts the models to consider the high confidence (lower difficulty) examples first before the low confidence (higher difficulty) examples. Such implementation avoids the scenario that SM has to learn from TM when TM has yet to achieve optimal detection accuracy on all examples. At the same time, it allows more training examples to be included for loss calculation, which we hypothesize can help the model increase its accuracy. As the prediction output of SM is in frame level, we apply a global max pooling on the time axis of SM's prediction to obtain the clip level prediction. Thus, the consistency loss, l_{con}, is given as

$$l_{\text{con}} = \begin{cases} \frac{1}{N_{\text{e}}} \sum_{i=1}^{N_{\text{e}}} (S_i^{\text{c}} - T_i)^2 & \text{if } \max(\mathbf{T}) > \lambda_{\text{curr}} \\ 0 & \text{otherwise} \end{cases} \tag{4.23}$$

where S_i^{c} and T_i represent the SM's and TM's predicted probability of event i in an audio clip and $\mathbf{T}$ represents the vector containing the TM's predicted probabilities of all events present in an audio clip. λ_{curr} is the current confidence level used to regulate the loss. The definition of λ_{curr} will be discussed in detail in the experiment setup.

The fourth loss component is the interpolated consistency cost (Verma et al., 2019) between the two models' predictions on the unlabeled data. The purpose of this loss is driven by the theoretical analysis provided in (Verma et al., 2019). Verma et al. (2019) explained that interpolated consistency training corresponds to a certain type of data-adaptive regularization with unlabeled points, reducing overfitting to labeled points under high confidence values.

Given that $\mathbf{U}_1^{\text{A}}$ and $\mathbf{U}_2^{\text{A}}$ represent the augmented feature representation of unlabeled sample 1 and 2. The interpolated consistency cost, l_{i}, can be defined as

$$\tilde{\mathbf{S}}^{\text{mc}} = M(\text{SM}(\text{mixup}(\mathbf{U}_1^{\text{A}}, \mathbf{U}_2^{\text{A}}))) \tag{4.24}$$

$$\tilde{\mathbf{T}}^{\text{m}} = \text{mixup}(\text{TM}(\mathbf{U}_1^{\text{A}}), \text{TM}(\mathbf{U}_2^{\text{A}})) \tag{4.25}$$

$$l_{\mathrm{i}} = \frac{1}{N_{\mathrm{e}}} \sum_{i=1}^{N_{\mathrm{e}}} (\tilde{S}_i^{\mathrm{mc}} - \tilde{T}_i^{\mathrm{m}})^2 \tag{4.26}$$

where $\tilde{\mathbf{S}}^{\mathrm{mc}}$ represents the vector that contains the events' probabilities from SM on the mixed unlabeled sample. $M(.)$ represents the max pooling operator. $\mathrm{mixup}(\mathbf{U}_1^{\mathrm{A}}, \mathbf{U}_2)^{\mathrm{A}}$ is defined as (Zhang et al., 2017)

$$\mathrm{mixup}(\mathbf{U}_1^{\mathrm{A}}, \mathbf{U}_2^{\mathrm{A}}) = \psi \mathbf{U}_1^{\mathrm{A}} + (1 - \psi)\mathbf{U}_2^{\mathrm{A}} \tag{4.27}$$

where ψ is the mixing factor in a range of 0 to 1 and is generated from a beta distribution (i.e., $\mathrm{Beta}(0.2, 0.2)$). $\tilde{\mathbf{T}}^{\mathrm{m}}$ represents the vector that contains the interpolated events' probabilities which are mixed using the probabilities of the predicted events in $\mathbf{U}_1^{\mathrm{A}}$ and $\mathbf{U}_2^{\mathrm{A}}$. $\tilde{S}_i^{\mathrm{mc}}$ and $\tilde{T}_i$ is the i element in $\tilde{\mathbf{S}}^{\mathrm{mc}}$ and $\tilde{\mathbf{T}}^{\mathrm{m}}$ which represent the probability of event i. Based on the formulation given in Equation 4.26, l_{i} is the MSE between $\tilde{\mathbf{S}}^{\mathrm{mc}}$ and $\tilde{\mathbf{T}}^{\mathrm{m}}$.

However, we only allow l_{i} to be calculated if interpolated probabilities from TM exceeds the confidence threshold. Thus, l_{i} is extended as a regularized interpolated consistency cost, l_{ri} which can be defined as

$$l_{\mathrm{ri}} = \begin{cases} \frac{w}{N_{\mathrm{e}}} \sum_{i=1}^{N_{\mathrm{e}}} (\tilde{S}_i^{\mathrm{mc}} - \tilde{T}_i^{\mathrm{m}})^2 & \text{if } \max(\tilde{\mathbf{T}}^{\mathrm{m}}) > \lambda_{\mathrm{curr}} \\ 0 & \text{otherwise} \end{cases} \tag{4.28}$$

where w is an additional weighing parameter which is also used in our previous framework in Chapter 3.3.3. w is defined as (Laine and Aila, 2017)

$$w = \exp(-5(1-P)^2) \tag{4.29}$$

where P is a positive value representing the training progression and will be discussed in more detail in the next subsection.

4.3.2. Experiment Setup

The dataset used for the subsequent experiments is the DESED 2020 dataset (Turpault et al., 2019), and the data distribution can be found in Table 3.6. As the methodology was not submitted to the DCASE challenge, results on the evaluation 2020 dataset cannot be obtained. Thus, results are only reported on the validation dataset.

Audio preprocessing and feature extraction remains similar, as described in Chapter 3.3.4. As mentioned in Chapter 4.1, the pseudo labeling process remains similar to the

labeling process described in Chapter 3.3.2. The only change is in the algorithm used (i.e., NMF is replaced with CNMF).

We began our experiment using only 1 layer of Macaron Net encoding layer with 4 heads and the models are trained in 2 different phases; 1) warm-up phase and 2) adaptation phase. In the first phase, models are trained using only synthetic and pseudo strongly labeled data, where they are augmented with Gaussian noise, time mask, and frequency mask. A batch size of 32 is used and is evenly split between the two data types.

In the warm-up phase, models are trained with an increasing learning rate. The calculation of the learning rate at each iteration is modified from Equation 3.10 and is given as

$$LR_{\text{curr}} = LR_{\text{min}} + 0.5(LR_{\text{max}} - LR_{\text{min}})(1 + \cos(\frac{P_{\text{w}} - P_{\text{curr}}}{P_{\text{w}}}\pi)) \tag{4.30}$$

where LR_{max} and LR_{min} are set as 0.0014 and 1e-6, respectively. P_{w} represents the total iterations during the warm-up phase which we set as the total number of iterations in 10 epochs. As unlabeled data is not utilized in this phase, the total loss is the summation of l_{f}, l_{c} and l_{con}. In the warm-up phase, we do not apply the curriculum consistency cost as such, λ_{curr} is set as 0.9 throughout the warm-up phase. Based on the losses calculated, models are updated using Lookahead (Zhang et al., 2019) with an alpha of 0.5 and step size of 20 with Adam (Kingma and Ba, 2015). As mentioned earlier, P is a positive value representing the training progression. Thus, P is defined as

$$P = \frac{P_{\text{w}} - P_{\text{curr}}}{P_{\text{w}}} \tag{4.31}$$

In the adaptation phase, models are trained with all types of data. Augmentation techniques remain the same for synthetic and unlabeled data, while unlabeled data is only augmented with Gaussian noise injection. A batch size of 64 is utilized where half of them is a mixture of synthetic and pseudo strongly labeled data while the other half is the unlabeled data. The calculation of the learning rate at each iteration is similar as Equation 3.10 and is given as (Loshchilov and Hutter, 2017)

$$LR_{\text{curr}} = LR_{\text{min}} + 0.5(LR_{\text{max}} - LR_{\text{min}})(1 + \cos(\frac{P_{\text{curr}}}{P_{\text{tot}}}\pi)) \tag{4.32}$$

LR_{max} and LR_{min} remains as 0.0014 and 1e-6. As cyclic learning scheme is not applied, P_{tot} is simply set as the total number of iterations in 100 epochs. In this phase, λ_{curr} does not remain fixed and is annealed using the same cosine function used to calculate LR_{curr} in

Equation 3.10 and Equation 4.32 (Loshchilov and Hutter, 2017). Thus λ_{curr} is defined as

$$\lambda_{\text{curr}} = \lambda_{\text{min}} + 0.5(\lambda_{\text{max}} - \lambda_{\text{min}})(1 + \cos(\frac{P_{\text{curr}}}{P_{\text{tot}}}\pi)) \tag{4.33}$$

where λ_{max} and λ_{min} represent the maximum and minimum confidence level and is set as 0.9 and 0.6, respectively. Thus, λ_{curr} will slowly decrease along the cosine curve from 0.9 to 0.6. It is important to point out that although the cosine function is designed initially to anneal the learning rate, the equation is only bounded by the maximum and minimum value, thus can be easily adapted and applied to other applications. Based on the losses calculated, models are updated using Lookahead (Zhang et al., 2019) with an alpha of 0.5 and step size of 20 with Adam (Kingma and Ba, 2015). P is then given as

$$P = \frac{P_{\text{curr}}}{P_{\text{tot}}} \tag{4.34}$$

In the inference stage, both models are used for audio tagging and SED. The method of detection and post-processing steps remain largely similar, as described in Chapter 3.3.4. We consider an event to be present if the clip-level prediction by TM exceeds 0.5. The corresponding frame-level prediction is then smoothed using the median filter with event-specific window size. Frames are considered activated if they exceed the event-specific threshold, and neighboring frames are considered to be activated if they exceed the event-specific lower bound threshold. Note that these event-specific parameters are tuned using the validation dataset. Subsequently, events with a duration shorter than 0.1s are removed. Finally, similar events are joined together if the difference between the offset of the first event and the onset of the second event is smaller than 0.2s.

In this section, all experiments were conducted on a system using an Intel Processor i7-10875H with a base frequency of 2.3GHz, 32GB ram and a RTX3070 GPU.

4.3.3. Results and Discussion

In this section, several different experiments are conducted on the validation dataset to investigate different aspects of the proposed system, and accuracy is measured using only the event-based F1-score (Mesaros et al., 2016). As mentioned earlier, the primary focus is to identify if CNMF can provide better pseudo labels and how will the system accuracy be affected if a Macaron Net encoding layer is added to the SM and TM. In contrast, the secondary focus of the experimentation is to investigate ways to improve the utilization

of unlabeled data. Thus experiments are planned according to the primary and secondary focus.

Setting	Event-based F1-score (%)
Pseudo labeling using CNMF	**46.3**
Pseudo labeling using NMF	45.0

Table 4.1 Accuracies of systems trained using different pseudo strongly labeled data

We first examine if there is any performance gain due to a change in the pseudo labeling algorithm. Results shown in Table 4.1 indicate that a system trained with pseudo strongly labeled data estimated using CNMF can obtain a higher accuracy of 1.3% compared to a system trained with pseudo strongly labeled data estimated using NMF. Such results infer that CNMF is a better approximator; however, due to the shift operation in CNMF, the formation of event dictionaries and the pseudo labeling process can be longer than the use of NMF. This can be made worse when using a sizeable time-shift value such as 100. Our preliminary studies compared several time-shift values and found that a time shift value of 10 works reasonably well without a significant increase in computation time. Considering that the pseudo labeling process only takes place once, CNMF should replace NMF as the pseudo labeling tool.

Warm-up epochs	Event-based F1-score (%)
0	43.9
5	44.4
10	**46.3**
20	44.4

Table 4.2 System accuracy with different warm-up epochs

We then investigated the importance of warm-up, which is considered a critical component to train a Transformer (Vaswani et al., 2017). As seen in Table 4.2, while models do not fail to converge, models can benefit from warm-up, which brings a maximum accuracy increment of 2.4%.

Subsequently, an ablation study is done to examine the importance of positional encoding, which injects information about the relative position of an input sequence. Based on the results tabulated in Table 4.3, it appears that the ablation of positional encoding only results in a very marginal drop in accuracy. But since the computational overhead

Setting	Event-based F1-score (%)
With positional encoding	**46.3**
Without positional encoding	46.2

Table 4.3 Importance of positional encoding

of including positional encoding is negligible, one can still include such a module to maximize the accuracy.

Setting	Event-based F1-score (%)
Proposed-Mish	**46.3**
Proposed-ReLU	43.6
Proposed-Vanilla Transformer	42.5

Table 4.4 Architecture accuracy using different settings

We then compared the use of ReLU against Mish (Misra, 2019), and the results in Table 4.4 show that Mish does outperform ReLU. We then replace the Macaron Net encoding layer with a vanilla Transformer encoder layer. Results in Table 4.4 indicate that a system utilizing a vanilla Transformer encoder layer performs poorly against the system using the Macaron Net encoder layer.

We then varied the number of attention heads and layers of the encoder layer. As seen in Table 4.5, using a single layer with 4 heads is sufficient, and any more can degrade the accuracy. We hypothesize that increasing the number of heads and layers can make the models more challenging to train, which results in a decrease in accuracy.

		Layer					
		1	2	3	4	5	6
N_{h}	4	**46.3**	45.2	44.8	42.3	43.6	44.2
	8	45.1	44.9	43.2	42.5	41.9	42.1
	16	45.4	44.5	44.9	43.0	42.8	44.0
	32	45.1	44.1	44.3	42.9	43.3	42.9

Table 4.5 System accuracy with different numbers of encoding layer and head

Loss included	Event-based F1-score (%)
l_{f}, l_{c}, l_{con}, l_{ri}	**46.3**
l_{f}, l_{c}, l_{con}	45.8
l_{f}, l_{c}	44.9

Table 4.6 Ablation of l_{con} and l_{ri}

An ablation study was subsequently performed to investigate the importance of l_{con} and l_{ri}. As seen in Table 4.6, the two losses are critical, and ablating them would result in an accuracy drop, although the impact of ablating l_{ri} is lower due to the regularizing term, w. Such results coincide with our previous study in Chapter 3.3.5, where the ablation studies exhibit consistent behavior and show that consistency losses are critical components to improve the frame-level prediction.

Curriculum consistency losses	Constant λ_{curr} of 0.9
46.3	44.9

Table 4.7 Comparison of accuracy with and without curriculum consistency losses

We then compare the use of curriculum consistency losses against consistency losses with a constant confidence threshold of 0.9. As seen in Table 4.7, the use of curriculum consistency losses can improve the system accuracy by 1.4%, which shows the importance of including not just the confident prediction during the training process.

λ_{min}	Event-based F1-score
0	43.6
0.2	45.2
0.4	45.3
0.6	**46.3**
0.8	44.3

Table 4.8 Effect of λ_{min} on accuracy

Since varying the threshold value is better than a fixed threshold, we investigate on the optimal λ_{min} value. As seen in Table 4.8, while it is important to include lesser

		Step size			Ablate Lookahead
		5	10	20	
Alpha	0.1	44.7	45.6	45.5	44.8
	0.5	45.5	45.2	**46.3**	
	1	44.9	45.3	45.8	

Table 4.9 Parameter analysis for Lookahead

confident examples into the losses calculation but it can have a negative impact if λ_{min} is set lower than 0.6. On the other hand, if λ_{min} is set too high (i.e., 0.8), it does not bring any performance gain.

We then investigate on the effect of using different alpha and step size for Lookahead (Zhang et al., 2019). Zhang et al. (2019) explained that Lookahead reduces the variance and is less sensitive to suboptimal hyperparameters and can reduce the need for extensive hyperparameter tuning. However, as shown in Table 4.9, we find that there is still a need to tune alpha and step size. As shown in Table 4.9, while Lookahead provides accuracy improvement, improvement will only be marginal if suboptimal parameters are used.

4.3.4. Comparison against SOTA

Methodology	Event-based F1-score
Proposed	**46.3**
(1st in DCASE 2020) CNN+Conformer (Miyazaki et al., 2020)	46.0
(1st in DCASE 2019) CNN (Lin et al., 2019)	44.5
Baseline with source separation (Turpault et al., 2020b)	35.6

Table 4.10 Effect of λ_{min} on accuracy

We then compared against the non-ensembled systems from the top submission in DCASE 2019, DCASE 2020 Challenge task 4, and the baseline system (Turpault et al., 2020b). As seen in Table 4.10, our system outperforms the baseline by a margin of over 10% and also the 1st place submission in DCASE 2019 by 1.8%. At the same time, we also show that using a single layer four head Macaron Net encoding layer is sufficient to produce comparable results as the 1st place submission in DCASE 2020, which utilized a

four layers four heads Conformer encoding layer. Such results showcase the potential of CNN-Macaron Net architecture in the frame-level prediction.

4.4. Multi-branch Convolutional Macaron Net for SED

In Chapter 3.3.3, a novel student-teacher framework is proposed where the SM will provide the frame-level prediction while the TM will provide the clip-level prediction. While such a framework can be competitive against SOTA, it has one weakness: the design of two completely different models. The need to tune two completely different models to achieve optimal results can be very time-consuming.

As seen in the previous section, the frame-level prediction can be improved with the inclusion of a Macaron Net encoding layer. However, in our preliminary experiments, we found that the inclusion of a Macaron Net encoding layer to the TM can also reduce the accuracy of the clip-level prediction.

These two issues call for a need to rethink our student-teacher framework, which should avoid extensive model tuning and at the same time improve a SED system as a whole rather than a single subtask (i.e., audio tagging or temporal localization). As such, we extend our proposed student-teacher framework by incorporating several changes and improvements.

1. Instead of designing two completely different models, we develop two slightly different networks with the inclusion of a new SOTA activation function called meta-ACtivate Or Not (meta-ACON) (Ma et al., 2021) and Squeeze and Excite (SE) module (Hu et al., 2020). The less complex model will provide the frame-level prediction, while the more complex model will give the clip-level prediction. The only differences between the two models lie in the number of convolutional layers, the temporal compression size, and the temporal pooling methods that produce the clip-level prediction. This minimizes the time to tune two completely different models.

2. We propose an improved Macaron Net encoding layer that adopts the Pre-Norm arrangement with additional FC layers. This allows us to train the models using only a 1-layer 1-head Macaron Net encoding layer without any performance degradation.

3. We propose a triple instance-level pooling approach, also known as multi-branch pooling, to our TM model to improve the clip-level prediction. This allows the model to learn unique characteristics from each branch and improve the overall performance (Huang et al., 2020b). Rather than using a combination of one main and multiple

auxiliary branches as suggested in (Huang et al., 2020a,b), our multi-branch pooling method weighs each branch equally and is used in both the training and inference stage. Not only does the multi-branch pooling improves the clip-level prediction, but it also forces the SM model to learn from a more complex model, which in turn improves the frame-level prediction.

4. We propose an improved cyclic learning scheme. Rather than an immediate learning rate reset as proposed in (Loshchilov and Hutter, 2017), we slowly transit the learning rate to the maximum, which is shown to increase the model accuracy. As explained in (Smith, 2017), such a learning scheme can allow a more rapid traversal of saddle point plateaus.

Based on the proposed improvements, our SED system can achieve an event-based F1 score of 48.5%. By ensembling the top 5 models by averaging the posterior probabilities, we can further raise the event-based F1-score to 50.4%. Such a result allows us to have a minimum margin of 12% against the baseline system and only has a 0.2% difference with the 1st place submission of DCASE 2020. The following subsections then provide an in-depth description of the proposed improvements.

This section was summarized and published as a journal paper; Chan, T. K., and Chin, C. S. (2021). Multi-branch convolutional macaron net for sound event detection. *IEEE/ACM Transactions on Audio, Speech and Language Processing*, 29:2972-2985.

4.4.1. Improved Models Architecture

Using Figure 4.6 as a reference, the SM model only has one convolutional block before the pooling layer, whereas the TM model has two convolutional blocks. In addition, the SM model does not pool along the time axis, unlike the TM model. Thus, the size of the feature map after each pooling layer is different for each model. We hypothesized that the clip-level prediction performance could be better by allowing TM model to have more convolution layers and more temporal compression.

Figure 4.7A and Figure 4.7B then illustrate the functions applied in the 2-dimensional and 1-dimensional convolutional blocks. Figure 4.7C illustrates how the SM model produces the frame-level and clip-level prediction, while Figure 4.7D explains how the AT model produces the clip-level prediction.

As seen in Figure 4.7D, the output layer of TM model is slightly different from the SM model as it does not produce the frame-level prediction and adopted a multi-branch

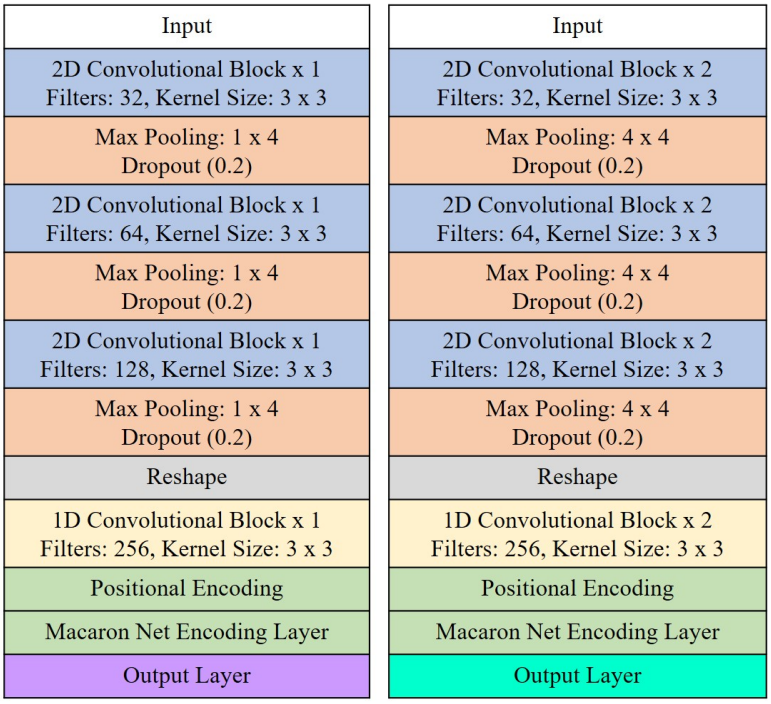


Figure 4.6 Improved SM and TM

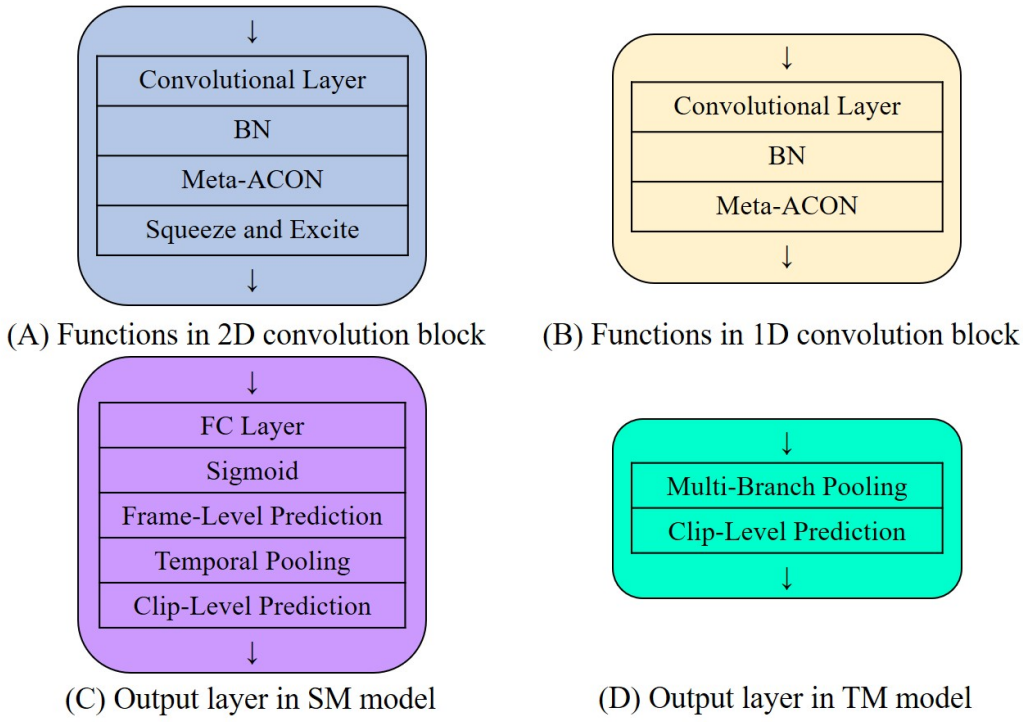


Figure 4.7 Modules description

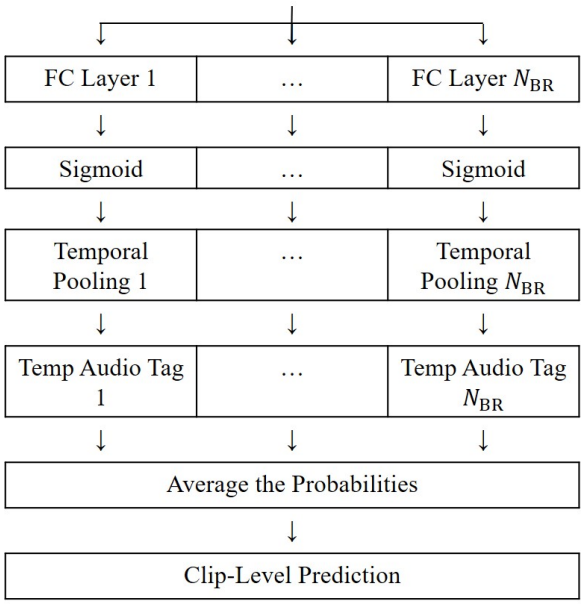


Figure 4.8 Multi-branch pooling approach

pooling scheme (which can be seen in Figure 4.8 where N_{BR} represents the total branches) to obtain the clip-level prediction.

Based on our models' architecture, the number of parameters used in the SM model and TM model would be approximately 2M and 2.5M, respectively. Thus, the total number of model parameters is approximately 4.5M. If the Mean-Teacher approach is used to train the TM, the total number of parameters used would be 5M, which is approximately 10% higher. One may argue that the total number of parameters can be lesser if one chooses to train the SM instead of the TM using the Mean-Teacher approach. However, as mentioned earlier, one model may not be designed optimally for both frame-level prediction and clip-level prediction. As compared to the Mean-Teacher approach, our framework does not require two identical models to be trained synchronously and thus offers more flexibility in the models' design. This can allow two different models to be optimal at different subtasks (i.e., audio tagging and temporal localization) and may offer parameter reduction.

Based on the modules shown in Figure 4.6 and Figure 4.7, there are four modules that we consider crucial and critical, 1) meta-ACON, 2) the SE module, 3) Macaron Net encoding Layer, and 4) the multi-branch pooling scheme. These modules will be discussed in the following subsections.

4.4.2. Meta-ACON and SE Module

Meta-ACON is a new activation function theorized in (Ma et al., 2021) that is a simple and effective method that learns to activate neurons or not. By allowing each neuron to adap-

tively activate or not, such customized activating behavior can help improve generalization (Ma et al., 2021). Ma et al. (2021) then showed that networks with this new activation could yield significant improvements in various image recognition tasks. Thus, it can be hypothesized that meta-ACON can also improve the detection accuracy of our models. As suggested in (Ma et al., 2021), meta-ACON can have several different design spaces where the best design is the channel-wise design, as shown below.

$$(\varsigma_1 - \varsigma_2) \cdot \sigma(\kappa(\varsigma_1 - \varsigma_2)\mathbf{x}) + \varsigma_2\mathbf{x} \tag{4.35}$$

where $\mathbf{x}$ is the input, σ is the sigmoid function, ς_1 and ς_2 are trainable parameters which are initialized as 1 and 0.25, respectively. On the other hand, the switching factor, κ, is derived using $\sigma(\mathbf{W}_2(\mathbf{W}_1(\text{GAP}(\mathbf{x}))))$ where $\mathbf{W}_1$ and $\mathbf{W}_2$ represent the weights and GAP represents global average pooling. Thus, the concept is to learn κ based on the input sample (Ma et al., 2021). Interestingly, it also resembles the SE module (Hu et al., 2020). However, in our preliminary experiments, we found that such implementation had a worse accuracy than simply setting the switching factor as a trainable parameter initialized as 1. We then simplified the function to derive κ into $\sigma(\mathbf{W}_1(\text{GAP}(\mathbf{x})))$. By doing so, it is capable of achieving better results as compared to simply initializing the switching factor as 1.

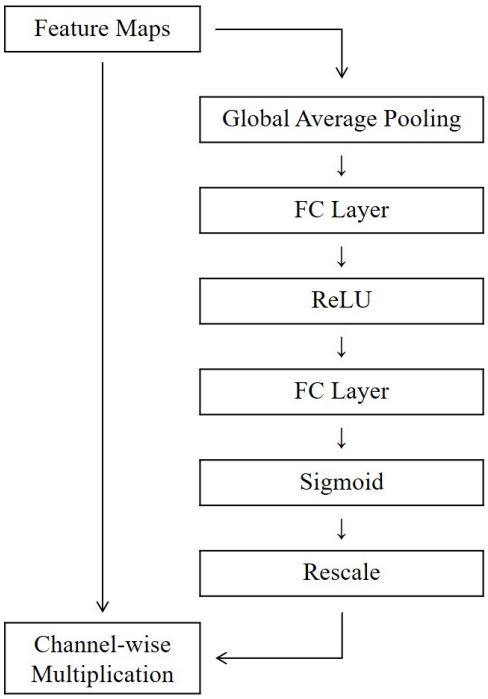


Figure 4.9 SE module

The SE module was developed by Hu et al. (2020), where the goal is to improve the quality of representations produced by a network by explicitly modeling the inter-dependencies between the channels of its convolutional features. A SE module can be easily implemented into any network, as shown in Figure 4.9. Although simple, such an implementation and its variants have shown positive improvements in acoustic-related domains (Naranjo-Alcazar et al., 2020; Xia and Koishida, 2019).

4.4.3. Improved Macaron Net Encoding Layer

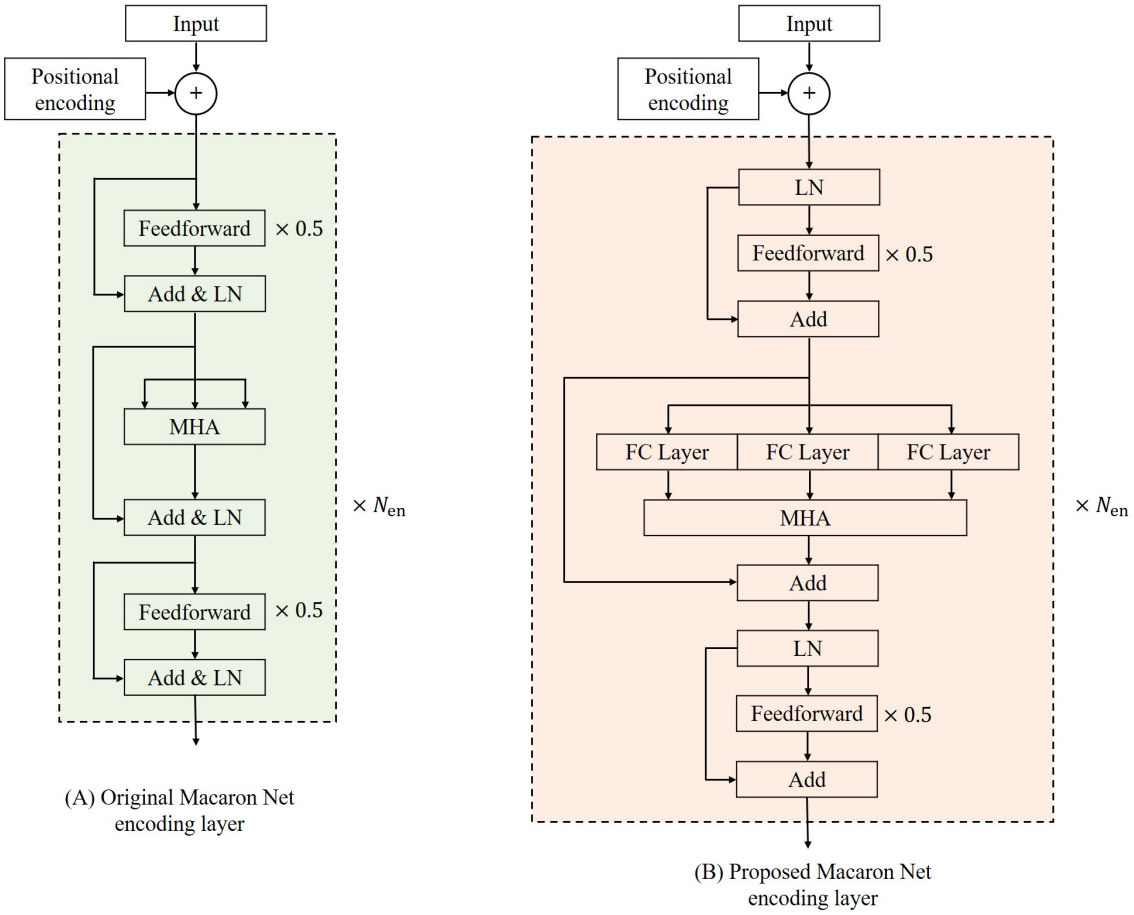


Figure 4.10 Difference between the encoding layers

The third critical module is the proposed Macaron Net encoding layer in Figure 4.10. The proposed Macaron Net encoding layer consists of two significant differences as compared to the original Macaron Net encoding layer (Lu et al., 2019). Firstly, the original Macaron Net encoding layer consists of three LN and are positioned after the residual connections (termed as Post-LN). Instead, we proposed only two LNs placed before the position feedforward networks (termed as Pre-Norm). It is motivated by an analysis that shows that a Post-LN Transformer is less robust than a Pre-Norm Transformer (Liu et al., 2020a).

Secondly, the original Macaron Net encoding layer does not consist of the additional FC layers before the multi-head attention module. This is because we propose using a single layer, single head encoding layer. Since multiple heads were found to be beneficial (Vaswani et al., 2017), the introduction of the additional FC layers is to stabilize the encoding layer.

The reason for using a single layer, single head encoding layer stems from the results shown in Chapter 4.3.2 where a single Macaron Net encoding layer with four heads is sufficient to achieve the optimal results. Moreoever, Voita et al. (2019) and Michel et al. (2019) found that most of the heads can be safely removed without significant performance degradation in machine translation tasks. Michel et al. (2019) even found that the number of heads can even be reduced to one in some of the encoding layers. Thus, in our proposed framework, a single layer single head encoding layer is utilized.

4.4.4. Multi-Branch Pooling

The last critical module is the multi-branch pooling scheme which helps to improve the AT accuracy and avoid overfitting. As the branches are independent of each other, they optimize the model according to their unique learning purposes. Thus, the model can obtain different characteristics which improve the overall performance (Huang et al., 2020b). However, there are several differences in the proposed multi-branch pooling as compared to multi-branch pooling in (Huang et al., 2020a,b).

Firstly, we only include the multi-branch pooling scheme in the TM model instead of the SM model as seen in (Huang et al., 2020a). This is because we found that the inclusion of a Transformer encoding layer can reduce the clip-level prediction accuracy even though the accuracy of the frame-level prediction can be improved. Thus, the multi-branch pooling is only added to the TM model to improve the clip-level prediction.

Secondly, Huang et al. (2020b) approach consist of one main and multiple auxiliary branches where the main branch is used in both the training and inference stage. In contrast, the auxiliary branches are only used in the inference stage. In our proposed multi-branch learning, all branches are used in both the training and inference stages and are weighted equally.

Thirdly, the main branch in Huang et al. (2020b) approach adopts an embedding level approach while the auxiliary branches adopt an instance-level approach. For the instance-level approach, the pooling method aggregates the frame-level prediction by a classifier into clip-level prediction. As for the embedding-level approach, the pooling method is used to map the frame-level feature representation from a classifier into a lower dimension

clip-level feature representation which is passed to another classifier to get the clip-level prediction. Unlike (Huang et al., 2020b), in our multi-branch pooling scheme, all branches adopt an instance-level approach. As for the number of branches, N_{BR}, and the choice of the temporal pooling combination, these will be discussed in the later section.

4.4.5. Proposed Semi-Supervised Learning Framework

The proposed semi-supervised learning framework remains largely similar to the semi-supervised framework in Chapter 4.3.1. However, in this framework, we modified the two consistency losses in Chapter 4.3.1. The first loss component is the frame-level loss, l_f, is defined as

$$l_{\mathrm{f}} = \frac{1}{N_{\mathrm{e}} \times N_{\mathrm{f}}} \sum_{i=1}^{N_{\mathrm{e}}} \sum_{j=1}^{N_{\mathrm{f}}} [g_{i,j} \log(S_{i,j}) + (1 - g_{i,j}) \log(1 - S_{i,j})] \tag{4.36}$$

where $S_{i,j}$ represents the SM's predicted probability of event i at frame j and $y_{i,j}$ represents the ground truth of event i at frame j. The second loss component is clip-level loss, l_c, is defined as

$$l_{\mathrm{c}} = \frac{1}{N_{\mathrm{e}}} \sum_{i=1}^{N_{\mathrm{e}}} [z_i \log(T_i) + (1 - z_i) \log(1 - T_i)] \tag{4.37}$$

where T_{i} represents the TM's predicted probability of event i in a sample and z_{i} represents the grouth truth of event i in the same sample.

The third loss component is the interpolated consistency cost. Given that $\mathbf{U}_1^{\mathrm{A}}$ and $\mathbf{U}_2^{\mathrm{A}}$ represent the feature representation of augmented unlabeled sample 1 and 2 and $\mathbf{U}_1^{\mathrm{U}}$ and $\mathbf{U}_2^{\mathrm{U}}$ represent the feature representation of unaugmented unlabeled sample 1 and 2. The interpolated consistency cost, l_{i}, can be defined as

$$\tilde{\mathbf{S}}^{\mathrm{mc}} = M(\mathrm{SM}(\mathrm{mixup}_{\mathrm{c}}(\mathbf{U}_1^{\mathrm{A}}, \mathbf{U}_2^{\mathrm{A}}))) \tag{4.38}$$

$$\acute{\mathbf{T}}^{\mathrm{m}} = \mathrm{mixup}_{\mathrm{c}}(\mathrm{TM}(\mathbf{U}_1^{\mathrm{U}}), \mathrm{TM}(\mathbf{U}_2^{\mathrm{U}})) \tag{4.39}$$

$$l_i = \mathrm{MSE}(\tilde{\mathbf{S}}^{\mathrm{mc}}, \acute{\mathbf{T}}^{\mathrm{m}}) \tag{4.40}$$

where $\tilde{\mathbf{S}}^{\mathrm{mc}}$ represents the vector that contains the events' probabilities from SM on the mixed unlabeled sample. $M(.)$ represents the max pooling operator. Rather than the use of mixup by linear interpolation (Zhang et al., 2017), we propose the use of mixup by concatenation (Ebbers and Haeb-Umbach, 2020). $\mathrm{mixup}_{\mathrm{c}}(\mathbf{U}_1^{\mathrm{A}}, \mathbf{U}_2^{\mathrm{A}})$ is defined as

$$\mathrm{mixup}_{\mathrm{c}}(\mathbf{U}_1^{\mathrm{A}}, \mathbf{U}_2^{\mathrm{A}}) = \mathrm{Concat}(\bar{\mathbf{U}}_1^{\mathrm{A}}, \bar{\mathbf{U}}_2^{\mathrm{A}}) \tag{4.41}$$

where $\bar{\mathbf{U}}_1^{\mathrm{A}}$ and $\bar{\mathbf{U}}_2^{\mathrm{A}}$ represent a portion of feature representations of $\mathbf{U}_1^{\mathrm{A}}$ and $\mathbf{U}_2^{\mathrm{A}}$. Mixup by concatenation (Ebbers and Haeb-Umbach, 2020) can be illustrated in Figure 4.11.

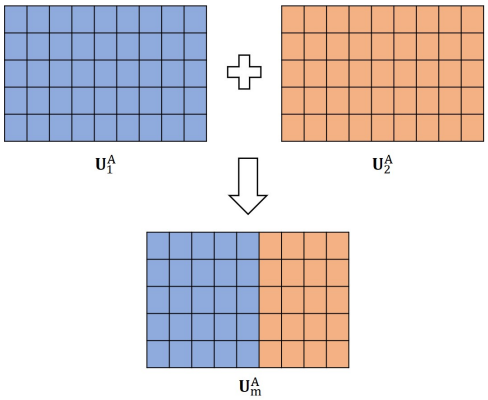


Figure 4.11 Mixup by concatenation

For mixup by concatenation, we cannot apply interpolation to combine the clip-level prediction of $\mathbf{U}_1^{\mathrm{U}}$ and $\mathbf{U}_2^{\mathrm{U}}$. Thus, we added up the clip-level prediction of $\mathbf{U}_1^{\mathrm{U}}$ and $\mathbf{U}_2^{\mathrm{U}}$. As such, $\acute{\mathbf{T}}^{\mathrm{m}}$ represents the vector that contains the combined events' probabilities added using the probabilities of the predicted events in $\mathbf{U}_1^{\mathrm{U}}$ and $\mathbf{U}_2^{\mathrm{U}}$.

Based on the implementation, we are restricting TM to make predictions on unaugmented samples, which will lower the difficulty of prediction by TM and increase the accuracy of clip-level prediction. On the other hand, SM must make its prediction on augmented samples. We hypothesize that forcing SM to learn from a more accurate prediction can increase its frame-level prediction.

However, we only allow l_{i} to be calculated if interpolated probabilities from TM exceeds the confidence threshold. Thus, l_{i} is extended as a regularized interpolated consistency cost, l_{ri} which can be defined as

$$l_{\mathrm{ri}} = \begin{cases} \frac{w}{N_{\mathrm{e}}} \sum_{i=1}^{N_{\mathrm{e}}} (\tilde{S}_i^{\mathrm{mc}} - \acute{T}_i^{\mathrm{m}})^2 & \text{if } \max(\acute{\mathbf{T}}^{\mathrm{m}}) > \lambda_{\mathrm{curr}} \\ 0 & \text{otherwise} \end{cases} \tag{4.42}$$

where $\tilde{S}_i^{\mathrm{mc}}$ and $\acute{T}_i^{\mathrm{m}}$ represent the SM's predicted probability of event i and TM's combined probability of event i. λ_{curr} represents the current confidence level and will be discussed in the next section. w is an additional weighing parameter which is defined as (Laine and Aila, 2017)

$$w = \exp(-5(1-P)^2) \tag{4.43}$$

P represents the training progression and will also be discussed in further detail in the next section.

The last loss component is the self-consistency cost, l_s. l_s is inspired by the concept of self-distillation proposed in (Xu and Liu, 2020). The main idea is to enforce the TM model to be consistent with its prediction regardless of the augmentation applied. The prediction using TM model on augmented samples is given as follows.

$$\hat{\mathbf{T}}^{m} = \mathrm{TM}(\mathrm{mixup}_c(\mathbf{U}_1^A, \mathbf{U}_2^A)) \tag{4.44}$$

where $\hat{\mathbf{T}}^{m}$ represents the TM's predicted probabilities in a mixed unlabeled sample.

l_s is defined as

$$l_s = \begin{cases} \frac{v \times w}{N_e} \sum_{i=1}^{N_e} (\hat{T}_i^{m} - \acute{T}_i^{m})^2 & \text{if } \max(\acute{\mathbf{T}}_m) > \lambda_{curr} \\ 0 & \text{otherwise} \end{cases} \tag{4.45}$$

where v is the loss multiplier for l_s and $\hat{T}_i^{m}$ represents the probability of event i in a mixed unlabeled sample.

4.4.6. Experiment Setup

The dataset used for the subsequent experiments is the DESED 2020 dataset (Turpault et al., 2019), and the data distribution can be found in Table 3.6. As the methodology was not submitted to the DCASE challenge, results on the evaluation 2020 dataset cannot be obtained. Thus, results are only reported on the validation dataset.

Audio preprocessing and feature extraction remain similar, as described in Chapter 3.3.4. As mentioned in Chapter 4.1, the pseudo labeling process remains similar to the labeling process described in Chapter 3.3.2. The only change is in the algorithm used (i.e., NMF is replaced with CNMF).

Similar to Chapter 4.3.2, the training procedure consist of two phases, 1) the warm-up phase and 2) the adaptation phase. The warm-up phase, which lasts 10 epochs, utilizes only the synthetic data and pseudo-strongly labeled data to train the models with a batch size of 32. Each batch of data is evenly split between the synthetic data and pseudo-strongly labeled data, and feature representations are augmented with Gaussian noise and time shift. Since the unlabeled data is not utilized in this phase, the loss component only consists of l_f and l_c. Learning rate begins at 1e-6 and is increased progressively to 0.0012 along a cosine curve which is defined as

$$LR_{curr} = LR_{min} + 0.5(LR_{max} - LR_{min})(1 + \cos(\frac{P_w - P_{curr}}{P_w}\pi)) \tag{4.46}$$

where LR_{max} and LR_{min} are set as 0.0012 and 1e-6, respectively. P_{w} represents the total iterations during the warm-up phase which we set as the total number of iterations in 10 epochs. Based on the calculated losses, models are updated using Adam (Kingma and Ba, 2015).

The adaptation phase (i.e., from the 11th epoch onwards) utilizes synthetic data, pseudo strongly labeled data, and unlabeled data to train the models with a batch size of 64. Each batch of data is split into the following proportion: 25% synthetic data, 25% pseudo strongly labeled data, and 50% weakly labeled data. Feature representations in this phase are also augmented with Gaussian noise and time shift. With the inclusion of unlabeled data, the loss component in this phase would consist of l_{f}, l_{c}, l_{ri} and l_{s}. ν which is the loss multiplier for l_{s} is set as 0.1.

In this phase, we propose an improved cyclic learning scheme to train the models. At the start of the adaptation phase, P_{curr} is reset to 0. The learning rate begins at 0.0012 and is annealed according to the following cosine function (Loshchilov and Hutter, 2017)

$$LR_{\text{curr}} = LR_{\text{min}} + 0.5(LR_{\text{max}} - LR_{\text{min}})(1 + \cos(\frac{P_{\text{curr}}}{P_{\text{tot}}}\pi)) \tag{4.47}$$

LR_{max} and LR_{min} remain at 0.0012 and 1e-6, respectively. P_{tot} represents the maximum training iterations before a learning rate reset. Based on the framework of learning rate reset (Loshchilov and Hutter, 2017), the learning rate will be reset when P_{curr} equals P_{tot} and P_{curr} will revert to 0 while P_{tot} is multiplied with an integer, P_{mult}, which can delay the next restart if P_{mult} is larger than 1.

Instead of an immediate learning rate reset, we propose to increase the learning rate slowly from the minimum to the maximum whenever the learning rate reaches its minimum. Thus, the learning rate during the increase phase can be calculated using Equation 4.46 but with a different P_{w}. Unlike P_{tot}, P_{w} will not be multiplied by P_{mult} and will always remain the same throughout the entire phase 2.

For clarity, the difference in the original cyclic learning scheme and our proposed cyclic learning scheme is illustrated in Figure 12. The two graphs are based on the following settings: LR_{max} as 0.0012, LR_{min} as 1e-6, an initial P_{tot} of 100 iterations, P_{w} as 100 iterations and P_{mult} as 2.

Since the models are trained using this proposed cyclic learning scheme, P which is used to calculate w which in turn affect l_{ri} and l_{s} is proposed to follow the cyclic learning

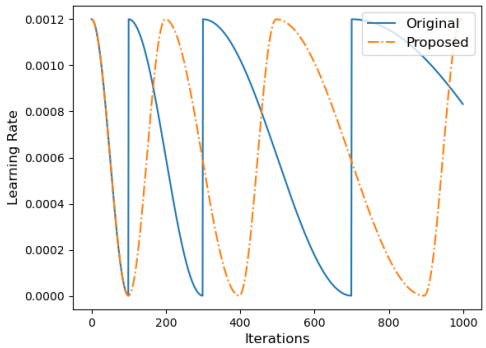


Figure 4.12 Difference in cyclic learning scheme

scheme. Thus, when the learning rate is decreasing, P is defined as

$$P = \frac{P_{\text{curr}}}{P_{\text{tot}}} \tag{4.48}$$

And if learning rate is increasing, P is defined as

$$P = \frac{P_{\text{w}} - P_{\text{curr}}}{P_{\text{w}}} \tag{4.49}$$

Using the same setting to illustrate the proposed learning rate transition, the regularizing parameter, w, which is affected by the value of P, will follow the pattern shown in Figure 4.13.

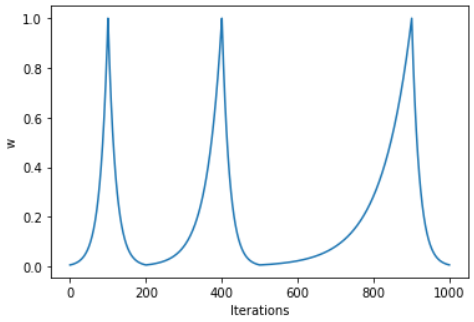


Figure 4.13 Transition of w in the proposed cyclic learning scheme

λ_{curr} is also designed to follow the cyclic pattern. Thus, λ_{curr} can be defined as

$$\lambda_{\text{curr}} = \lambda_{\text{min}} + 0.5(\lambda_{\text{max}} - \lambda_{\text{min}})(1 + \cos(P\pi)) \tag{4.50}$$

where λ_{max} is set as 0.9 and λ_{min} is set as 0.6. P is calculated using Equation 4.48 or Equation 4.49 depending on the transition of learning (i.e., increasing or decreasing).

In this phase, P_{tot} is set as 1 epoch, P_{mult} is set as 2 while P_{w} is set as 1 epoch. The cyclic learning phase will end when P_{tot} reaches 64 epoch. This effectively means the total

Pooling method in both models	AT F1-score (%)	Event-based F1-score (%)
Max	74.1	45.8
Avg	74.8	44.4
LSM	74.8	43.2
ESM	**76.1**	**45.9**
Power	75.1	45.7
Auto	**76.1**	45.1
Att	75.9	**45.9**

Table 4.11 AT and event-based F1 score using different pooling methods

number of epochs in this phase is 133 epochs. Similar to the warm-up phase, models are updated using Adam (Kingma and Ba, 2015) based on the calculated losses.

In the inference stage, both models are used for audio tagging and SED. The method of detection and post-processing steps remain similar, as described in Chapter 4.3.2.

In this section, all experiments were conducted on a system using an AMD Ryzen Processor 5900x with a base frequency of 3.7GHz, 32GB ram and a RTX3090 GPU.

4.4.7. Results and Discussion

In this section, we provide an in-depth analysis of the proposed model layout and hyperparameters. The primary evaluation metric is the event-based F1-score (Mesaros et al., 2016) that determines the accuracy of the frame-level prediction. We then compare our models against the other SOTA.

The first experiment was to investigate the combinations of different temporal pooling methods for our multi-branching pooling scheme. Using a diverse variety of temporal pooling methods, the effects on both SM and TM are examined. To establish a baseline on whether the combination of multi-branch pooling is effective and superior to the single pooling method, the Audio Tagging (AT) and event-based F1-score are tabulated using the single temporal pooling method. A total of 7 different pooling methods was tested. They are namely: Max, Average (Avg), Linear Softmax (LSM), Exponential Softmax (ESM), Power (PP) (Liu et al., 2020b), Auto (AP) (McFee et al., 2018), and Attention (Att).

As seen in Table 4.11, ESM and Auto pooling produce the highest AT F1-score, while ESM and Att pooling generate the highest event-based F1 score. Ideally, with the inclusion of the multi-branch pooling scheme, we should see an increase in AT F1-score while maintaining the event-based accuracy.

N_{BR}	Combination of branches	AT F1-score (%)	Event-based F1-score (%)
2	Max-Avg	76.0	46.2
3	Max-Avg-LSM	**76.4**	**47.1**
4	Max-Avg-LSM-ESM	75.9	46.7

Table 4.12 Effects of N_{BR} on accuracy

In order to determine the optimal N_{BR}, we conducted experiments using different N_{BR}. Based on the results shown in Table 4.12, the use of three branches yielded the best result and is the only option that produces an increase in performance for both AT and event-based F1-score. Thus, we combine only three different temporal pooling methods out of the seven other pooling methods for our subsequent experiments. With the use of only three branches, our pooling method can then be termed as a triple instance-level pooling approach.

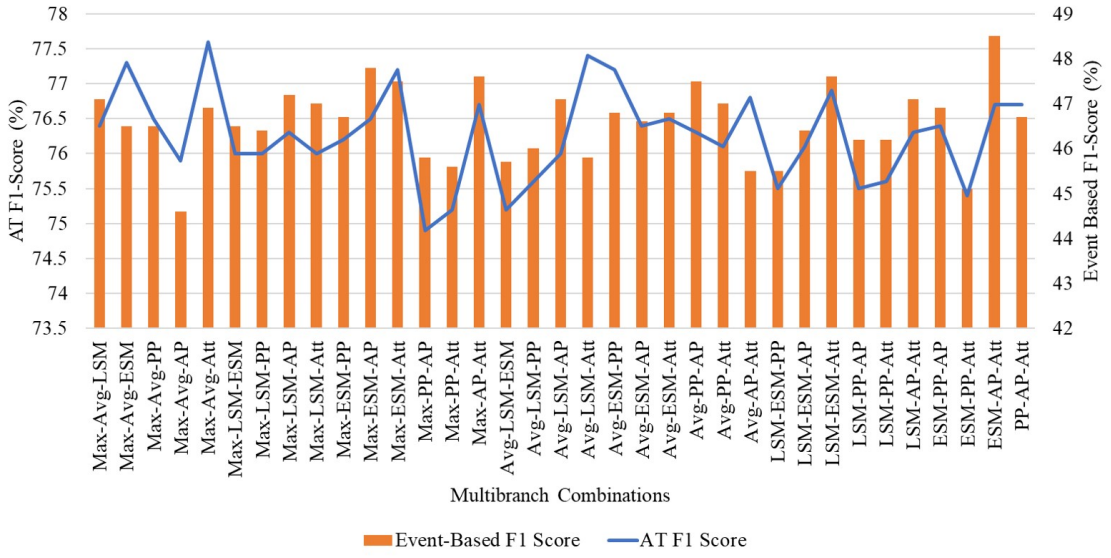


Figure 4.14 Accuracy chart using different multi-branch combinations

This results in 35 different combinations and the respective AT and event-based F1 score can be seen in Table 4.13 and Figure 4.14. Interestingly, there is no guarantee if good AT accuracy will give a good event-based F1-score. However, some combinations can improve the AT by a more considerable margin. Based on the criteria set, there are a total of 19 combinations that provide a higher AT F1-score of 76.1% and a higher event-based F1-score of 45.9% (shaded in grey in Table 4.13). Since the primary evaluation metric is based on the event-based F1-score, ESM-AP-Att exhibits the best combination for multi-branch pooling. It achieves an AT F1-score and event-based F1-score of 76.7% and 48.5%, respectively.

Max Avg LSM	Max Avg ESM	Max Avg PP	Max Avg AP	Max Avg Att
76.4	77.3	76.5	75.9	**77.6**
47.1	46.3	46.5	44.6	46.9
Max LSM ESM	Max LSM PP	Max LSM AP	Max LSM Att	Max ESM PP
76.0	76.0	76.3	76.0	76.2
46.5	46.4	47.2	47.0	46.7
Max ESM AP	Max ESM Att	Max PP AP	Max PP Att	Max AP Att
76.5	77.2	74.9	75.2	76.7
47.8	47.5	45.8	45.6	47.6
Avg LSM ESM	Avg LSM PP	Avg LSM AP	Avg LSM Att	Avg ESM PP
75.2	75.6	76.0	77.4	77.2
45.7	46.0	47.1	45.8	46.8
Avg ESM AP	Avg ESM Att	Avg PP AP	Avg PP Att	Avg AP Att
76.4	76.5	76.3	76.1	76.8
46.6	46.8	47.5	47.0	45.5
LSM ESM PP	LSM ESM AP	LSM ESM Att	LSM PP AP	LSM PP Att
75.5	76.1	76.9	75.5	75.6
45.5	46.4	47.6	46.2	46.2
LSM AP Att	ESM PP AP	ESM PP Att	ESM AP Att	PP AP Att
76.3	76.4	75.4	76.7	76.7
47.1	46.9	45.1	**48.5**	46.7

Table 4.13 AT (first row) and event-based F1-score (second row) (%) using different pooling combinations

Further experiments using the combination of ESM-AP-Att are then conducted. We first ablate the use of l_{ri} to investigate the hypothesis that frame-level prediction can be improved if the prediction of the SM is near to the prediction of TM. As seen in Table 4.14, without the use of l_{ri}, accuracy can drop by 4%.

With L_{ri}	Without L_{ri}
48.5	44.5

Table 4.14 Effects of L_{ri} on event-based F1-score

ν	AT F1-score (%)
0	**76.7**
0.1	**76.7**
0.5	74.4
1.0	70.0

Table 4.15 Sensitivity analysis of ν

Multi-branch combination	Event-based F1-score	
	Multi-branch in TM	Multi-branch in both models
ESM-AP-Att	**48.5**	46.1

Table 4.16 Effect of having multi-branch pooling

Pooling method in SM	Event-based F1 score (%)
Max	**48.5**
Avg	43.6
LSM	46.0
ESM	44.8
Power	45.0
Auto	45.4
Att	46.4

Table 4.17 Accuracy using different pooling methods in SM and ESM-AP-Att in TM

We then investigate the sensitivity of ν on the AT accuracy, and the results of setting different values of ν are analyzed in Table 4.15. Using l_{s} in the proposed framework has no merit. In fact, AT accuracy can degrade with higher ν. This degradation can be due to the overconfident predictions by the AT model. It may also be due to how the clip-level predictions are combined (clip-level predictions are added together for mixup by concatenation). As the event-based evaluation is our primary evaluation, only the event-based accuracy will be considered in our subsequent tests.

To raise the accuracy of the frame-level prediction, the multi-branch pooling into the SM model is included. However, as seen in Table 4.16, event-based accuracy cannot be

improved with multi-branch pooling. Instead, it can decrease. Since the multi-branch pooling scheme does not work well with the SM, the best pooling method for the SM is being investigated. Based on the results in Table 4.17, max-pooling is the best pooling operator.

In contrast, average pooling performed the worst among the other pooling operators. This is also seen in Chapter 3.3.5. The nature of average pooling is to assign equal weights to all frames (McFee et al., 2018; Wang et al., 2019a); as such, during the backpropagation, the gradient is distributed evenly across all frames (Wang et al., 2019a). For negative clips (clips that do not contain an event of interest), this will suppress the frames' probabilities, which is the correct behavior (Wang et al., 2019a). But for positive clips (clips containing an event of interest), not all frames should be boosted, and the average pooling function can produce many false-positive frames (Wang et al., 2019a). Such behavior can explain why using average pooling to obtain the clip-level prediction from the SM model and forcing it to be consistent with the clip-level prediction from TM may not be a good idea.

By considering only the event-based F1 score, max-pooling also works relatively well in a single temporal pooling setting (i.e., no multi-branch module included in any model) as seen in Table 4.11. These results contradict the earlier comparative study (Wang et al., 2019a) where LSM was shown to be the best pooling operator while max-pooling was the worst pooling operator for temporal localization. Our results coincide with the findings shown in (Kao et al., 2020), where the best pooling operator is the max pooling. Kao et al. (2020) hypothesized that the difference in the results is due to the dataset's difference. However, we find that the difference in the model and/or the post-processing method can also contribute to the difference in pooling performance. As explained in (McFee et al., 2018; Wang et al., 2019a), one possible limitation of max-pooling is that during the backpropagation, only one value gets updated (i.e., the maximum value), which can cause many frame-level false negatives. Recall that we use an event-specific frame threshold in our post-processing method to decide if a frame is activated (i.e., containing an event of interest) and subsequently using a lower-bound threshold to decide if neighboring frames should be activated. Thus, the use of a lower bound threshold may help circumvent the issue of false negatives, which may explain why max-pooling appears to work relatively well.

After the optimal choice of pooling methods is established, a sensitivity test is done on the number of heads and encoding layers. As seen in Table 4.18, there is no benefit in having more heads or more encoding layers as the accuracy does not increase with the increased complexity. In our experiment, using a single head single encoding layer is the

		No. of layers				Ablation of encoding layer
		1	2	3	4	
N_h	1	**48.5**	47.0	46.5	47.0	46.0
	4	47.0	47.2	47.8	47.3	
	8	47.3	46.8	46.6	47.8	
	16	47.0	47.5	48.1	47.1	

Table 4.18 Accuracy of model using different number of heads and encoding layer

Norm arrangement	Event-based F1 score (%)
(a) Pre-Norm 2 times (proposed)	**48.5**
(b) Post-Norm 2 times	47.1
(c) Pre-Norm 3 times	45.6
(d) Post-Norm 3 times	46.9
(e) Pre-Norm 2 times without additional FC layers	46.2
(f) Post-Norm 2 times without additional FC layers	44.3
(g) Pre-Norm 3 times without additional FC layers	45.7
(h) Post-Norm 3 times without additional FC layers	45.6

Table 4.19 Effects of LN and proposed feedforward networks

optimal choice. It coincides with the previous study (Michel et al., 2019) that more heads do not improve accuracy. Instead, it could reduce the accuracy.

Such a conclusion then raises the question of whether the encoding layer is beneficial or redundant. We then ablate the encoding layer in both models, but the result shown in Table 4.18 indicates that using an encoding layer is still beneficial to the overall detection accuracy.

The effects of the different numbers of LN and their positions are then studied. The need for having additional FC layers in the encoding layer to stabilize the architecture is also examined. A total of 4 different LN positions were tested, (a) as proposed in Figure 4.10, (b) having normalization after the position-wise feedforward network residual connections, (c) as proposed in Figure 4.10 with extra LN before the proposed additional FC layers and (d) having LN after each residual connection. These tests were repeated without the proposed additional feedforward network.

Type of SE arrangements	Event-based F1 score (%)
(a) SE module in every convolutional block	**48.5**
(b) SE module in every two convolutional block	46.5
(c) No SE module	45.9

Table 4.20 Effects of SE modules

Type of cyclic learning rate	Event-based F1 score (%)
None	46.1
$P_{\text{tot}} = 1$, $P_{\text{w}} = 0$, $P_{\text{mult}} = 2$	46.8
$P_{\text{tot}} = 5$, $P_{\text{w}} = 0$, $P_{\text{mult}} = 2$	47.4
$P_{\text{tot}} = 10$, $P_{\text{w}} = 0$, $P_{\text{mult}} = 2$	45.3
$P_{\text{tot}} = 1$, $P_{\text{w}} = 1$, $P_{\text{mult}} = 2$	**48.5**
$P_{\text{tot}} = 5$, $P_{\text{w}} = 1$, $P_{\text{mult}} = 2$	47.7
$P_{\text{tot}} = 10$, $P_{\text{w}} = 1$, $P_{\text{mult}} = 2$	47.7

Table 4.21 Effects of different T_{i} and proposed slow learning rate transition

From the results tabulated in Table 4.19, having Pre-Norm encoding layer can result in better accuracy with or without the proposed additional networks, which is consistent with the analysis shown in (Liu et al., 2020a). But with the inclusion of additional FC layers, there is no need for another LN after the first residual connection. From the results shown in Table 4.19, an increase in performance can be observed with the additional FC layers except for the three LN in a Pre-Norm setting.

The contribution of the SE module is also analyzed. Three different SE placements are performed. They are namely: a) having SE module as proposed in Figure 4.6, b) having only 1 SE module after 2 convolution operations in TM, and c) having no SE module. As seen in Table 4.20, the SE module can increase the accuracy of the SM. Without the use of SE, the model can result in an accuracy drop of 2.6%. Although we only varied the number of SE in the TM, there is also a drop in accuracy of 2%. It validates that the SM generally learns better from a more complex TM. With this improvement, the use of a more sophisticated SE module (Naranjo-Alcazar et al., 2020; Xia and Koishida, 2019) may yield even better performance.

A test with different values of P_{tot} (in terms of epoch) while setting P_{w} as 1 epoch and P_{mult} as 2 is also performed. As seen in Table 4.21, having a slow transition to the maximum learning rate is beneficial, although the benefit diminishes as P_{tot} increases.

Pseudo labeling method	Event-based F1 score (%)
Type-1 Labeling (Set all frames of weakly labeled data as 0)	29.7
Type-2 Labeling (All frames are assumed to contain the annotated event labels)	29.8
NMF	45.8
CNMF	**48.5**

Table 4.22 Effectiveness of CNMF for pseudo labeling (Illustration of type-1 and type-2 labeling is given in Section 3.3.5)

Nevertheless, models trained using cyclic learning rate perform better than models trained without cyclic learning rate. As suggested in (Loshchilov and Hutter, 2017), reducing the $LR_{\max}$ and $LR_{\min}$ after every restart may yield even better results. However, a longer model tuning time is required to obtain the optimal reduction factor.

Finally, we conducted additional experiments to investigate on the usefulness of pseudo labels. Based on the results shown in Table 4.22, models trained with pseudo labels can obtain better accuracy. The results also indicate that the use of CNMF for pseudo labeling is more effective than the use of NMF, which is consistent with our previous finding in Chapter 4.3.3. However, using naïve labeling method (all frames are assumed to contain the annotated event labels) may not be very useful due to the large amount of noise introduced.

One interesting question about training with pseudo labels is why it works so well, given that pseudo labels will still contain a certain level of noise. One common perspective and understanding is that maximizing accuracy on noisy labels can lead to overfitting. However, recent studies have shown that training with noisy labels can increase the model generalizing ability (Chen et al., 2020; Li et al., 2020). Thus, based on the results shown in Table 4.22, as long as the pseudo labels are not of inferior quality (i.e., naïve labeling method), they can be used for model training.

4.4.8. Comparison against SOTA

We then compared the proposed model with the top three submissions from DCASE 2020, DCASE 2019 Challenge Task 4, and the baseline of DCASE 2020 Challenge Task 4. Without using any ensembling techniques, the proposed model can achieve an event-based F1-score of 48.5% on the validation dataset. As seen in Table 4.23, it translates to the best among the top submissions. The proposed model can outperform the ensembled system in

2019. When compared to the baseline systems, the proposed method provides a margin of over 12%.

Methodology	Event-based F1 score (%)		No. of parameters		Training time (Hr)	Hardware
	Non-ensembled	Ensembled	Non-ensembled (M)	Ensembled (M)		
Proposed	**48.5**	50.4	4.5	22.5	6	1 RTX 3090
CNN+Conformer (Miyazaki et al., 2020) 1st DCASE 2020	46.0	**50.6**	2	17	12	1 TITAN XP
CRNN (Yang et al., 2020) 2nd DCASE 2020	48.3	-	2	-	-	-
FBCRNN (Ebbers and Haeb-Umbach, 2020) 3rd DCASE 2020	46.4	49.2	2	20	24	4 RTX 2080
CNN (Lin et al., 2019) 1st DCASE 2019	44.5	45.4	-	7	3	1 GTX 1080 Ti
CRNN (Delphin-Poulat et al., 2020) 2nd DCASE 2019	43.6	-	1	-	21	1 GTX 1080
CRNN (Shi et al., 2019) 3rd DCASE 2019	42.5	-	-	6	24	1 TITAN XP
Baseline CRNN (Turpault and Serizel, 2020)	34.8	-	1	-	3	1 GTX 1080 Ti
Baseline CRNN With Source Separation (Turpault et al., 2020b)	35.6	-	1	-	3	1 GTX 1080 Ti

Table 4.23 Comparison of system against other SOTA

The top five models are ensembled (models that use the following multi-branch combinations: ESM-Auto-Att, Max-ESM-AP, LSM-ESM-ATT, Max-AP-Att, and Max-ESM-Att) by averaging the posterior probabilities; and this can raise the event-based F1-score to 50.4%. In summary, the proposed framework has shown to be competitive as compared to the SOTA.

In terms of system parameters, the proposed methodology has a higher number of parameters used. This is because, during the inference stage, we are using two models for inference, one for clip-level prediction while the other for frame-level prediction. On the other hand, other methodologies are using only one model for inference. However, it should be pointed out that other methodologies that utilized the Mean-Teacher (Delphin-Poulat et al., 2020; Miyazaki et al., 2020; Shi et al., 2019; Turpault and Serizel, 2020; Turpault et al., 2020b) or guided learning (Lin et al., 2019) approach also utilizes more than one model during the training stage. Thus, the number of parameters for methodologies (Delphin-Poulat et al., 2020; Miyazaki et al., 2020; Shi et al., 2019; Turpault and Serizel, 2020; Turpault et al., 2020b) that utilized the Mean-Teacher approach should multiply by 2 during the training phase. In this sense, the number of parameters used by the proposed methodology is comparable to the top 3 submissions from DCASE 2020 during the training phase.

In terms of training time, our system took a much shorter time than the top submissions from DCASE 2020. While it may be argued that we are training the system using better hardware but based on the information given in (Dettmers, 2020), the performance difference using an RTX 3090 as compared to the RTX 2080 and Titan XP is not too huge. Compared to the other methodologies (Lin et al., 2019; Turpault and Serizel, 2020; Turpault et al., 2020b) our system took a longer training time. However, we argue that the gain in accuracy justified the longer training time.

As for our system inference time on the validation set, it took around 32s to complete the prediction on 1168 audio clips where each clip has a duration of 10s. This shows that it can be suitable for online or streaming applications.

We then performed an analysis of the classwise accuracy of our proposed method. Based on the results shown in Table 4.24, our method does not have good detection accuracy on Dog and Dishes as compared to the other event labels. We hypothesis that there may be several reasons for this observation. Firstly, our method performs better for background sound with a longer duration while it does not perform well for short impulse sound. Secondly, our post-processing method may also be a cause for such a result. As we remove events with a duration shorter than 0.1s, there may be a chance that

Event label	Classwise event-based F1 score (%)	
	Non-ensembled	Ensembled
Speech	51.4	52.2
Dog	34.5	35.3
Cat	45.6	45.4
Alarm Bell Ringing	50.2	52.3
Dishes	24.8	27.4
Frying	47.7	47.7
Blender	51.9	54.9
Running Water	48.9	50.7
Vacuum Cleaner	70.7	75.9
Electric Shaver/Toothbrush	59.1	63.0
Mean	48.5	50.4

Table 4.24 Classwise accuracy of proposed methodology

such short impulsive noise is removed. Since we concatenate two similar events, if the difference between the first event offset and the second event onset is shorter than 0.2s, successive short impulse sound may be concatenated together, which reduces the detection accuracy. As observed, Dishes achieve a much lower accuracy as compared to Dog. It may be because the Dishes events co-occur with other events such as Frying most of the time, which increases the training and detection difficulty. Although the ensembling technique can help increase the accuracy, there is a need to further improve the detection accuracy for short impulse sound.

4.5. Summary

Based on the results shown in this chapter, there are several critical take-home points. Firstly, pseudo strong labels provided using supervised CNMF appear to be of better quality than the pseudo strong labels provided using supervised NMF since models trained using the former can obtain a higher detection accuracy. However, due to the shift operations in CNMF, the formation of event dictionaries and the pseudo-labeling process can take a much longer time. Nevertheless, since the feature extraction process is only a one-time process, CNMF should replace NMF in the pseudo labeling framework.

The inclusion of the Macaron Net encoding layer does provide an improvement in the frame-level prediction. However, it can also reduce the clip-level prediction. To

this end, we propose a triple instance pooling approach to allow the TM to learn unique characteristics from each branch which allows the clip-level prediction to improve.

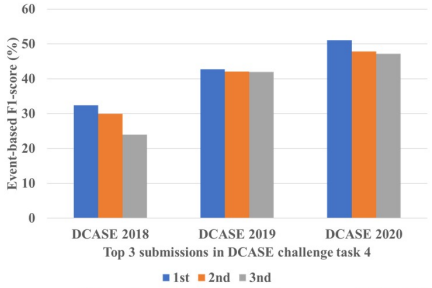
Subsequently, based on the observation that DNN learns an easier example before moving to a more difficult example, we propose a curriculum consistency loss where the confidence threshold is adjusted according to the learning stage. Adjusting the confidence threshold from a high value to a lower value restricts the models to consider the high confidence (lower difficulty) examples first before the low confidence (higher difficulty) examples. Such implementation avoids the scenario that SM has to learn from TM when TM has yet to achieve optimal detection accuracy on all examples. At the same time, it allows more training examples to be included for loss calculation, which is shown to improve the model's accuracy.

While our proposed student-teacher framework can be competitive against the SOTA, the design of two completely different models can be tedious and time-consuming. Chapter 4.4 provides a more straightforward design for the two models where the differences only lie in the number of convolutional layers, pooling size, and temporal pooling method. Together with our proposed ideas, such as the use of meta-ACON, SE module, an improved Macaron Net encoding layer, triple instance-level pooling approach, and improved cyclic learning scheme, we show that even with a straightforward design, it can still be effective for SED where our system can achieve an event-based F1-score of 48.5%. By ensembling the top five models, the event-based F1-score can be increased to 50.4%. Such results allow our model to have a minimum margin of over 12% against the baseline system and be competitive against the other SOTA.

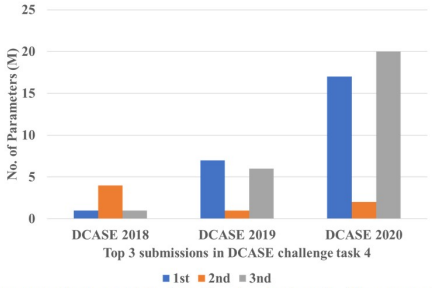
Chapter 5. Lightweight Convolutional-iConformer For SED

5.1. Motivation

As seen in the previous chapters, deep learning can be considered the gold standard for modeling a SED system. However, deep learning does have its limitations.



(A) Accuracy of top 3 submission in the annual DCASE challenge



(B) Number of parameters used in top 3 submission in the annual DCASE challenge

Figure 5.1 Accuracy versus parameter used by the top 3 submissions in the annual DCASE challenge task 4 (evaluation dataset)

Firstly, deep learning approaches are highly parameterized. As seen in Figure 5.1, while deep learning approaches have allowed participants to produce models with higher accuracy each year in the annual DCASE challenge task 4, the models' complexities have also increased significantly over the years.

Although the use of larger models can achieve higher accuracy, they can have several drawbacks. Larger models may require a higher hardware investment cost due to the need for multiple dedicated hardware such as the Graphics Processing Unit (GPU) to perform model training and tuning purposes. In addition, larger models may also incur a longer training time as compared to smaller models. This can result in a larger carbon footprint due to the energy required to power the hardware for a longer duration (Strubell

et al., 2019). Subsequently, large models may also face deployment issues in many real-world applications as recognition tasks need to be carried out in a timely fashion on a computationally limited platform (Howard et al., 2017).

As mentioned earlier, to maximize the potential of a SED system trained using deep learning approaches, there may be a need for a large set of strongly labeled data for model development. Unlike image categorization, annotating the onset and offset of an acoustic event can be difficult and subjective due to the fade in and fade out effect (Chan and Chin, 2020). Even with the use of the multi-annotator approach, the resolution may, at best, be up to 0.1s (based on an informal spot check) (Hershey et al., 2021). As such, perfect onset and offset annotation can be considered unrealistic, and in fact, with the increase in data size, the issue of label noise is inevitable (Fonseca et al., 2020). The issue of label noise is naturally extended to pseudo labels as well. Although noisy labels are found to improve generalization (Chen et al., 2020; Li et al., 2020), too much label noise can significantly degrade the mode's performance.

In this chapter, we attempt to address the challenges as discussed earlier. Firstly, in order to reduce the number of parameters and make our SED system as lightweight as possible, we adopted depthwise separable convolutions (Chollet, 2017b) to replace the conventional convolutions in our models. A depthwise separable convolution is a form of factorized convolution that factorize a standard convolution into a depthwise convolution and a 1×1 convolution called a pointwise convolution (Howard et al., 2017). The differences between the convolutions can be seen in Figure 5.2 where N_c represents the number of channels. As seen in Figure 5.2A, a 3 x 3 filter will span through all the channels in the conventional convolution. In constrast, a depthwise convolution will require N_c 3 x 3 filters where each filter will only convolve with one input channel, which can be seen in Figure 5.2B. As mentioned earlier, in a depthwise separable convolution, a pointwise convolution is then applied with a 1 x 1 filter which is illustrated in Figure 5.2C.

These two operations are usually implemented sequentially without nonlinearities (Chollet, 2017b). In our implementation, we added BN and Swish (Ramachandran et al., 2017) after the first depthwise convolution, which can increase the overall detection accuracy. With a lesser number of parameters, not only is the model computationally cheaper to train and run but is also less prone to overfitting.

We then propose a new Conformer encoding layer which we termed as improved-Conformer (iConformer). This encoding layer will only be added to the shallower model to improve the frame-level prediction. As compared with the original Conformer (Gulati

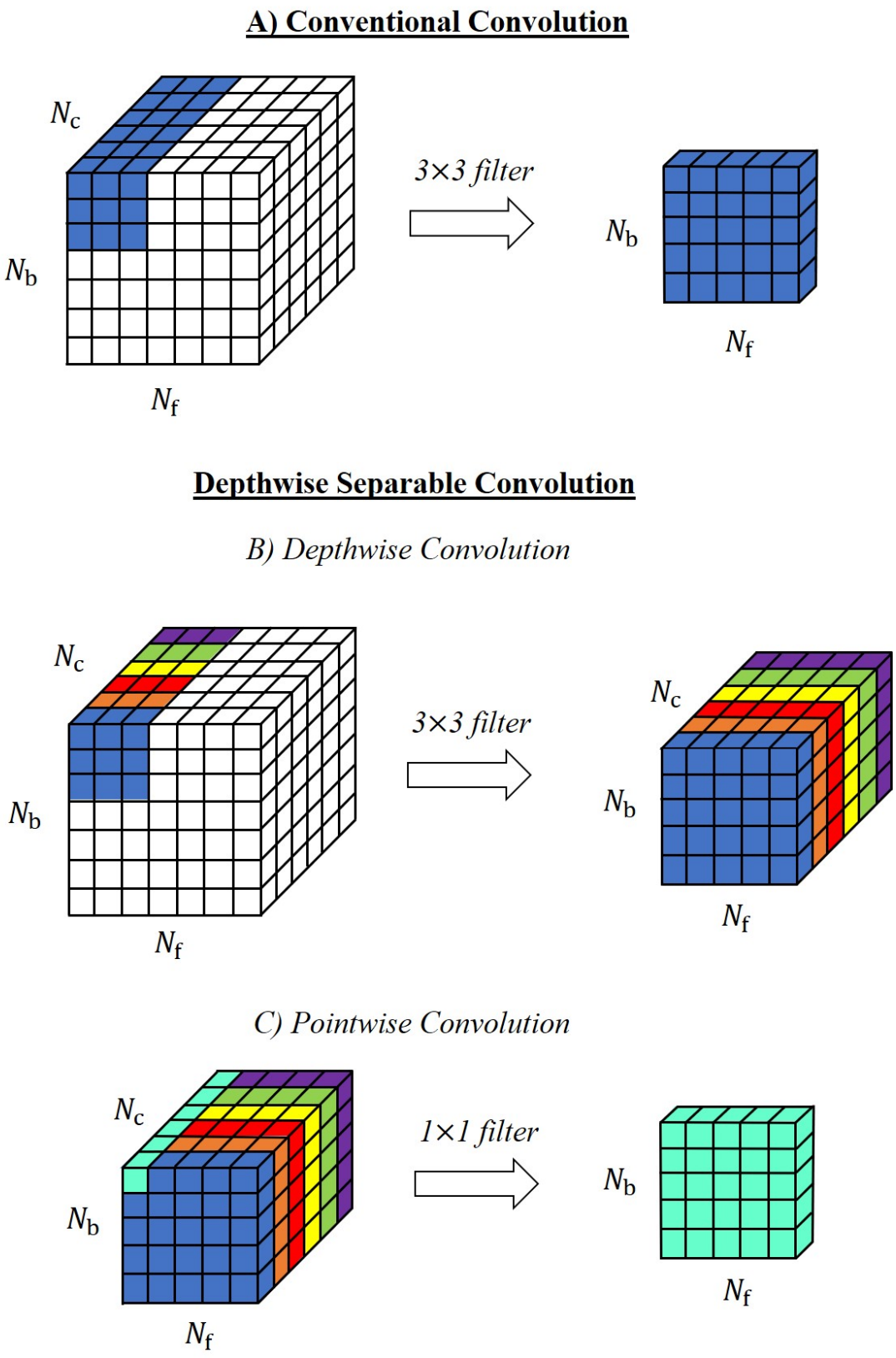


Figure 5.2 Differences between the conventional convolution and depthwise separable convolution

et al., 2020), our encoding layer consists of the multi-branch feedforward module, lesser convolution operation, and also utilizes lesser parameters.

Since the presence of label noise is inevitable, we propose to extend the BCE loss function by including the Reverse Binary Cross Entropy (RBCE) to combat the label noise caused by pseudo labeling.

Thus, the key contributions of this chapter can be summarized as follow

1. A lightweight SED system using modified depthwise separable convolutions which only has 509k parameters.

2. An improved Conformer (iConformer) encoding layer.

3. The extension of BCE with RBCE to combat label noise.

Based on the proposed idea, our lightweight system can obtain an event-based F1-score of 52%, and the ensemble of four systems can further improve the accuracy to 53.5%. Such results indicate a minimum margin of 16% against the DCASE 2020 challenge task 4 baseline system. Compared to the first-place system of the DCASE 2020 challenge task 4, our non-ensembled system can achieve a higher event-based F1-score of 6% with 75% lesser parameters. In terms of the performance of the ensembled system, our system remains competitive and has a winning margin of 2.9% despite using 15 million lesser parameters. Comparison with other state-of-the-art also indicates that our system performance is better despite using a lightweight system. The following subsections then present an in-depth explanation and discussion of our proposed methodology.

This chapter was summarized as journal paper and is currently under review.

5.2. Preliminaries of Conformer and Symmetrical Cross Entropy

5.2.1. Conformer

As mentioned earlier, a Transformer is a good candidate for sequence modeling and was found to outperform RNN in various domains such as language translation (Vaswani et al., 2017) and speech recognition (Karita et al., 2019). However, they are less capable of extracting fine-grained local feature patterns (Gulati et al., 2020). One solution to this issue is to combine convolution and Transformer, which is found to improve the overall performance instead of using them individually (Bello et al., 2019). As such, Gulati et al. (2020) proposed Conformer, which is an improved variant of Macaron Net (Lu et al., 2019).

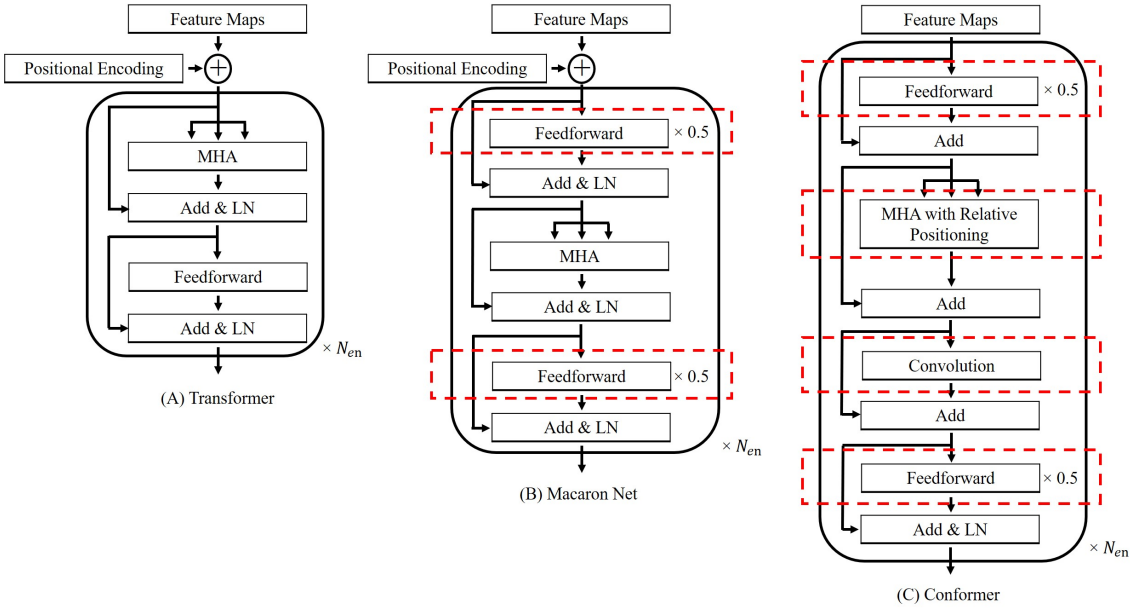


Figure 5.3 Difference between the encoding layers

In Figure 5.3, the architectures of the encoding layer of a Transformer (Vaswani et al., 2017), Macaron Net (Lu et al., 2019), and a Conformer (Gulati et al., 2020) are illustrated. The key differences of each implementation (i.e., Macaron Net and Conformer) with the Transformer encoding layer are highlighted with red dotted boxes in Figure 5.3. As seen from Figure 5.3 to Figure 5.11, the critical differences for Conformer are 1) the use of relative positional encoding (Shaw et al., 2018) instead of absolute positional encoding (Vaswani et al., 2017), 2) the position of layer normalization, and 3) the inclusion of convolution module.

A relative positional encoding can be considered a carefully designed bias term incorporated inside the self-attention module to encode the distance between any two positions (Ke et al., 2021), allowing the self-attention module to generalize better on different input lengths (Gulati et al., 2020) . The resulting encoder can be more robust to the variance of the utterance length (Gulati et al., 2020) and Shaw et al. (2018) demonstrated that such encoding could improve the translation quality on two machine translation tasks.

As seen in Figure 5.3, layer normalization is placed after the residual connection for Transformer (Vaswani et al., 2017) and Macaron Net (Lu et al., 2019). In contrast, based on the information given in Figure 5.4, Figure 5.5, and Figure 5.6, Conformer adopts a Pre-Norm architecture, which may be motivated by an analysis that shows that a Pre-Norm Transformer is less robust than a Post-Norm Transformer (Liu et al., 2020a).

The convolutional module adopted in the Conformer starts with a layer normalization followed by a pointwise convolution and a GLU (Dauphin et al., 2018) activation function. This is followed by a single 1-D depthwise convolution layer with Swish (Ramachandran

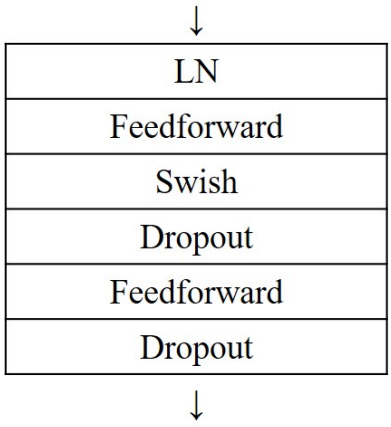


Figure 5.4 Positionwise feedforward module in a Conformer

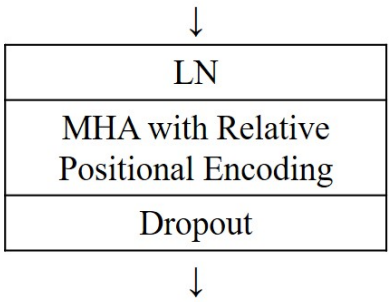


Figure 5.5 Multi-head attention with relative positional encoding in Conformer

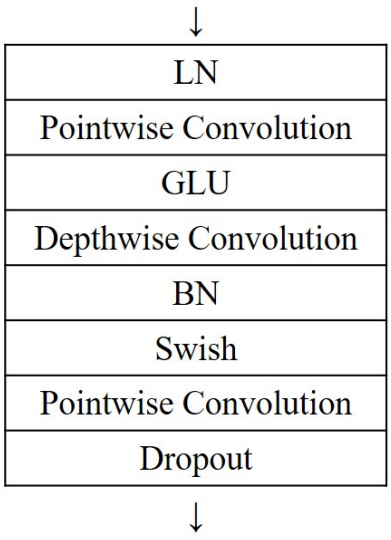


Figure 5.6 Convolutional module in Conformer

et al., 2017) activation function. BN is applied after the final convolution, which aids the training of deep models (Gulati et al., 2020). Notice that they are using a lightweight convolution in this module instead of the conventional convolution to reduce the computational cost.

Based on such an arrangement, Gulati et al. (2020) reported significant improvements over the Transformer (Vaswani et al., 2017) for language modeling. Moreover, the first place submission in the DCASE 2020 challenge task 4 was a CNN with Conformer (Miyazaki et al., 2020). Such results motivate us to proceed with the experimentation using Conformer.

5.2.2. Symmetrical Cross Entropy

In a typical multiclass classification problem (i.e., one sample can only belong to one class), the use of the Cross Entropy (CE) to calculate the loss can be considered one of the most commonly used functions (Wang et al., 2019b). Consider a N_e-class dataset, a sample's ground truth and predicted probability for the i class can be represented as z_i and $\hat{z}_i$, respectively. The CE loss, l_{ce}, can be defined as

$$l_{ce} = -\frac{1}{N_e}\sum_{i=1}^{N_e} z_i \log(\hat{z}_i) \tag{5.1}$$

However, as shown in (Wang et al., 2019b), l_{ce} can exhibit overfitting to noisy labels on easier classes and suffers from significant under-learning on harder classes. Thus, Wang et al. (2019b) extend the idea of symmetric Kullback-Leibler divergence to l_{ce}. This leads to a Symmetric Cross Entropy (SCE) loss, l_{sce}, that consider both l_{ce} and Reverse Cross Entropy (RCE) loss, l_{rce}. Formally, l_{sce} is defined as (Wang et al., 2019b)

$$l_{sce} = \mu l_{ce} + \tau l_{rce} \tag{5.2}$$

where μ and τ represent two hyperparameters for controlling the contribution of each loss term. l_{rce} is defined as

$$l_{rce} = -\frac{1}{N_e}\sum_{i=1}^{N_e} \hat{z}_i \log(z_i) \tag{5.3}$$

As l_{rce} was proven to be noise robust (Wang et al., 2019b), the use of l_{sce} can strike a balance between sufficient learning and robustness to noisy labels. Thus, l_{sce} can allow overall learning to be improved in all classes.

Given that l_{sce} can be a promising loss function to combat noisy labels, we attempt to extend such loss function to a multi-label scenario (i.e., one sample can belong to multiple class). In a multi-label scenario, the l_{ce} is extended to BCE loss, l_{bce}, and is defined as

$$l_{\text{bce}} = -\frac{1}{N_{\text{e}}}\sum_{i=1}^{N_{\text{e}}}(z_i \log(\hat{z}_i) + (1-z_i)\log(1-\hat{z}_i)) \tag{5.4}$$

Thus, the RBCE, l_{rbce}, would be defined as

$$l_{\text{rbce}} = -\frac{1}{N_{\text{e}}}\sum_{i=1}^{N_{\text{e}}}(\hat{z}_i \log(z_i) + (1-\hat{z}_i)\log(1-z_i)) \tag{5.5}$$

Following the implementation of l_{sce} loss, the Symmetric Binary Cross Entropy (SBCE), l_{sbce}, loss can be defined as

$$l_{\text{sbce}} = \mu l_{\text{bce}} + \tau l_{\text{rbce}} \tag{5.6}$$

Notice that the log terms will become infinity in Equation 5.4 and Equation 5.5 when z_i or $\hat{z}_i$ is equal to 1 or 0. In such cases, the log terms are clipped to -100, which is similar to the PyTorch implementation of BCE loss. In this chapter, we propose using SBCE to calculate the loss between the frame-level prediction and the pseudo strong labels to combat the noise contained in the pseudo strong labels.

5.3. Model Layout

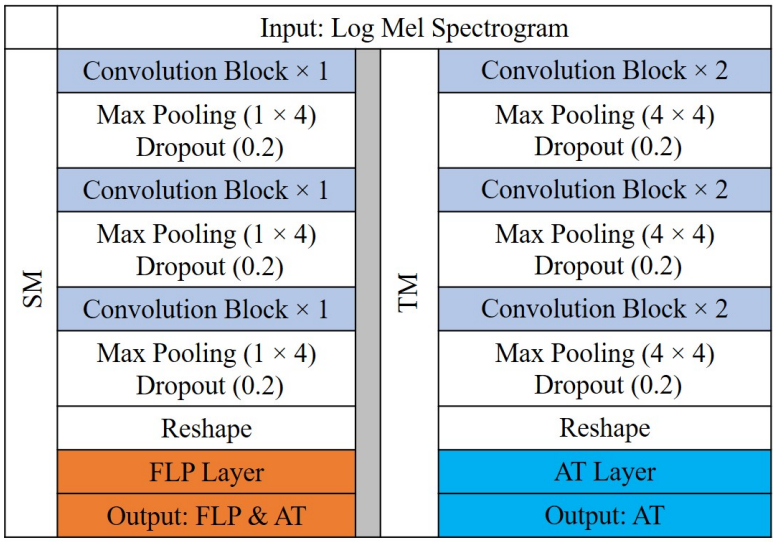


Figure 5.7 SM and TM

We adopted the framework as proposed in Chapter 3 and Chapter 4, where two models are designed differently but trained synchronously. By referring to Figure 5.7, one would

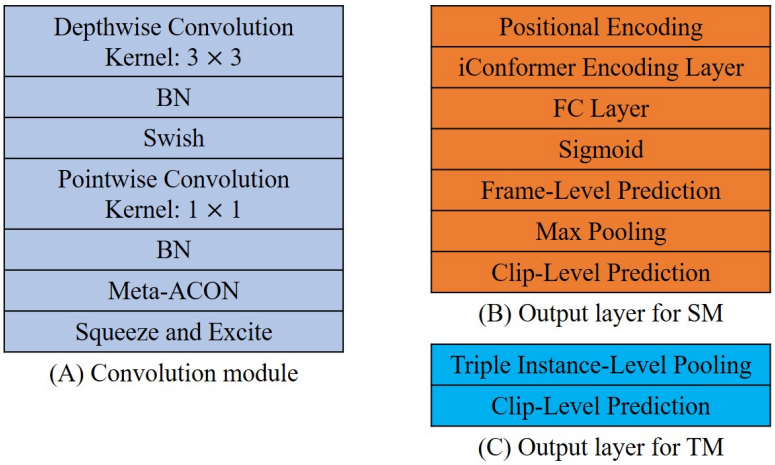


Figure 5.8 Modules description

notice that the model designs are based on the proposed idea in Chapter 4 (refer to Figure 4.6), where the differences in models are in the number of convolutional layers, pooling size, temporal pooling method. Similar to before, the SM will provide two outputs, the frame-level prediction, and the clip-level prediction. In contrast, the TM only provides the clip-level prediction. TM model adopts a triple instance-level pooling which combines exponential softmax, auto pooling (McFee et al., 2018), and attention pooling to obtain the clip-level prediction. SM only adopts a max pooling operation to obtain the clip-level prediction. Such settings are based on the experiment done in Chapter 4, which yielded the best results.

However, by comparing the modules proposed in this chapter (Figure 5.8) and the modules proposed in Chapter 4 (Figure 4.7), one would notice the difference in the convolution operations. In Chapter 4, the use of the conventional convolution method is proposed. In contrast, this chapter proposed an improved depthwise separable convolution. Unlike the conventional depthwise separable convolution (Chollet, 2017b), which only applies a depthwise convolution followed by a pointwise convolution without any nonlinearities, our proposed depthwise separable convolution consists of BN and Swish (Ramachandran et al., 2017) activation function.

In addition, we do not propose the use of positional encoding and iConformer in both models but only integrating it in the SM. The reason for excluding the positional encoding and iConformer in TM is because we hypothesized that the use of the Transformer layer has little effect on AT performance. This is because we believe that the sequence of an event does not affect or change the nature of an event. For example, the following two sentences, “I am happy” and “happy am I” will still belong to the Speech category.

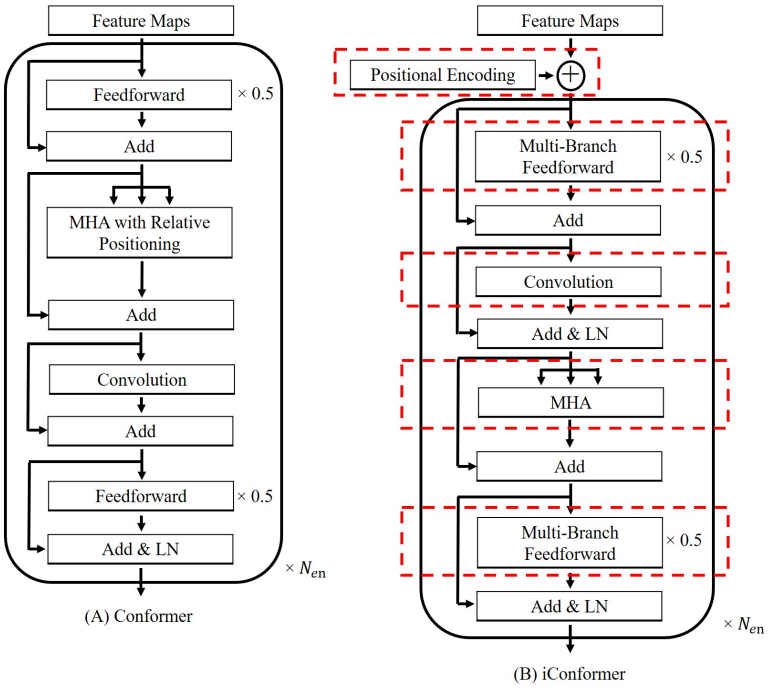


Figure 5.9 Difference between Conformer and iConformer

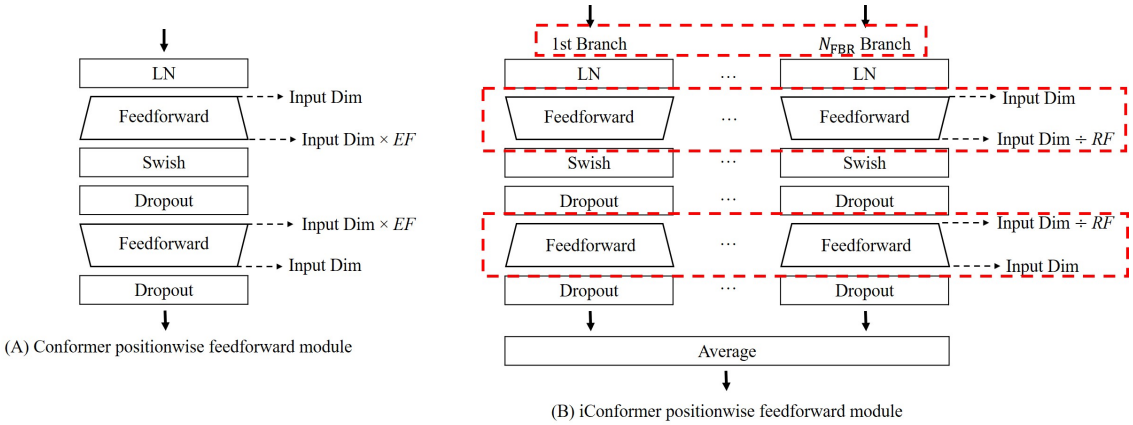


Figure 5.10 Difference between positionwise feedforward module in Conformer and iConformer (LN represents layer normalization and FF represents feedforward)

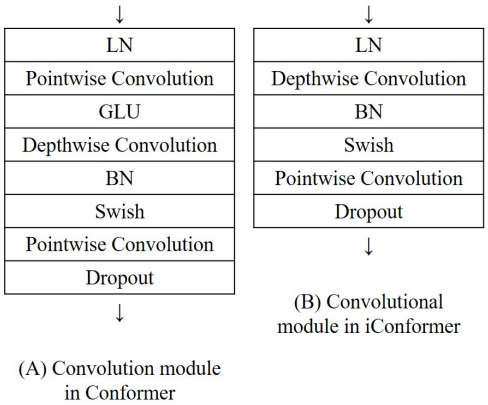


Figure 5.11 Difference between the convolutional module in Conformer and iConformer

The differences between the proposed iConformer and Conformer (Gulati et al., 2020) are highlighted with dotted red boxes from Figure 5.9 to Figure 5.11. Firstly, the relative positional encoding (Shaw et al., 2018) is utilized in Conformer (Gulati et al., 2020); however, such a module is not included in iConformer. Instead, we propose the use of the absolute positional encoding (Vaswani et al., 2017).

Secondly, we propose a multi-branch positionwise feedforward module where the differences between the modules are illustrated in Figure 5.10. In Conformer (Gulati et al., 2020), the input dimension is expanded using the first feedforward layer (the degree of expansion is denoted as Expansion Factor (EF) in Figure 5.10, and Conformer uses an EF of 4) followed by reducing the expanded dimension using the second feedforward layer. In iConformer, we reduce the input dimension using the first feedforward layer (the degree of reduction is denoted as Reduction Factor (RF) in Figure 5.10, the optimal value will be examined in the subsequent section) followed by expanding the reduced dimension using the second feedforward layer.

By reducing the input dimension instead of expanding, the number of parameters can be reduced. With the reduced number of parameters, we increase the number of positionwise feedforward modules (thereby multi-branch positionwise feedforward module) where the outputs are averaged. We hypothesize that a higher number of branches (the optimal number of branches, N_{FBR}, will be discussed in the later section) can act as a small ensemble of positionwise feedforward modules, which in turn increases the detection accuracy.

Such design is also related to (Mehta et al., 2021; Tan et al., 2021). While Mehta et al. (2021) also applied the reduction of input dimension through the feedforward layers, they did not apply the multi-branch concept, and they propose to improve Transformer (Vaswani et al., 2017) instead of Conformer (Gulati et al., 2020). On the other hand, Tan et al. (2021) applied the multi-branch concept to both the positionwise feedforward and multi-head attention modules. In their implementation, they then applied a gating mechanism where only one branch will be used. In contrast, our implementation is only applied to the feedforward module, and all branches are weighed equally.

The next difference lies in the sequence of modules. In Conformer (Gulati et al., 2020), the convolutional module is positioned after the multi-head attention module. Whereas in iConformer, the convolutional module is positional before the multi-head attention module. Such an arrangement was also proposed in (Li et al., 2021). In addition, our convolution module has lesser operations (without a pointwise convolution and GLU (Dauphin et al., 2018)).

In the following subsection, we then discuss the type of loss functions used in our semi-supervised learning framework.

5.4. Proposed Semi-Supervised Learning Framework

In this section, we present the proposed semi-supervised learning framework. The first loss component is the frame-level loss, l_{f} calculated using SBCE. Based on Equation 5.4 to Equation 5.6, l_{f} is defined as

$$l_{\mathrm{f}} = \mu l_{\mathrm{bce}} + \tau l_{\mathrm{rbce}} \tag{5.7}$$

where l_{bce} is defined as

$$\frac{1}{N_{\mathrm{e}} \times N_{\mathrm{f}}} \sum_{i=1}^{N_{\mathrm{e}}} \sum_{j=1}^{N_{\mathrm{f}}} [g_{i,j} \log(S_{i,j}) + (1 - g_{i,j}) \log(1 - S_{i,j})] \tag{5.8}$$

And l_{rbce} is defined as

$$\frac{1}{N_{\mathrm{e}} \times N_{\mathrm{f}}} \sum_{i=1}^{N_{\mathrm{e}}} \sum_{j=1}^{N_{\mathrm{f}}} [S_{i,j} \log(g_{i,j}) + (1 - S_{i,j}) \log(1 - g_{i,j})] \tag{5.9}$$

where $S_{i,j}$ represents the SM's predicted probability of event i at frame j and $g_{i,j}$ represents the ground truth of event i at frame j. μ and τ represent the hyperparameter that control the contribution of each loss term and will be given in the next section.

On the other hand, the BCE is used to calculate the clip-level loss, l_c, is defined as

$$l_{\mathrm{c}} = \frac{1}{N_{\mathrm{e}}} \sum_{i=1}^{N_{\mathrm{e}}} [z_i \log(T_i) + (1 - z_i) \log(1 - T_i)] \tag{5.10}$$

where T_i represents the TM's predicted probability of event i in an audio clip and z_i represents the grouth truth of event i in the same audio clip.

The third loss component is the interpolated consistency cost, $l_{\mathrm{i}}^{\mathrm{L}}$, calculated on labeled samples (i.e., pseudo strongly labeled data) and can be defined as

$$\mathbf{S}^{\mathrm{mc}} = M(\mathrm{SM}(\mathrm{mixup}(\mathbf{R}_1^{\mathrm{A}}, \mathbf{R}_2^{\mathrm{A}}))) \tag{5.11}$$

$$\mathbf{T}^{\mathrm{m}} = \mathrm{mixup}(\mathrm{TM}(\mathbf{R}_1^{\mathrm{A}}), \mathrm{TM}(\mathbf{R}_2^{\mathrm{A}})) \tag{5.12}$$

$$l_{\mathrm{i}}^{\mathrm{L}} = \mathrm{MSE}(\mathbf{S}^{\mathrm{mc}}, \mathbf{T}^{\mathrm{m}}) \tag{5.13}$$

where $\mathbf{R}_1^{\mathrm{A}}$ and $\mathbf{R}_2^{\mathrm{A}}$ represent the augmented feature representations of labeled sample 1 and 2, respectively. $\mathbf{S}^{\mathrm{mc}}$ represents the vector that contains the events' probabilities from SM on the mixed labeled sample. $M(.)$ represents the max pooling operation. $\mathrm{mixup}(\mathbf{R}_1^{\mathrm{A}}, \mathbf{R}_2^{\mathrm{A}})$ is defined as (Zhang et al., 2017)

$$\mathrm{mixup}(\mathbf{R}_1^{\mathrm{A}}, \mathbf{R}_2^{\mathrm{A}}) = \psi \mathbf{R}_1^{\mathrm{A}} + (1-\psi)\mathbf{R}_2^{\mathrm{A}} \tag{5.14}$$

where ψ is the mixing factor. $\mathbf{T}^{\mathrm{m}}$ represents the vector that contains the TM's interpolated predicted probabilities on a labeled sample. However, $l_{\mathrm{i}}^{\mathrm{L}}$ is regularized and extended as $l_{\mathrm{ri}}^{\mathrm{L}}$ and is defined as follow

$$l_{\mathrm{ri}}^{\mathrm{L}} = \begin{cases} \frac{1}{N_{\mathrm{e}}} \sum_{i=1}^{N_{\mathrm{e}}} (S_i^{\mathrm{mc}} - T_i^{\mathrm{m}})^2 & \text{if } \max(\mathbf{T}^{\mathrm{m}}) > \lambda_{\mathrm{curr}} \\ 0 & \text{otherwise} \end{cases} \tag{5.15}$$

where T_i^{m} represents the TM's interpolated predicted probablity for event i. λ_{curr} represents the current confidence level which will be explained in further detail in the next section.

The final loss component is also the interpolated consistency loss but applied on the unlabeled sample. This loss is represented by l_i and is defined as

$$\tilde{\mathbf{S}}^{\mathrm{mc}} = M(\mathrm{SM}(\mathrm{mixup}(\mathbf{U}_1^{\mathrm{A}}, \mathbf{U}_2^{\mathrm{A}}))) \tag{5.16}$$

$$\tilde{\mathbf{T}}^{\mathrm{m}} = \mathrm{mixup}(\mathrm{TM}(\mathbf{U}_1^{\mathrm{U}}), \mathrm{TM}(\mathbf{U}_2^{\mathrm{U}})) \tag{5.17}$$

$$l_{\mathrm{i}} = \mathrm{MSE}(\tilde{\mathbf{S}}^{\mathrm{mc}}, \tilde{\mathbf{T}}^{\mathrm{m}}) \tag{5.18}$$

where $\mathbf{U}_1^{\mathrm{A}}$ and $\mathbf{U}_2^{\mathrm{A}}$ represent the augmented feature representations of unlabeled sample 1 and 2, respectively. $\mathbf{U}_1^{\mathrm{U}}$ and $\mathbf{U}_2^{\mathrm{U}}$ represent the unaugmented feature representations of unlabeled sample 1 and 2, respectively. $\tilde{\mathbf{S}}^{\mathrm{mc}}$ represents the vector containing the SM's predicted probabilities for all events in a mixed sample. $\tilde{\mathbf{T}}^{\mathrm{m}}$ represents the interpolated predicted probabilities for all events. Similarly, l_{i} is also regularized and extended as l_{ri} and is defined as

$$l_{\mathrm{ri}} = \begin{cases} \frac{w}{N_{\mathrm{e}}} \sum_{i=1}^{N_{\mathrm{e}}} (\tilde{S}_i^{\mathrm{mc}} - \acute{T}_i^{\mathrm{m}})^2 & \text{if } \max(\acute{\mathbf{T}}_{\mathrm{m}}) > \lambda_{\mathrm{curr}} \\ 0 & \text{otherwise} \end{cases} \tag{5.19}$$

w which is a weighing parameter is given as (Laine and Aila, 2017)

$$w = \exp(-5(1-P)^2) \tag{5.20}$$

P represents the training progression and will also be discussed in further detail in the next section.

5.5. Experiment Setup

The dataset used for the subsequent experiments is the DESED 2020 dataset (Turpault et al., 2019), and the data distribution can be found in Table 3.6. As the methodology was not submitted to the DCASE challenge, results on the evaluation 2020 dataset cannot be obtained. Thus, results are only reported on the validation dataset.

Audio preprocessing and feature extraction remains similar, as described in Chapter 4.4.6. Similar to Chapter 4.4.6, the training procedure consists of two phases, 1) the warm-up phase and 2) the adaptation phase. The warm-up phase, which lasts 10 epochs, utilizes only the synthetic data and pseudo-strongly labeled data to train the models with a batch size of 32. Each batch of data is evenly split between the synthetic data and pseudo-strongly labeled data, and feature representations are augmented with Gaussian noise, time mask, and time shift. Since the unlabeled data is not utilized in this phase, the loss component only consists of l_f, l_c, and l_{ri}^{L}. μ and τ which are used to calculate l_{f} are set as 1 and 0.1, respectively. On the other hand, l_{ri}^{L} is affected by λ_{curr} which is defined as

$$\lambda_{\text{curr}} = \lambda_{\text{min}} + 0.5(\lambda_{\text{max}} - \lambda_{\text{min}})(1 + \cos(P\pi)) \tag{5.21}$$

where λ_{max} represents the maximum confidence threshold and is set as 0.9. λ_{min} represents the minimum confidence threshold and is set as 0.6. P, which represents the training progression is defined as

$$P = \frac{P_{\text{w}} - P_{\text{curr}}}{P_{\text{w}}} \tag{5.22}$$

P_{w} represents the total iterations during the warm-up phase, which we set as the total number of iterations in 10 epochs. P_{curr} represents the current confidence threshold. Thus, on the definition of P, the confidence threshold will increase progressively from 0.6 to 0.9.

Learning rate is also increased progressively along a cosine curve which is defined as

$$LR_{\text{curr}} = LR_{\text{min}} + 0.5(LR_{\text{max}} - LR_{\text{min}})(1 + \cos(P\pi)) \tag{5.23}$$

where LR_{max} and LR_{min} are set as 0.0012 and 1e-6, respectively. Based on the calculated losses, SM is updated using Adam (Kingma and Ba, 2015), and TM is updated using Adabelief (Zhuang et al., 2020).

The adaptation phase (i.e., from the 11th epoch onwards) utilizes synthetic data, pseudo strongly labeled data, and unlabeled data to train the models with a batch size of 64. Each batch of data is split into the following proportion: 25% synthetic data, 25% pseudo strongly labeled data, and 50% weakly labeled data. Similar to the warm-up phase, feature representations are augmented with Gaussian noise, time mask, and time shift. With the inclusion of unlabeled data, the loss component in this phase would consist of l_f, l_c, l_{ri}^{L} and l_{ri}. μ and τ which are used to calculate l_f remain as 1.0 and 0.1, respectively. In the adaptation phase, we adopt the improved cyclic learning scheme as proposed in Chapter 4. Thus, P, which represents the training progression, becomes a critical component that affects the calculation of LR_{curr}, λ_{curr} and w.

In the adaptation phase, P has two states. At the start of the adaptation phase, P_{curr} is reset to 0, and the learning rate is set to decrease progressively from LR_{max} to LR_{min}. Thus, P is defined as

$$P = \frac{P_{\text{curr}}}{P_{\text{tot}}} \tag{5.24}$$

where P_{tot} represents the maximum training iterations before a change in state (i.e., from decreasing learning rate to increasing learning rate). Based on our proposed cyclic learning scheme in Chapter 4, when $P_{\text{curr}} = P_{\text{tot}}$, the learning rate is set to increase from the minimum to the maximum and P_{curr} will revert back to 0. Thus, when the learning rate is increasing, P is defined similarly to Equation 5.22. On the other hand P_{tot} is multiplied with an integer, P_{mult}, which can delay the next change of state (i.e., from decreasing learning rate to increasing learning rate) if P_{mult} is larger than 1. Note that our proposed cyclic learning scheme does not multiply P_{w} with P_{mult}. Thus during the state of increasing learning rate, LR_{min} will always transit back to LR_{max} in a fixed number of iterations. The reader may refer to Figure 4.12 and Figure 4.13 for more information.

In this phase, LR_{max} and LR_{min} remains as 0.0012 and 1e-6, respectively. λ_{max} and λ_{min} will also remain at 0.6 and 0.6. P_{tot} is set as 1 epoch, P_{mult} is set as 2 while P_{w} is set as 1 epoch. The cyclic learning phase will end when P_{tot} reaches 64 epoch. This effectively means the total number of epochs in this phase is 133 epochs. Similar to the warm-up phase, SM is updated using Adam (Kingma and Ba, 2015), and TM is updated using Adabelief (Zhuang et al., 2020).

In the inference stage, both models are used for audio tagging and SED. The method of detection and post-processing steps remain similar, as described in Chapter 4.3.2.

In this section, all experiments were conducted on a system using an AMD Ryzen Processor 5900x with a base frequency of 3.7GHz, 32GB ram and a RTX3090 GPU.

		Kernel size			
		3	7	15	31
Filter size	128	**52.0**	49.6	49.0	48.5
	256	50.2	49.6	48.1	45.4
	512	51.4	49.2	49.1	46.2

Table 5.1 Effects of filter size and kernel size on event-based F1-score (%)

		N_{FBR}			
		1	2	4	8
RF	1	49.2	50.2	49.5	51.1
	2	49.4	49.8	49.5	49.6
	4	49.7	49.7	**52.0**	50.1
	8	49.5	50.2	51.6	49.7

Table 5.2 Effects of N_{FBR} and *RF* on event-based F1-score (%)

5.6. Results and Discussion

In this section, various experiments and ablation studies are carried out to examine different aspects of the proposed model. The primary evaluation metric is the event-based F1-score (Mesaros et al., 2016) which determines the optimal setting and hyperparameters.

The first experiment we conducted investigates the effect of filter size and kernel size of the convolution module in iConformer. As seen in Table 1, using a larger number of filters generally does not produce a higher accuracy. However, it can be observed that a large kernel size usually reduces the detection accuracy. As mentioned in (Xia et al., 2020), smaller kernels are preferred when input features have high variability, while larger kernels are more suitable for input features with low variability. An audio clip can contain multiple events that may overlap or similar events but with different sound characteristics (i.e., the barking sound of a chihuahua and husky is different). Therefore, the time-frequency representation of an audio clip can be considered to be a high variability input. Thus, this can explain why the use of smaller kernel size produces better accuracy than bigger kernel size.

We then examine the effect of N_{FBR} and *RF* on the detection accuracy. Although it was hypothesized that increasing the dimension of the input through the positionwise feedforward module may increase a Transformer's expressivity and capacity (Mehta et al.,

	Event-based F1-score (%)
Additional pointwise convolution and GLU	51.0
Proposed	**52.0**

Table 5.3 Importance of additional pointwise convolution and GLU

2021) but, as seen in Table 5.2, the reduction of the input dimension does not reduce the accuracy drastically, but, can in fact, increase the accuracy marginally.

We hypothesized that the macaron-style positionwise feedforward modules (two positionwise feedforward modules with half-step residual connection) (Lu et al., 2019) might play a part in this phenomenon. As suggested in (Lu et al., 2019), the use of only one positionwise feedforward module can bring bias and leads to higher local truncation error. However, it can be mitigated if two positionwise feedforward modules with half-step residual connection are used. Such arrangement can then lead to higher order accuracy in terms of truncation error. We hypothesized that this in turn allows the input dimension to be reduced without sacrificing accuracy. Also, in (Lu et al., 2019), the expansion factor in the Macaron Net encoding layer was set to half of the original Transformer (Vaswani et al., 2017) (in order to use the same number of parameters) and was found to perform better in various dataset. This may also suggest that expanding the input dimension may not be necessary.

As seen in Table 5.2, while multi-branch positionwise feedforward module can generally produce higher accuracy than using only a single branch positionwise feedforward module, we find that the increment in accuracy is only marginally in most cases and increasing N_{FBR} may not necessarily produce a higher accuracy (i.e., N_{FBR}=8 may not perform as well as N_{FBR}=4). We hypothesize that the multi-branch positionwise feedforward module can be improved by using different activation functions or RF in each branch. This promotes diversity and can allow different characteristics to be learned, which may increase the accuracy. Such an idea is similar to the triple instance-level pooling (Chan and Chin, 2021). Based on the results in Table 5.2, the best result is obtained by setting N_{FBR} and RF as 4, therefore, we continue with the same setting.

We then investigate the effects of ablating the pointwise convolution and GLU (Dauphin et al., 2018) in the convolution module of iConformer (refer to Figure 5.9). Based on the results shown in Table 5.3, the two operations can be safely ablated without sacrificing accuracy.

The effect of using a different number of encoding layers and heads is then examined. Based on the results shown in Table 5.4, the accuracy does not always improve with

		No. layers				
		1	2	3	4	Ablate encoding layer
No. heads	1	**52.0**	48.9	48.7	50.1	48.0
	4	49.8	49.1	49.4	49.2	
	8	48.8	48.1	49.6	50.6	
	16	50.5	48.9	50.1	50.3	

Table 5.4 Accuracy using different number of encoding layers and heads

Type of positional encoding	Event-based F1-score (%)
None	49.6
Absolute	**52.0**
Relative	46.9
Absolute + Relative	45.7

Table 5.5 Effect of using different positional encoding

additional layers or additional heads. Also, some settings only provide marginal gain as compared to a SED system trained without iConformer.

Such a phenomenon may be due to the fact that very deep Transformer can be difficult to train. Moreover, as explained in (Michel et al., 2019), multiple heads may not always leverage its theoretically superior expressiveness over a single head to the fullest extent.

As the accuracy of a SED system can be improved by up to 4% using iConfomer, such a module may still be considered to be necessary for onset-offset estimation.

We then examine the use of different positional encoding. Based on the results shown in Table 5.5, the use of absolute positional encoding (sinusoidal positional encoding) (Vaswani et al., 2017) produces the best result. In our case, learnable positional encoding (Shaw et al., 2018) does not provide any benefits. Similar to the results shown in (Shaw et al., 2018), we find that combining two encoding modules does not provide any further improvement and would produce a lower accuracy compared to using only one mode of positional encoding. Although the convolution module in the encoding layer may implicitly provide relative positional information (Li et al., 2021), accuracy can be improved with absolute positional encoding.

We then compared the performance of iConfomer and Conformer in the SED domain. In this comparison, the settings of Conformer are set according to the illustrations given in Figure 5.9 to Figure 5.11, with an EF of 4 in the positionwise feedforward module, a kernel size of 3 in the convolutional module. As shown in Table 5.6, Conformer with

	Event-based F1-score (%)
iConformer	**52.0**
Conformer (EF=4, Relative)	46.3
Conformer (EF=1, Absolute)	49.0

Table 5.6 Comparison between iConformer and Conformer

	Event-based F1-score (%)
Proposed depthwise-separable	**52.0**
Ablate Swish Retain BN	47.4
Retain Swish Ablate BN	48.0
Ablate Swish and BN	47.8
Conventional convolution	49.6

Table 5.7 Analysis on depthwise-separable module

	AT F1-score (%)
Without encoding layer	**79.2**
With encoding layer	77.1

Table 5.8 Importance of encoding layer for AT model

the original setting performs poorly as compared to iConformer. Based on the earlier experiments, we deduced that the causes are due to the large EF used in the positionwise feedforward module and the relative positional encoding. We then set EF as 1, and proceed with the use of absolute positional encoding instead of relative position encoding. Based on the proposed setting, Conformer performs much better but still has a margin of 3% against iConformer.

As seen in Table 5.7, due to the reduction in parameters, conventional convolution can outperform depthwise separable convolution, even though it was hypothesized that mapping cross channel correlations and spatial correlations separately is more efficient than mapping them at once (Chollet, 2017b). In our experiment, the efficiency of mapping cross channel correlations and spatial correlations separately can be increased if nonlinearities and normalization are added, which leads to higher accuracy.

One interesting finding is that the use of nonlinearities should be accompanied by normalization and vice versa; otherwise, there is little to no performance gain compared to the conventional depthwise separable module.

		μ				
		0.01	0.05	0.1	0.5	1
τ	0	33.7	38.9	44.9	47.2	49.3
	0.1	33.1	40.3	46.0	49.3	**52.0**

Table 5.9 Effects of different μ

		μ
		1
τ	0	49.3
	0.01	50.1
	0.05	50.5
	0.1	**52.0**
	0.5	50.0
	1	48.5

Table 5.10 Effects of different τ

As it was hypothesized that encoding layer use does not help improve the AT performance, we conducted an experiment to verify this hypothesis. As seen in Table 5.8, the encoding layer can be safely removed from the TM without the risk of a drastic performance drop. In fact, removing the encoding layer can make the model less difficult to train, and, in our experiment, the AT F1-score is shown to be better by excluding the use of the encoding layer in TM.

We then investigate the effects of different μ and τ in the range of [0.0,1.0]. It should be pointed out that when $\tau = 0$, this becomes the BCE. It can be observed in Table 5.9 that accuracy is low when $\mu = 0.01$, which indicates poor convergence (this is also observed in (Wang et al., 2019b)). The addition of RBCE does not provide any benefit at low μ. However, when the value of τ increases, an improvement with the addition of RBCE can be seen.

As seen in Table 5.10, the benefit of adding RBCE may not reach its full potential when τ is small, and the maximal gain occurs when $\tau = 0.1$. However, it can also be observed that accuracy can start to decrease when τ is above 0.1. Such results suggest the need to finetune these two values.

A comparison of the proposed loss function is made against the theoretical noise-robust Mean Abosolute Error (MAE) (Ghosh et al., 2017). The results shown in Table 5.11 indicate that the use of MAE is suboptimal, which is consistent with the finding

Loss function	Event-based F1-score (%)
Proposed	**52.0**
MAE	22.0

Table 5.11 Loss functions use for calculating frame-level loss

Loss function	Event-based F1-score (%)
With l_{rc} and l_{ru}	**52.0**
Ablate l_{ru}	49.0
Ablate l_{rc} and l_{ru}	47.1

Table 5.12 Importance of l_{rc} and l_{ru}

in (Fonseca et al., 2019). Studies suggested that the cause of this issue might be due to gradient saturation which can be challenging to use for model training (Ghosh et al., 2017; Wang et al., 2019b).

We then conduct an ablation study on the importance of l_{rc} and l_{ru}. Results in Table 5.12 show that ablating them would result in a drop in accuracy and are considered essential components in our training scheme.

5.7. Comparison against SOTA

We then compare our system with the SOTA. We first compare the accuracy for non-ensembled systems. As shown in Table 5.13, Kim and Kim system (Kim and Kim, 2021) is considered the best in the literature. However, our system can achieve a higher event-based F1-score of 1.4%.

Compared to the first-place submissions in DCASE 2020 challenge task 4 (Miyazaki et al., 2020), our system outperforms them by 6%. Compared to the second-place (Yang et al., 2020) and third-place submissions (Ebbers and Haeb-Umbach, 2020), our system outperforms them by 3.7% and 5.6%, respectively.

Finally, by comparing our system against the DCASE 2020 challenge task 4 baseline systems (Turpault and Serizel, 2020; Turpault et al., 2020b), we can have a winning margin of over 16%. In terms of models parameters used, the SM model utilized 245,595 parameters, and TM model utilized 263,634 parameters. The total number of parameters used is 509,229 parameters.

Despite using two models to perform SED, our system utilized the least number of parameters despite utilizing two models for inference. One provides the frame-level prediction, and the other provides the audio tags.

We then compare the performance of the ensembled systems. We combined four models by averaging the posterior probabilities. Note that all models are trained using the same setting that produced the best results in the previous analysis.

Compared with Kim and Kim system (Kim and Kim, 2021), our system can win by 1.9%. As only the first-place (Miyazaki et al., 2020) and third-place (Ebbers and Haeb-Umbach, 2020) submissions have the ensembled system, we only compared our ensembled system against them. Compared to the first-place (Miyazaki et al., 2020) and third-place (Ebbers and Haeb-Umbach, 2020) ensembled system, we can still maintain a winning margin of 2.9% and 4.3%, respectively.

By ensembling models, model parameters can increase drastically. In our case, the ensemble of four models would result in 2,036,916 parameters. The number of parameters utilized by our ensembled system may still be considered low since it is still comparable to the non-ensembled system proposed by the top-3 submissions in DCASE 2020 challenge task 4 (Ebbers and Haeb-Umbach, 2020; Miyazaki et al., 2020; Yang et al., 2020).

However, for the ensembled system proposed by Miyazaki et al. (2020) and Ebbers and Haeb-Umbach (2020), the total number of parameters used are 17M and 20M, respectively. Compared to our ensembled system, they are using up to 18M more parameters.

Methodology	Event-based F1 score (%)		No. of parameters	
	Non-ensembled	Ensembled	Non-ensembled (M)	Ensembled (M)
Proposed	**52.0**	**53.5**	**0.51**	**2**
Conformer (Miyazaki et al., 2020) 1st DCASE 2020	46.0	50.6	2	17
CRNN (Yang et al., 2020) 2nd DCASE 2020	48.3	-	2	-
FBCRNN (Ebbers and Haeb-Umbach, 2020) 3rd DCASE 2020	46.4	49.2	2	20
Baseline CRNN (Turpault and Serizel, 2020)	34.8	-	1	-
Baseline CRNN With Source Separation (Turpault et al., 2020b)	35.6	-	1	-

Table 5.13 Comparison of system against other SOTA

5.8. Summary

In this chapter, we present a lightweight system for polyphonic SED. The key idea is to use the depthwise separable convolution, which factorizes the conventional convolution into a depthwise convolution followed by a pointwise convolution. However, we found that such implementation may yield a worse accuracy than a conventional convolution, where the most probable reason is the significant reduction in parameters. Nevertheless, we found that such an issue can be mitigated by adding nonlinearities and BN.

This chapter also proposes iConformer, which further improves on Conformer by removing the redundant modules and the inclusion of multi-branch positionwise feedforward module. While multi-branch positionwise feedforward module can generally produce higher accuracy than using only a single branch positionwise feedforward module, we find that the increment in accuracy is only marginally in most cases. We hypothesize that this can be improved by using different activation functions or RF in each branch, which promotes diversity. Such an implementation can allow different characteristics to be learned, which may further increase accuracy.

Finally, we also explore the use of SBCE, which is the extension of BCE by considering RBCE. Based on the results, the use of SBCE to calculate the frame-level loss does help to increase the system accuracy, which showcases its potential. However, we found that the weighing parameters for BCE and RBCE have to be tuned appropriately; otherwise, it can degrade the accuracy.

Based on our proposed ideas and framework, our lightweight system can obtain an event-based F1-score of 52%, and the ensemble of four systems can further improve the accuracy to 53.5%. Such results indicate a minimum margin of 16% against the DCASE 2020 challenge task 4 baseline system. Compared to the first-place system of the DCASE 2020 challenge task 4, our non-ensembled system can achieve a higher event-based F1-score of 6% with 75% lesser parameters. In terms of the performance of the ensembled system, our system remains competitive and has a winning margin of 2.9% despite using 15 million lesser parameters. Comparison with other state-of-the-art also indicates that our system performance is better despite using a lightweight system.

Chapter 6. Conclusion

6.1. Summary and Contributions

In this thesis, the topic of SED was studied. An ideal SED system is a system that can accurately predict the presence of an event as well as the annotation of the identified event's onset and offset. However, the development of a SED system is by no means trivial and can be hindered by many different obstacles. In the following paragraphs, we listed various issues or problems faced and possible room for improvement during the model development phase and provided a point-by-point summarized contribution made in this thesis.

The first and foremost is the lack of strongly labeled data. While the use of weakly labeled data can help alleviate this issue, it was found that strongly labeled data can improve the accuracy of audio classification (Hershey et al., 2021), which infers the need for strongly labeled data to maximize a SED system's performance.

- To this end, we propose a pseudo-labeling method using NMF to provide a pseudo strong label for a weakly labeled audio clip. In Chapter 3, we demonstrate that the use of pseudo strong labels can effectively increase the accuracy of the frame-level prediction.

As mentioned earlier, weakly labeled data can be an alternative to train a SED system. However, most of the SOTA utilized the Mean-Teacher approach (Tarvainen and Valpola, 2017), which requires training two identical models in a semi-supervised manner. Such methodology can have two critical limitations. Firstly, it can be computationally expensive if a very deep model is designed. Secondly, a model designed might only be optimal for either audio tagging or frame-level prediction but not both.

- In order to achieve optimal performance at both subtasks, in Chapter 3, we propose training two different models synchronously using a novel student-teacher framework so that the developed SED system is adept at both audio tagging and temporal localization. By comparing against the SOTA, our system is shown to be competitive.

With the proof of concept that pseudo strong labels can effectively increase the accuracy of temporal localization, we further improve the quality of pseudo labeling. At the same time, we investigate the effectiveness of adding the Macaron Net encoding layer (Lu et al., 2019) into our system, which was found to perform relatively well in several speech-related tasks.

- To improve the quality of pseudo labels, we propose an improved pseudo labeling method using supervised CNMF, which is demonstrated to be better than NMF in Chapter 4. In addition, extensive experiments were also carried out to investigate various aspects of a Macaron Net. We found that there is no need to implement a large Macaron Net with many heads to achieve SOTA performance.

As our proposed framework utilized two completely different models for SED, this may increase the overall model and hyperparameter tuning time. Thus, it would be ideal if the design of models can be simplified and yet maintain the level of competitiveness against the SOTA.

- In Chapter 4, we improve on the design of our framework by proposing a more straightforward design for the two models where the differences only lie in the number of convolutional layers, pooling size, and temporal pooling method. Together with our newly proposed ideas, such as the use of meta-ACON, SE module, an improved Macaron Net encoding layer, triple instance-level pooling approach, and improved cyclic learning scheme, we show that even with a straightforward design, it can still be effective and competitive against the SOTA.

While the use of deep learning models can be considered a norm in the current research landscape, it should be pointed out that deep learning models are highly parameterized. A very deep model not only requires multiple dedicated hardware for training purposes but can also face deployment issues in a resource-constrained environment. As mentioned earlier, strongly labeled data may be required to maximize the potential of a SED system; however, annotation of events' timestamps is prone to error and prone to disagreement due to the difference in perception of when the onset and offset should be. Thus, strongly labeled data will inevitably contain a certain level of noise.

- Chapter 5 proposes a lightweight system for SED, which comprises an improved depthwise separable convolution and an improved Conformer encoding layer. At the same time, to combat label noise, we propose the extension of BCE with RBCE. Our extensive experiments show that our lightweight system can outperform the SOTA despite using a much lesser number of parameters.

6.2. Future Work

This thesis aims to address 1) the issue caused by the lack of strongly labeled data and 2) the issues caused by the use of the Mean-Teacher approach (Tarvainen and Valpola, 2017). Although we have proposed several proposals to address the issues mentioned above, there are various research directions that can be extended from the work presented in this thesis. These include:

1. **Improved granularity of detection.** In this thesis, we tested our proposed ideas on the DESED dataset (Turpault et al., 2019) which comprises sound events commonly heard in a domestic environment. Each event class has a general label that classifies all sounds made by the subject or object in the same category (i.e., a dog barking and a dog growling is grouped under the Dog category). Naturally, a SED system trained based on the given labels can only predict the general label of the sound made. However, a SED can be more useful if the detection granularity is improved. One would know precisely the type of scenario that had happened and act accordingly. For example, a SED system can trigger a distress call when it detected an old lady who fell in the bathroom groaning in pain instead of classifying the sound made by the old lady as Speech. This is an exciting direction to determine how a detected event can be further segregated into a more precise event even if such labels are not present in the training labels.

2. **Deployment of the SED system into real-life application.** In a real-life scenario, a SED system is more likely to operate on sound segments that are longer than 10s, where the sound event density will be much lower than the sound density of the validation clips. This would indicate that a considerable segment of a real-life recording may not contain any events at all. In addition, an event recorded may occur far away from the microphone deployed. It would be interesting to find out the system's robustness in low event density audio clips and how well the system can perform on events with varying loudness.

3. **The generation of synthetic audio samples using GAN (Goodfellow et al., 2014).** GAN (Goodfellow et al., 2014) is currently one of the dominant paradigms for generating virtual images, which are almost indistinguishable from real images (Binkowski et al., 2019). It may also be the next SOTA to produce synthetic audio samples that are almost indistinguishable from real audio samples. This direction

can effectively alleviate the problem caused by a lack of strongly labeled data and also be a valuable technique to rebalance an imbalanced dataset.

4. **Further improvement to the SED system.** Although the proposed system can achieve the SOTA performance based on the event-based F1-score. It should be noted that the SOTA event-based F1-score is still at the 40% to 50% range. Compared to the audio tagging F1-score, which is generally in the 70% to 80% range, there is still a large room for improvement for temporal localization. As a start, we could further improve on the multi-branch feedforward module in the proposed iConformer. The use of different activation functions in each branch can be attempted, which promotes diversity and may allow different characteristics to be learned, increasing accuracy.

   Another interesting direction would be the incorporation of source separation, which was demonstrated in (Heittola et al., 2013b; Turpault et al., 2020b). Heittola et al. (2013b) proposed the use of NMF Lee and Seung (1999) for source separation, which acts as a preprocessing step before feature extraction. In contrast, Turpault et al. (2020b) proposed using a universal source separation algorithm (Kavalerov et al., 2019; Tzinis et al., 2020) and integrated it into a SED system in three different manners. Both studies (Heittola et al., 2013b; Turpault et al., 2020b) suggest that source separation has the potential to improve the SED performance.

   Finally, the exploration of CapsNet as the SED system. As mentioned earlier, inter-capsules are connected through a process known as dynamic routing. This can be viewed as a parallel attention mechanism that allows each capsule at one level to attend to some active capsules at the level below and to ignore others (Sabour et al., 2017). Such a process is hypothesized to allow the model to recognize multiple objects in the image even if objects overlap. The issue of overlapping objects in the image resembles the SED problem, where multiple events can coincide. Therefore, the use of CapsNet may very well address the overlapping issues, and as demonstrated in (Iqbal et al., 2018; Vesperini et al., 2019), the use of CapsNet can be promising.

References

Adavanne, S., Parascandolo, G., Pertila, P., Heittola, T., and Virtanen, T. (2016). Sound event detection in multichannel audio using spatial and harmonic features. In *Proceedings of the Workshop on Detection and Classification of Acoustic Scenes and Events*, pages 1–5, Budapest, Hungary.

Adavanne, S., Pertila, P., and Virtanen, T. (2017). Sound event detection using spatial features and convolutional recurrent neural network. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 771–775, New Orleans, LA, USA.

Adavanne, S., Politis, A., and Virtanen, T. (2018). Multichannel sound event detection using 3d convolutional neural networks for learning inter-channel features. In *Proceedings of the International Joint Conference on Neural Networks*, pages 1–7, Rio de Janeiro, Brazil.

Baker, J. M., Deng, L., Glass, J., Khudanpur, S., hui Lee, C., Morgan, N., and O'Shaughnessy, D. (2009). Developments and directions in speech recognition and understanding, part 1. *IEEE Signal Processing Magazine*, 26(3):75–80.

Bello, I., Zoph, B., Vaswani, A., Shlens, J., and Le, Q. V. (2019). Attention augmented convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3286–3295, Seoul, South Korea.

Binkowski, M., Donahue, J., Dieleman, S., Clark, A., Elsen, E., Casagrande, N., Cobo, L. C., and Simonyan, K. (2019). High fidelity speech synthesis with adversarial networks. *arXiv preprint arXiv:1909.11646*, pages 1–15.

Bisot, V., Essid, S., and Richard, G. (2017). Overlapping sound event detection with supervised nonnegative matrix factorization. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 31–35, New Orleans, LA, USA.

Bregman, A. (1990). *Auditory Scene Analysis: The Perceptual Organization of Sound*. MIT Press, London.

Bui, M.-Q., Duong, V.-H., Mathulaprangsan, S., Pham, B.-T., Lee, W.-J., and Wang, J.-C. (2016). A survey of polyphonic sound event detection based on nonnegative matrix factorization. In *Proceedings of the International Computer Symposium*, pages 351—354, Chiayi, Taiwan.

Cai, M., Shi, Y., and Liu, J. (2014). Stochastic pooling maxout networks for low-resource speech recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 3266–3270, Florence, Italy.

Cakir, E., Heittola, T., Huttunen, H., and Virtanen, T. (2015a). Multi-label vs. combined single-label sound event detection with deep neural networks. In *Proceedings of the 23rd European Signal Processing Conference*, pages 2551–2555, Nice, France.

Cakir, E., Heittola, T., Huttunen, H., and Virtanen, T. (2015b). Polyphonic sound event detection using multi label deep neural networks. In *Proceedings of the International Joint Conference on Neural Networks*, pages 1–7, Killarney, Ireland.

Cakir, E., Parascandolo, G., Heittola, T., Huttunen, H., and Virtanen, T. (2017). Convolutional recurrent neural networks for polyphonic sound event detection. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(6):1291–1303.

Carreira-Perpinan, M. A. and Hinton, G. E. (2005). On contrastive divergence learning. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pages 33–40, Bridgetown, Barbados.

Castaneda, G., Morris, P., and Khoshgoftaar, T. M. (2019). Evaluation of maxout activations in deep learning across several big data domains. *Journal of Big Data*, 6(72):992–1006.

Chan, T. K. and Chin, C. S. (2019). Health stages diagnostics of underwater thruster using sound features with imbalanced dataset. *Neural Computing and Applications*, 31:5767–5782.

Chan, T. K. and Chin, C. S. (2020). A comprehensive review of polyphonic sound event detection. *IEEE Access*, 8:103339–103373.

Chan, T. K., Chin, C. S., and Li, Y. (2019). Non-negative matrix factorization-convolutional neural network (nmf-cnn) for sound event detection. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2019 Workshop*, pages 40–44, New York, NY, USA.

Chaudhary, M., Prakash, V., and Kumari, N. (2018). Identification vehicle movement detection in forest area using mfcc and knn. In *Proceedings of the International Conference on System Modeling and Advancement in Research Trends*, pages 158–164, Moradabad, India.

Chen, P., Ye, J., Chen, G., Zhao, J., and Heng, P.-A. (2020). Robustness of accuracy metric and its inspirations in learning with noisy labels. *arXiv preprint arXiv:2012.04193*, pages 1–11.

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1724–1734, Doha, Qatar.

Chollet, F. (2017a). *Deep Learning with Python*. Manning Publications.

Chollet, F. (2017b). Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1800–1807, Honolulu, HI, USA.

Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Learning phrase representations using rnn encoder–decoder for statistical machine translation. *arXiv preprint arXiv:1412.3555*, pages 1–9.

Clavel, C., Ehrette, T., and Richard, G. (2005). Events detection for an audio-based surveillance system. In *Proceedings of the IEEE International Conference Multimedia and Expo*, pages 1–4, Amsterdam, Netherlands.

Dauphin, Y. N., Fan, A., Auli, M., and Grangier, D. (2018). Language modeling with gated convolutional networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 933–941, Sydney, Australia.

Delphin-Poulat, L., Nicol, R., Plapous, C., and Peron, K. (2020). Comparative assessment of data augmentation for semi-supervised polyphonic sound event detection. In *Proceedings of the 27th Conference of Open Innovations Association*, pages 46–53, Trento, Italy.

Dettmers, T. (2020). Which gpu(s) to get for deep learning: My experience and advice for using gpus in deep learning. https://timdettmers.com/2020/09/07/which-gpu-for-deep-learning/.

Diment, A., Heittola, T., and Virtanen, T. (2013). Sound event detection for office live and office synthetic aasp challenge. Technical report.

Ding, W. and He, L. (2020). Adaptive multi-scale detection of acoustic events. *IEEE/ACM Transactions on Audio, Speech and Language Processing*, 28:294–306.

Du, J., Tu, Y., Xu, Y., Dai, L., and Lee, C.-H. (2014). Speech separation of a target speaker based on deep neural networks. In *Proceedings of the 12th International Conference on Signal Processing*, pages 473–477, Hangzhou, China.

Ebbers, J. and Haeb-Umbach, R. (2020). Forward-backward convolutional recurrent neural networks and tag-conditioned convolutional neural networks for weakly labeled semi-supervised sound event detection. In *Proceedings of the Workshop on Detection and Classification of Acoustic Scenes and Events*, pages 41–45, Tokyo, Japan.

Ferroni, G., Bonfigli, R., Principi, E., Squartini, S., and Piazza, F. (2015). A deep neural network approach for voice activity detection in multi-room domestic scenarios. In *Proceedings of the International Joint Conference on Neural Networks*, pages 1–8, Killarney, Ireland.

Florentin, J., Dutoit, T., and Verlinden, O. (2016). Identification of european woodpecker species in audio recordings from their drumming rolls. *Ecological Informatics*, 35:61–70.

Fonseca, E., Favory, X., Pons, J., Font, F., and Serra, X. (2020). Fsd50k: an open dataset of human-labeled sound events. *arXiv preprint arXiv:2010.00475*, pages 1–24.

Fonseca, E., Plakal, M., Ellis, D. P. W., Font, F., Favory, X., and Serra, X. (2019). Learning sound event classifiers from web audio with noisy labels. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 21–25, Brighton, UK.

Forney, G. D. (1973). The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278.

Gales, M. and Young, S. (2007). The application of hidden markov models in speech recognition. *Foundations and Trends in Signal Processing*, 1(3):195–304.

Gemmeke, J. F., Ellis, D. P. W., Freedman, D., Jansen, A., Lawrence, W., Moore, R. C., Plakal, M., and Ritter, M. (2017). Audio set: An ontology and human-labeled dataset for audio events. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 776–780, New Orleans, LA, USA.

Gencoglu, O., Virtanen, T., and Huttunen, H. (2014). Recognition of acoustic events using deep neural networks. In *Proceedings of the 22nd European Signal Processing Conference*, pages 506–510, Lisbon, Portugal.

Ghosh, A., Kumar, H., and Sastry, P. S. (2017). Robust loss functions under label noise for deep neural networks. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, pages 1919–1925, San Francisco, California, USA.

Gong, M., Xu, Y., Li, C., Zhang, K., and Batmanghelich, K. (2019). Twin auxiliary classifiers gan. In *Proceedings of the 33rd Conference on Neural Information Processing Systems*, pages 1–10, Vancouver, Canada.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.

Goodfellow, I., Warde-Farley, D., Mirza, M., Courville, A., and Bengio, Y. (2013). Maxout networks. In *Proceedings of the 30th International Conference on Machine Learning*, pages 1319–1327, Atlanta, GA, USA.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Proceedings of the 27th Conference on Neural Information Processing Systems*, pages 2672–2680, Montreal, Canada.

Graves, A., Jaitly, N., and rahman Mohamed, A. (2013). Hybrid speech recognition with deep bidirectional lstm. In *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 273–278, Olomouc, Czech Republic.

Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., and Schmidhuber, J. (2015). Lstm: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, 28(10):2222–2232.

Grollmisch, S., Abeber, J., Liebetrau, J., and Lukashevich, H. (2019). Sounding industry: Challenges and datasets for industrial sound analysis. In *Proceedings of the 27th European Signal Processing Conference*, pages 1–5, A Coruna, Spain.

Gulati, A., Qin, J., Chiu, C.-C., Parmar, N., Zhang, Y., Yu, J., Han, W., Wang, S., Zhang, Z., Wu, Y., and Pang, R. (2020). Conformer: Convolution-augmented transformer for speech recognition. In *Proceedings of the Interspeech*, pages 5036–5040, Shanghai, China.

Hayashi, T., Watanabe, S., Toda, T., Hori, T., Roux, J. L., and Takeda, K. (2016). Bidirectional lstm-hmm hybrid system for polyphonic sound event detection. Technical report.

Heck, M., Sakti, S., and Nakamura, S. (2016). Iterative training of a dpgmm-hmm acoustic unit recognizer in a zero resource scenario. In *Proceedings of the IEEE Spoken Language Technology Workshop*, pages 57–63, San Diego, CA, USA.

Heittola, T., Mesaros, A., Eronen, A., and Virtanen, T. (2013a). Context-dependent sound event detection. *EURASIP Journal on Audio, Speech, and Music Processing*, 2013(1):1–13.

Heittola, T., Mesaros, A., Virtanen, T., and Gabbouj, M. (2013b). Supervised model training for overlapping sound events based on unsupervised source separation. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8677–8681, Vancouver, BC, Canada.

Hershey, S., Ellis, D. P. W., Fonseca, E., Jansen, A., Liu, C., Moore, R. C., and Plakal, M. (2021). The benefit of temporally-strong labels in audio event classification. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 366–370, Toronto, ON, Canada.

Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-R., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., and Kingsbury, B. (2012). Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal Processing Magazine*, 29(6):82–97.

Hinton, G., Sabour, S., and Frosst, N. (2018). Matrix capsules with em routing. In *Proceedings of the 6th International Conference on Learning Representations*, pages 1–15, Vancouver, BC, Canada.

Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, pages 1–9.

Hoffman, M. D. (2012). Poisson-uniform nonnegative matrix factorization. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 5361–5364, Kyoto, Japan.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, pages 1–9.

Hu, J., Shen, L., Albanie, S., Sun, G., and Wu, E. (2020). Squeeze-and-excitation networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(8):2011–2023.

Huang, Y., Lin, L., Ma, S., Wang, X., Liu, H., Qian, Y., Liu, M., and Ouchi, K. (2020a). Guided multi-branch learning systems for sound event detection with sound separation. In *Proceedings of the Workshop on Detection and Classification of Acoustic Scenes and Events*, pages 61–65, Tokyo, Japan.

Huang, Y., Wang, X., Lin, L., Liu, H., and Qian, Y. (2020b). Multi-branch learning for weakly-labeled sound event detection. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal*, pages 641–645, Barcelona, Spain.

Iqbal, T., Xu, Y., Kong, Q., and Wang, W. (2018). Capsule routing for sound event detection. In *Proceedings of the 26th European Signal Processing Conference*, pages 2255–2259, Rome, Italy.

Jiang, X., Wang, Y., Liu, W., Li, S., and Liu, J. (2018). Capsnet, cnn, fcn: Comparative performance evaluation for image classification. *International Journal of Machine Learning and Computing*, 9(6):840–848.

Jung, S., Park, J., and Lee, S. (2019). Polyphonic sound event detection using convolutional bidirectional lstm and synthetic data-based transfer learning. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 885–889, Brighton, UK.

Kao, C.-C., Sun, M., Wang, W., and Wang, C. (2020). A comparison of pooling methods on lstm models for rare acoustic event classification. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 316–320, Barcelona, Spain.

Karita, S., Chen, N., Hayashi, T., Hori, T., Inaguma, H., Jiang, Z., Someki, M., Soplin, N. E. Y., Yamamoto, R., Wang, X., Watanabe, S., Yoshimura, T., and Zhang, W. (2019). A comparative study on transformer vs rnn in speech applications. In *Proceedings of the IEEE Automatic Speech Recognition and Understanding Workshop*, pages 449–456, Singapore.

Kavalerov, I., Wisdom, S., Erdogan, H., Patton, B., Wilson, K., Roux, J. L., and Hershey, J. R. (2019). Universal source separation. In *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 175–179, New Paltz, NY, USA.

Ke, G., He, D., and Liu, T.-Y. (2021). Rethinking positional encoding in language pre-training. In *Proceedings of the International Conference on Learning Representations*, pages 1–14, Vienna, Austria.

Kim, B. and Pardo, B. (2019). Sound event detection using point-labeled data. In *Proceedings of the IEEE Workshop on Applications Signal Processing to Audio and Acoustics*, pages 1–5, New Platz, NY, USA.

Kim, N. K. and Kim, H. K. (2021). Polyphonic sound event detection based on residual convolutional recurrent neural network with semi-supervised loss function. *IEEE Access*, 9:7564–7575.

Kim, S., Park, S., Lim, S., and Kim, D. (2018). Classification performance analysis of weight update method applied to various convnet models. In *Proceedings of the International Conference on Control and Robots*, pages 78–83, Hong Kong.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*, pages 1–15, San Diego, USA.

Kong, Q., Cao, Y., Iqbal, T., Xu, Y., Wang, W., and Plumbley, M. D. (2019a). Cross-task learning for audio tagging, sound event detection and spatial localization: Dcase 2019 baseline systems. *arXiv preprint arXiv:1904.03476*, pages 1–5.

Kong, Q., Xu, Y., Sobieraj, I., Wang, W., and Plumbley, M. D. (2019b). Sound event detection and time–frequency segmentation from weakly labelled data. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27(4):777–787.

Kong, Q., Xu, Y., Wang, W., and Plumbley, M. D. (2020). Sound event detection of weakly labelled data with cnn-transformer and automatic threshold optimization. *IEEE/ACM Transactions on Audio, Speech and Language Processing*, 28:2450–2460.

Kothinti, S., Imoto, K., Chakrabarty, D., Sell, G., Watanabe, S., and Elhilali, M. (2019). Joint acoustic and class inference for weakly supervised sound event detection. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 36–40, Brighton, UK.

Laine, S. and Aila, T. (2017). Temporal ensembling for semi-supervised learning. In *Proceedings of the 5th International Conference Learning Representations*, pages 1–13, Toulon, France.

Lee, D., Lee, S., Han, Y., and Lee, K. (2017). Ensemble of convolutional neural networks for weakly-supervised sound event detection using multiple scale input. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2017 Workshop*, pages 1–6, Munich, Germany.

Lee, D. D. and Seung, H. S. (2000). Algorithms for non-negative matrix factorization. In *Proceedings of the 13th International Conference on Neural Information Processing Systems*, pages 535–541, Denver, CO, USA.

Lee, D. D. and Seung, S. (1999). Learning the parts of bbjects by non-negative matrix factorization. *Nature*, 401:788–791.

Li, B., Gulati, A., Yu, J., Sainath, T. N., Chiu, C.-C., Narayanan, A., Chang, S.-Y., Pang, R., He, Y., Qin, J., Han, W., Liang, Q., Zhang, Y., Strohman, T., and Wu, Y. (2021). A better and faster end to end model for streaming asr. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 5634–5638, Toronto, Canada.

Li, J., Zhang, M., Xu, K., Dickerson, J. P., and Ba, J. (2020). Noisy labels can induce good representations. *arXiv preprint arXiv:2012.12896*, pages 1–27.

Lin, L., Wang, X., Liu, H., and Qian, Y. (2019). Guided learning convolution system for dcase 2019 task 4. In *Proceedings of the Workshop on Detection and Classification of Acoustic Scenes and Events*, pages 134–138, New York, NY, USA.

Lin, L., Wang, X., Liu, H., and Qian, Y. (2020). Guided learning for weakly-labeled semi-supervised sound event detection. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 641–645, Barcelona, Spain.

Liu, L., Liu, X., Gao, J., Chen, W., and Han, J. (2020a). Understanding the difficulty of training transformers. *arXiv preprint arXiv:2004.08249*, pages 1–5.

Liu, Y., Chen, H., and Zhang, P. (2020b). Power pooling operators and confidence learning for semi-supervised sound event detection. *arXiv preprint arXiv:2005.11459*, pages 1–5.

Loshchilov, I. and Hutter, F. (2017). Sgdr: Stochastic gradient descent with warm restarts. In *Proceedings of the 5th International Conference of Learning Representations*, pages 1–16, Toulon, France.

Lu, J. (2018). Mean teacher convolution system for dcase 2018 task 4. Technical report.

Lu, Y., Li, Z., He, D., Sun, Z., Dong, B., Qin, T., Wang, L., and Liu, T. (2019). Understanding and improving transformer from a multi-particle dynamic system point of view. *arXiv preprint arXiv:1906.02762*, pages 1–15.

Ma, N., Zhang, X., Liu, M., and Sun, J. (2021). Activate or not: Learning customized activation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8032–8042, Vancouver, Canada.

Mangalam, K. and Prabhu, V. (2019). Do deep neural networks learn shallow learnable examples first? In *Proceedings of the Workshop on Identifying and Understanding Deep Learning Phenomena at 36th International Conference on Machine Learning*, pages 1–6, Long Beach, California, USA.

Martin, E. and Cundy, C. (2018). Parallelizing linear recurrent neural nets over sequence length. In *Proceedings of the 6th International Conference on Learning Representations*, pages 1–9, Vancouver, BC, Canada.

Mayorga, P., Ibarra, D., Zeljkovic, V., and Druzgalski, C. (2015). Quartiles and mel frequency cepstral coefficients vectors in hidden markov-gaussian mixture models classification of merged heart sounds and lung sounds signals. In *Proceedings of the International Conference on High Performance Computing and Simulation*, pages 298–304, Amsterdam, Netherlands.

McFee, B., Salamon, J., and Bello, J. P. (2018). Adaptive pooling operators for weakly labeled sound event detection. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(11):2180–2193.

Mehta, S., Ghazvininejad, M., Iyer, S., Zettlemoyer, L., and Hajishirzi, H. (2021). Delight: Deep and light-weight transformer. In *Proceedings of the International Conference on Learning Representations*, pages 1–19, Vienna, Austria.

Mesaros, A., Diment, A., Elizalde, B., Heittola, T., Vincent, E., Raj, B., and Virtanen, T. (2019). Sound event detection in the dcase 2017 challenge. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27(6):992–1006.

Mesaros, A., Heittola, T., Dikmen, O., and Virtanen, T. (2015). Sound event detection in real life recordings using coupled matrix factorization of spectral representations and class activity annotations. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 151–155, South Brisbane, QLD, Australia.

Mesaros, A., Heittola, T., Eronen, A., and Virtanen, T. (2010). Acoustic event detection in real life recordings. In *Proceedings of the 18th European Signal Processing Conference*, pages 1267–1271, Aalborg, Denmark.

Mesaros, A., Heittola, T., and Virtanen, T. (2016). Metric for polyphonic sound event detection. *Applied Sciences*, 6(6):1–17.

Michel, P., Levy, O., and Neubig, G. (2019). Are sixteen heads really better than one. In *Proceedings of the 33rd Conference on Neural Information Processing Systems*, pages 164–171, Vancouver, Canada.

Miech, A., Laptev, I., and Sivic, J. (2018). Learnable pooling with context gating for video classification. *arXiv preprint arXiv:1706.06905*, pages 1–8.

Misra, D. (2019). Mish: A self regularized non-monotonic activation function. In *Proceedings of the 31st British Machine Vision Conference*, pages 1–14, Shanghai, China.

Miyazaki, K., Komatsu, T., Hayashi, T., Watanabe, S., Toda, T., and Takeda, K. (2020). Conformer-based sound event detection with semi-supervised learning and data augmentation. In *Proceedings of the Workshop on Detection and Classification of Acoustic Scenes and Events*, pages 100–104, Tokyo, Japan.

Mnih, V., Larochelle, H., and Hinton, G. E. (2011). Conditional restricted boltzmann machines for structured output prediction. In *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence*, pages 514–522, Barcelona, Spain.

Mukhometzianov, R. and Carrillo, J. (2018). Capsnet comparative performance evaluation for image classification. *arXiv preprint arXiv:1805.11195*, pages 1–14.

Naranjo-Alcazar, J., Perez-Castanos, S., Zuccarello, P., and Cobos, M. (2020). Acoustic scene classification with squeeze-excitation residual networks. *IEEE Access*, 8:112287–112296.

Newell, A., Yang, K., and Deng, J. (2016). Stacked hourglass networks for human pose estimation. In *Proceedings of the European Conference on Computer Vision*, pages 483–499, Amsterdam, Netherlands.

Nguyen, C. and Tran, D. D. (2013). Sound classification for event detection: Application into medical telemonitoring. In *Proceedings of the International Conference on Computing, Management and Telecommunications*, pages 330–333, Ho Chi Minh City, Vietnam.

Ohishi, Y., Mochihashi, D., Matsui, T., Nakano, M., Kameoka, H., Izumitani, T., and Kashino, K. (2013). Bayesian semi-supervised audio event transcription based on markov indian buffet process. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 3163–3167, Vancouver, BC, Canada.

Okewu, E., Adewole, P., and Sennaike, O. (2019). Experimental comparison of stochastic optimizers in deep learning. In *Proceedings of the International Conference on Computational Science and Its Applications*, pages 704–715, Saint Petersburg, Russia.

Pan, S. J. and Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359.

Parascandolo, G., Huttunen, H., and Virtanen, T. (2016). Recurrent neural networks for polyphonic sound event detection in real life recordings. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6440–6444, Shanghai, China.

Patrick, M. K., Adekoya, A. F., Mighty, A. A., and Edward, B. Y. (2019). Capsule networks – a survey. *Journal of King Saud University - Computer and Information Sciences*, pages 1–16.

Pellegrini, T. and Cances, L. (2019). Cosine-similarity penalty to discriminate sound classes in weakly-supervised sound event detection. In *Proceedings of the International Joint Conference on Neural Networks*, pages 1–8, Budapest, Hungary.

Pellegrini, T. and Masquelier, T. (2021). Fast threshold optimization for multi-label audio tagging using surrogate gradient learning. *arXiv preprint arXiv:2103.00833*, pages 1–5.

Piczak, K. J. (2015). Environmental sound classification with convolutional neural networks. In *Proceedings of the IEEE 25th International Workshop on Machine Learning for Signal Processing*, pages 1–6, Boston, MA, USA.

Rabiner, L. R. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

Rajapakse, M. and Wyse, L. (2005). Generic audio classification using a hybrid model based on gmms and hmms. In *Proceedings of the 11th International Multimedia Modelling Conference*, pages 1–6, Melbourne, VIC, Australia.

Ramachandran, P., Zoph, B., and Le, Q. V. (2017). Searching for activation functions. *arXiv preprint arXiv:1710.05941*, pages 1–13.

Renals, S. and Swietojanski, P. (2014). Neural networks for distant speech recognition. In *Proceedings of the 4th Joint Workshop on Hands-free Speech Communication and Microphone Arrays*, pages 172–176, Villers-les-Nancy, France.

Sabour, S., Frosst, N., and Hinton, G. E. (2017). Dynamic routing between capsules. In *Proceedings of the 31st Conference on Neural Information Processing Systems*, pages 1–11, Long Beach, CA, USA.

Salamon, J. and Bello, J. P. (2017). Deep convolutional neural networks and data augmentation for environmental sound classification. *IEEE Signal Processing Letters*, 24(3):279–283.

Salamon, J., MacConnell, D., Cartwright, M., Li, P., and Bello, J. P. (2017). Scaper: A library for soundscape synthesis and augmentation. In *Proceedings of the Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 344–348, New Paltz, NY, USA.

Schroder, J., Moritz, N., Anemuller, J., Goetze, S., and Kollmeier, B. (2017). Classifier architectures for acoustic scenes and events: Implications for dnns, tdnns, and perceptual features from dcase 2016. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(6):1304–1314.

Serizel, R. and Turpault, N. (2019). Sound event detection from partially annotated data: Trends and challenges. In *Proceedings of the International Conference on Electrical, Electronic, and Computing Engineering*, pages 1–11, Srebrno Jezero, Serbia.

Shaw, P., Uszkoreit, J., and Vaswani, A. (2018). Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155*, pages 1–5.

Shi, L., Ahmad, I., He, Y., and Chang, K. (2018). Hidden markov model based drone sound recognition using mfcc technique in practical noisy environments. *Journal of Communications and Networks*, 20(5):509–518.

Shi, Z., Liu, L., Lin, H., Liu, R., and Shi, A. (2019). Hodgepodge: Sound event detection based on ensemble of semi-supervised learning methods. In *Proceedings of the Workshop on Detection and Classification of Acoustic Scenes and Events*, pages 224–228, New York, NY, USA.

Smaragdis, P. (2004). Non-negative matrix factor deconvolution; extracation of multiple sound sources from monophonic inputs. Technical report, Mitsubishi Electric Research Laboratories.

Smaragdis, P. (2007). Convolutive speech bases and their application to supervised speech separation. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(1):1–12.

Smaragdis, P. and Brown, J. C. (2003). Non-negative matrix factorization for polyphonic music transcription. In *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 177–180, New Pallz, NY, USA.

Smith, L. N. (2017). Cyclical learning rates for training neural networks. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision*, pages 464–472, Santa Rosa, CA, USA.

Snyder, D., Garcia-Romero, D., and Povey, D. (2015). Time delay deep neural network-based universal background models for speaker recognition. In *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 92–97, Scottsdale, AZ, USA.

Strubell, E., Ganesh, A., and McCallum, A. (2019). Energy and policy considerations for deep learning in nlp. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, Florence, Italy.

Stuttle, M. N. (2003). *A Gaussian Mixture Model Spectral Representation for Speech Recognition*. PhD thesis, Cambridge University, Cambridge, UK.

Swietojanski, P., Li, J., and Huang, J.-T. (2014). Investigation of maxout networks for speech recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 7649–7653, Florence, Italy.

Swietojanski, P. and Renals, S. (2014). Learning hidden unit contributions for unsupervised speaker adaptation of neural network acoustic models. In *Proceedings of the IEEE Spoken Language Technology Workshop*, pages 171–176, South Lake Tahoe, NV, USA.

Takahashi, N., Gygli, M., and Gool, L. V. (2018). Aenet: Learning deep audio features for video analysis. *IEEE Transactions on Multimedia*, 20(3):513–524.

Tan, Z., Sun, M., and Liu, Y. (2021). Dynamic multi-branch layers for on-device neural machine translation. *arXiv preprint arXiv:2105.06679*, pages 1–7.

Tarvainen, A. and Valpola, H. (2017). 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*, pages 1195–1204, Long Beach, California, USA.

Thewissen, J. G. M. H. and Nummela, S., editors (2008). *Sensory Evolution on the Threshold: Adaption in Secondarily Aquatic Vertebrates*. University of California Press.

Toth, L. (2015). Phone recognition with hierarchical convolutional deep maxout networks. *EURASIP Journal on Audio, Speech, and Music Processing*, 6(25):1–13.

Turpault, N. and Serizel, R. (2020). Training sound event detection on a heterogenous dataset. In *Proceedings of the Workshop on Detection and Classification of Acoustic Scenes and Events*, pages 200–204, Tokyo, Japan.

Turpault, N., Serizel, R., Shah, A., and Salamon, J. (2019). Sound event detection in domestic environments with weakly labeled data and soundscape synthesis. In *Proceedings of the Workshop on Detection and Classification of Acoustic Scenes and Events*, pages 253–257, New York, NY, USA.

Turpault, N., Serizel, R., and Vincent, E. (2020a). Limitations of weak labels for embedding and tagging. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 131–135, Barcelona, Spain.

Turpault, N., Serizel, R., Wisdom, S., Erdogan, H., Hershey, J., Fonseca, E., Seetharaman, P., and Salamon, J. (2021). Sound event detection and separation: A benchmark on desed synthetic soundscapes. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 840–844, Toronto, Canada.

Turpault, N., Wisdom, S., Erdogan, H., Hershey, J. R., Serizel, R., Fonseca, E., Seetharaman, P., and Salamon, J. (2020b). Improving sound event detection in domestic environments using sound separation. In *Proceedings of the Workshop on Detection and Classification of Acoustic Scenes and Events*, pages 205–209, Tokyo, Japan.

Tzinis, E., Wisdom, S., Hershey, J. R., Jansen, A., and Ellis, D. P. W. (2020). Improving universal sound separation using sound classification. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 96–100, Barcelona, Spain.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In *Proceedings of the 31st Conference on Neural Information Processing Systems*, pages 6000–6010, Long Beach, CA, USA.

Verma, V., Lamb, A., Kannala, J., Bengio, Y., and Lopez-Paz, D. (2019). Interpolation consistency training for semi-supervised learning. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 3635–3641, Macao, China.

Vesperini, F., Gabrielli, L., Principi, E., and Squartini, S. (2019). Polyphonic sound event detection by using capsule neural networks. *IEEE Journal of Selected Topics in Signal Processing*, 13(2):310–322.

Virtanen, T., Plumbley, M. D., and Ellis, D., editors (2017). *Computational Analysis of Sound Scenes and Events*. Springer.

Voita, E., Talbot, D., Moiseev, F., Sennrich, R., and Titov, I. (2019). Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808, Florence, Italy.

Wang, Y., Li, J., and Metze, F. (2019a). A comparison of five multiple instance learning pooling functions for sound event detection with weak labeling. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 31–35, Brighton, UK.

Wang, Y., Ma, X., Chen, Z., Luo, Y., Yi, J., and Bailey, J. (2019b). Symmetric cross entropy for robust learning with noisy labels. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 322–330, Seoul, South Korea.

Xia, W. and Koishida, K. (2019). Sound event detection in multichannel audio using convolutional time-frequency-channel squeeze and excitation. In *Proceedings of the Interspeech*, pages 3629–3633, Graz, Austria.

Xia, X., Togneri, R., Sohel, F., and Huang, D. (2018). Confidence based acoustic event detection. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 306–310, Calgary, AB, Canada.

Xia, X., Togneri, R., Sohel, F., and Huang, D. (2019). Auxiliary classifier generative adversarial network with soft labels in imbalanced acoustic event detection. *IEEE Transactions on Multimedia*, 21(6):1359–1371.

Xia, X., Togneri, R., Sohel, F., Zhao, Y., and Huang, D. D. (2020). Sound event detection using multiple optimized kernels. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:1745–1754.

Xu, T. B. and Liu, C. L. (2020). Deep neural network self-distillation exploiting data representation invariance. *IEEE Transactions on Neural Networks and Learning Systems*, Early Access:1–13.

Xu, Y., Kong, Q., Wang, W., and Plumbley, M. D. (2018). Large-scale weakly supervised audio classification using gated convolutional neural network. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 121–125, Calgary, AB, Canada.

Yang, L., Hao, J., Hou, Z., and Peng, W. (2020). Two-stage domain adaptation for sound event detection. In *Proceedings of the Workshop on Detection and Classification of Acoustic Scenes and Events*, pages 41–45, Tokyo, Japan.

Zeiler, M. D. (2012). Adadelta: An adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, pages 1–6.

Zeyer, A., Doetsch, P., Voigtlaender, P., Schlüter, R., and Ney, H. (2017). A comprehensive study of deep bidirectional lstm rnns for acoustic modeling in speech recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2462–2466, New Orleans, LA, USA.

Zhang, B., Xiong, D., and Su, J. (2018a). Accelerating neural transformer via an average attention network. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1789–1798, Melbourne, VIC, Australia.

Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. (2017). mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, pages 1–13.

Zhang, M. R., Lucas, J., Hinton, G., and Ba, J. (2019). Lookahead optimizer:k steps forward, 1 step back. In *Proceedings of the 33rd Conference on Neural Information Processing Systems*, pages 1–19, Vancouver, Canada.

Zhang, X.-L. and Wang, D. (2016). A deep ensemble learning method for monaural speech separation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(5):967–977.

Zhang, Y., Xiang, T., Hospedales, T. M., and Lu, H. (2018b). Deep mutual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4320–4328, Salt Lake City, Utah, United States.

Zhao, Z., Zhang, S. H., Xu, Z. Y., Bellisario, K., Dai, N. H., Omrani, H., and Pijanowski, B. C. (2017). Automated bird acoustic event detection and robust species classification. *Ecological Informatics*, 39:99–108.

Zhuang, J., Tang, T., Ding, Y., Tatikonda, S., Dvornek, N., Papademetris, X., and Duncan, J. S. (2020). Adabelief optimizer: Adapting stepsizes by the belief in observed gradients. In *Proceedings of the 34th Conference on Neural Information Processing Systems*, pages 1–30.

Zohrer, M. and Pernkopf, F. (2017). Virtual adversarial training and data augmentation for acoustic event detection with gated recurrent neural networks. In *Proceedings of the Interspeech*, pages 493–497, Stockholm, Sweden.