# Deep Learning for Acoustic Scene Classification

By

Xing Yong Kek

Submitted for the qualification for the

Degree of Doctor of Philosophy in

Science, Agriculture and Engineering

Faculty of Science, Agriculture & Engineering

Newcastle University

2022

# Abstract

Acoustic scene classification is the ability to automatically detect the vicinity based on the sound produced by the environment. The current framework uses a log Mel spectrogram with a convolution neural network. There is an advancement toward low complexity modeling where the convolution neural network is compressed. Log Mel-spectrogram suffers from Heisenberg Uncertainty Principle, where the representation is based on a fixed timescale and becomes unstable to time-wrapping deformation when the timescale is greater than 25ms. Hence, instead of using log Mel-spectrogram, this thesis investigates the viability of using wavelet scattering. Wavelet scattering is the cascade of wavelet transform and provides a scaling factor based on wavelet theory. It is stable to deformation when the timescale is greater than 25ms. However, the averaging operation in the wavelet scattering has a limiting effect on the maximum scaling factor of the wavelet. Hence, wavelet scattering also suffers from Heisenberg Uncertainty Principle. In addition, it is observed that the first and second-order coefficients have a considerable difference in magnitude, making the acoustic classification model difficult. The huge disparity in data representation can cause an internal covariate shift, leading to a slower convergence rate and even poor classification accuracy.

Hence, this thesis proposed a multi-timescale using genetic algorithm for feature selection on the limitation of 'fixed' timescale and a two-stage convolution neural network architecture framework to tackle the magnitude disparity between first and second-order coefficients in wavelet scattering. Despite the challenge of designing low complexity models, this thesis adapted the model compression technique to compress the convolution block of the two-stage convolution neural network architecture. In the feature representation, using wavelet scattering and multi-timescale are not a viable option as having multiple timescales will increase the model size and the time complexity. Hence, a simple 'mixing' of first and second-order to increase the variety of timescale slightly is being proposed. In addition, the genetic algorithm for feature selection is adapted to reduce the size of the second-order frequency dimension due to a large frequency dimension (e.g., > 500) as compared to first-order (e.g., <100) and fine-scale resolutions which can result in unnecessary over-representation of acoustic profiles.

## Acknowledgments

First and foremost, I would like to express my sincere gratitude to Dr. Cheng Siong Chin and Dr. Ye Li, my mentors, for their guidance, patience, and encouragement throughout my entire research journey. They provided me with valuable insights and constructive feedback that steered me to the right direction. Furthermore, they provide huge amount of support and strongly belief that I can complete my research journey.

I would also like to take this opportunity to express my gratitude to Xylem Ltd and Economic Development Board (EDB) for giving me this opportunity to enrol into this special arrangement of working on my research together with the company.

In Xylem Ltd, I want to thank my colleagues who provided me with knowledge and a way to destress through coffee talks: Dr. Ehsan Shafiee, Felipe F Corral Jr, Joanne Caisip, and Van Koh.

I want to express my gratitude to my "brother-in-arms", Dr. Teck Kai Chan, who have journeyed with me. Throughout this incredible journey he provided me with great morale support. He also shares his wisdom with me when I have doubt.

I want to thank my friends for their unfailing support.

Most importantly, I want to express my heartfelt gratitude to my wife, Jessica Chia, and my family members: my parents, Moon Chew Kek and Jenny Ng, siblings: Xing Yi Kek, Xing Yao Kek and his families, his wife and my two nieces, for their love and support. Without them I would not have come this far. They are my mental support when going through difficult times during this journey.

# Table of Contents

# LIST OF TABLES

# LIST OF FIGURES

# Chapter 1: Introduction

## 1.1. Acoustic Scene Classification

Acoustic Scene Classification (ASC) or Environmental Sound Classification (ESC), is the task of recognizing the vicinity you are in, based on the sound produced by the scene. The scene, in this case, is a pre-defined semantic description of a location like “beach” or “airport”. Hence, the audio recordings in these locations are called acoustic scenes (Mesaros et al., 2018, Abeßer, 2020). ASC originated from the topic of auditory scene analysis (ASA) (Bregman and McAdams, 1994, Sawhney and Maes, 1997), which later branches into Computational Auditory Scene Analysis (CASA) (DeLiang and Guy, 2006). The specification of having the word computational is intended for the entire classification process to be handled or automated by an artificial intelligent (A.I.).

However, it is challenging, as a sound recording from the scene is a collective of sound sources with varying acoustic characteristics (Uzkent et al., 2012). They further categorize the characteristics of environmental sounds into three groups. Stationary sounds: are sounds that produce constant, with little to no changes to the frequencies over time (e.g., a train moving at constant speed); Quasi-stationary sounds: are sounds that are produced from the combination of multiple stationary sounds where the frequencies pattern can be observed over time (e.g., sea wave, orchestral); Non-stationary sounds: are sounds that have erratic frequencies over time (e.g. conversation between a group of people). These sound sources are usually interleaving and overlapping in a single audio recording, proving it difficult to identify the number of sound sources, and even if we can separate the sound source, recognizing the quasi-stationary and non-stationary sounds is also challenging due to their changing nature. Adding on to the challenges, some sound sources might be an event present only in a certain period. The random occurrence of such events might confuse the classifier. In addition, different scenes can share similar sound events or profiles, making it extremely difficult to differentiate. Hence, right from

the beginning (Sawhney and Maes, 1997, Clarkson et al., 1998), proposed a general framework on how to tackle ASC or ESC. The framework can be simplified into two steps as illustrated in Figure 1.1: 1) Pre-processing of an audio signal or feature extraction (assuming that the raw signal is digitized); 2) These features extracted from the pre-processing steps will act as an input to a classifier to determine the class it belongs. This framework is also called machine learning (ML) approach as the classifier employed are usually machine learning algorithm and even till date, this paradigm is still producing state-of-the-art algorithms for ASC and is being used by many researchers as observed in various review papers (Heittola et al., 2020, Gharib et al., 2018, Mesaros et al., 2018, Chandrakala and Jayalakshmi, 2019, Abeßer, 2020) and research platforms such as Detection and Classification for Audio Scene and Events (DCASE). (DCASE).

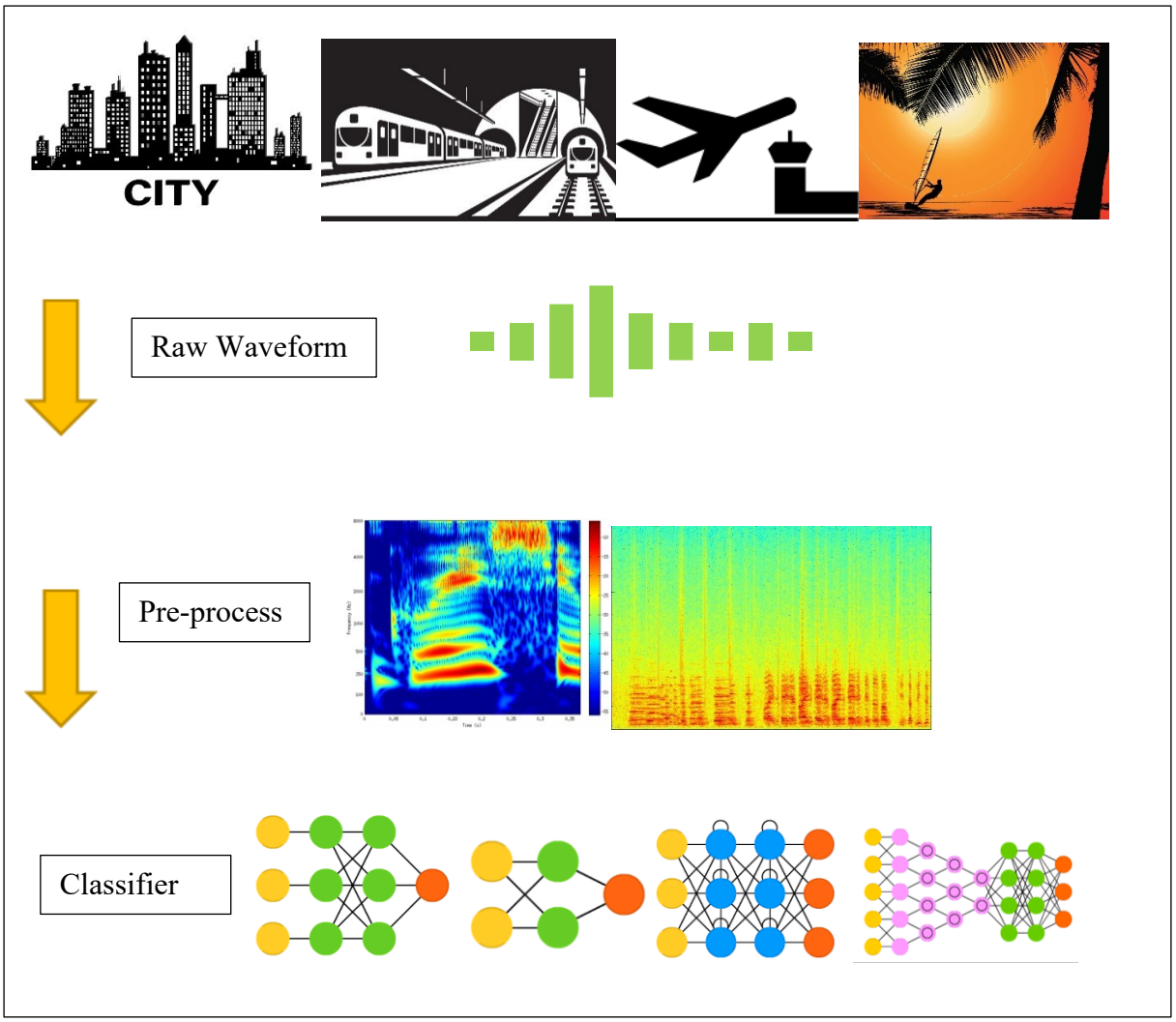



**Figure 1.1.** ASC framework from raw waveform.

While the framework remains intact, the classifier algorithm has evolved tremendously into newer algorithms such as deep learning (DL). One DL technique heavily utilized for ASC is

the convolution neural network (CNN). The CNN is a family of artificial neural networks (ANN) that learn through updating the weights through backpropagation (Rumelhart et al., 1986). Unlike the other ANN, CNN was initially exclusively built for image classification. The input representation is usually a picture source or a 2-Dimensional model with multiple channels reflecting the RGB of an image. Hence, CNN is generally fed with a 3-Dimensional representation of *(H x W x C)*, where *H* denotes the height of the image, *W* denotes the width of the image, and *C* denotes the RGB channels of the image. CNN has grown tremendously due to the extensive research in image classification and the advancement of hardware technologies such as the Graphical Processing Unit (GPU) which enable the acceleration of this research.

While other deep learning techniques are used for ASC, CNN is still the most popular choice (Chandrakala and Jayalakshmi, 2019, Gharib et al., 2018, Mesaros et al., 2018, Abeßer, 2020). As aforementioned, this can be attributed to the extensive research on CNN in either image domain (Alzubaidi et al., 2021, Khan et al., 2020) or ASC, and what contributes to the extensive research on CNN is the advancement of hardware and the way of computing DL. Notably, the ability to process CNN algorithm using GPU has sped up the progress of CNN, coupled with open-source deep learning frameworks such as TensorFlow (Abadi et al., 2016), and PyTorch (Paszke et al., 2019) further accelerate this process. However, one caveat of CNN or DL is the need for adequate training data; they tend to overfit and generalize poorly (Alzubaidi et al., 2021).

Hence, another essential element is the availability of large open datasets such as ImageNet (Deng et al., 2009) for image classification, COCO (Lin et al., 2014) for object detection, DCASE (DCASE) for a variety of acoustic challenges. AudioSet (Gemmeke et al., 2017) is another extensive acoustic dataset. It has provided a platform for researchers to benchmark their models while relieving the pain of self-curating datasets.

In addition, with the effort to advance DL, by coupling open-source deep learning framework and open datasets, the research communities and framework's owners have released tutorials and pre-trained DL models (Han et al., 2021), leading to an easy uptake for both the industry and the academia using CNN. Pre-trained CNN are models that have a fixed CNN architecture based on some of the state-of-the-art (SOTA) presented in Chapter 3 and undergone the training phase, usually over a considerable amount of data; hence, the models have already captured a wealth of knowledge. Pre-trained models can be used directly if the tasks are similar to the image domain. Therefore, one downside of transfer learning is it can be quite a domain specified, and the transfer learning from image classification to audio classification is hardly implemented as exhibited in ESC-50, US8K, and DCASE. However, there is an immergence towards transfer

learning with pre-trained learning using the large-scale audio dataset, such as Audioset in the ESC-50 (Piczak, 2015) and US8K (Salamon et al., 2014).

In DCASE 2016 ASC challenge (Valenti et al., 2017, DCASE), where several challengers adapted to the current ASC framework using CNN as shown in Figure 1.2. It achieved a considerable performance advantage against classical approaches (Barchiesi et al., 2015, Abeßer, 2020, Chandrakala and Jayalakshmi, 2019) such as hidden makrov machine (Clarkson et al., 1998), artificial neural network, support vector machine (Jia-Ching et al., 2006) and K-nearest neighbour (Piczak, 2015).

As aforementioned, CNN is designed specifically to tackle image classification and for it to be applicable to sound classification such as ASC, we need to pre-process the raw waveform into a 2-D representation. In this case, a time-frequency representation such as log mel-spectrogram is an ideal candidate to be the input representation. In fact, a time-frequency representation is an image with a single channel, hence, the adaptation from image domain is rather straightforward, thus, empowering the popularity of time-frequency representation. In summary, the current trend of ASC is the conversion of the raw signal into log mel-spectrogram, then feeding it to CNN, as depicted in Figure 1.2. (Refer to Chapter 3 for description of the CNN topology)

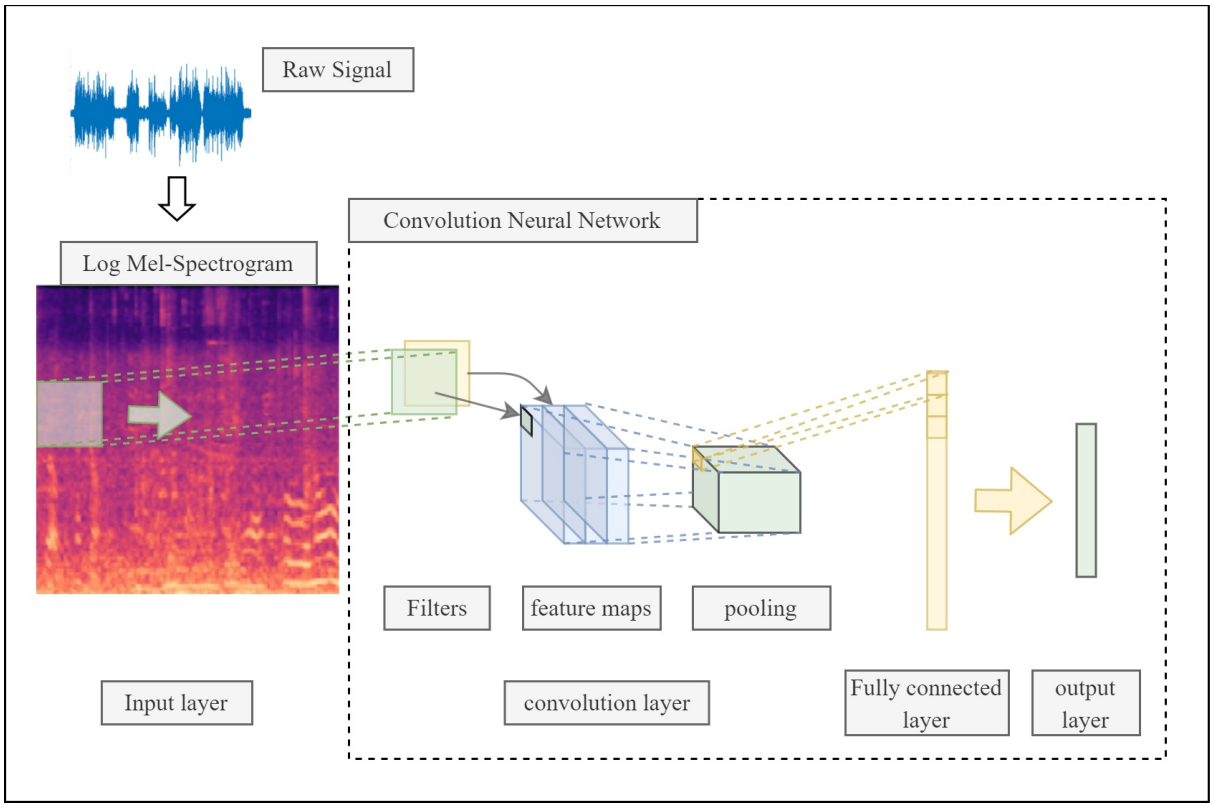



**Figure 1.2.** Current SOTA framework for ASC**.**

### 1.2. Limitation of Wavelet Scattering a Time-frequency representation

Considering this development, this thesis looks at other possible time-frequency representations that can serve as an alternative to log mel-spectrogram. Log mel-spectrogram is locally translation invariant and stable to deformation when the timescale or window size is below 25ms (Andén and Mallat, 2014). Translation invariant means that a slight shift in time should not result in a new label. For example, the opening and closing the bus's door can happen anytime in a 10s recording. Hence, regardless of when the event is captured in the 10s, the desired representation should be able to relate this event to a scene recorded in a bus. Stability to deformation in this context means that the translation from the audio signal to the feature representation should satisfy Lipschitz continuous condition. Hence, a slight deformation in the raw signal should reflect a small change to the output representation based on the Euclidean norm.

These two properties are the reason why the log mel-spectrogram is so successful. However, log mel-spectrogram suffers from deformation instability when the timescale is larger than 25ms and the effect of Heisenberg Uncertainty Principle due to having a fixed window size or timescale. Hence, (Andén and Mallat, 2014, Mallat, 2012, Bruna and Mallat, 2013) developed wavelet scattering (WS) to combat these two weaknesses. In simplicity, wavelet scattering is constructed by cascading wavelet transform of a given signal (Andén and Mallat, 2014, Mallat, 2012, Bruna and Mallat, 2013) and has the property of both local translations invariant and stable to time-wrap deformation even when timescale is larger than 25ms.

However, WS also has a few limitations which this thesis intends to resolve:

- A scattering transform usually produces two useful sets of coefficients (Andén and Mallat, 2014, Mallat, 2012, Bruna and Mallat, 2013), namely the first and second-order coefficients. As the cascading term suggests, the second-order is derived from the first order. Hence, due to the multiplicative operations of first-order coefficients with wavelet transformation, second-order coefficients have a sparse representation, and their magnitude is extremely small compared to first-order ones. This thesis termed this huge magnitude disparity between first and second-order as a 'magnitude disparity problem.' Hence, there is a need to give consideration when using WS. Notably, second-order is needed to capture high-frequency resolution.
    - A two-stage CNN architecture framework and 'batch normalization' is proposed to tackle the magnitude disparity problem.

- While wavelet scattering comprises wavelet transform with multiple wavelet scales, the averaging operation to make the wavelet scattering invariant to translation limit the maximum timescale. Hence, wavelet scattering is affected by Heisenberg Uncertainty Principle, and there is a need to increase the timescale resolution. With the addition of CNN as the classifier, one can quickly build an ensemble model with a different timescale of WS and perform late fusion methods (Goodfellow et al., 2016), which combines the classification scores of all the models via averaging. Even though this is a straightforward method, it requires high computational cost during deployment as the server or devices need to perform inference on multiple models.
    - The solution is to leverage on two-stage CNN approach to adapt an early fusion technique where WS with different timescales is combined during the feature level extraction phase of the CNN. By combining multiple timescales, the possibility of redundancy and irrelevancy is very high. Thus, genetic algorithm for feature selection (GAFS) is used to get the optimal subset of the combined multi-timescales of WS. The choice of genetic algorithm is related to the initial problem of magnitude disparity. Fortunately, the by-product of using a reduced feature size directly impacts the model's time complexity.
- In addition to the need for multi-timescales WS, the second-order has a large frequency dimension (e.g., > 500 rows). The large frequency is due to the addition of fine-scale resolution to capture high-frequency acoustic profiles. Hence, similar to the multi-timescales problem, there lies a possibility of redundancy and irrelevancy even in WS itself, which might impede the performance of the CNN.
    - Hence, the solution is similar to 'multi-timescaling', by applying genetic algorithm for feature selection (GAFS) onto WS.

### 1.3. Towards low-complexity Modelling

In recent event in DCASE challenge (Heittola et al., 2020, DCASE), there is a focus on developing low complexity model that can be deployed into IoT or consumer portable devices such as wireless earpieces or smartphones. The use case of the low computational complexity model is in line with the advancement and mass adaptation of the Internet of Things and robotics, such as in hearing-aid (Tchorz et al., 2017, Hüwel et al., 2020), autonomous surveillance using environmental audio scene and sound event recognition (Chandrakala and Jayalakshmi, 2019), and even navigation system for autonomous vehicles (Barchiesi et al., 2015, Gharib et al., 2018). In addition, if deployed in a video recording device, there can improve general scene classification with aided audio-visual classification (Heittola et al., 2020, DCASE, López-

Cifuentes et al., 2020). For these use cases to work efficiently, on-node detection is paramount to making near real-time decisions.

However, one of the challenges in deploying into mobile devices or small IoT or consumer wearables devices is the limitation of hardware capabilities that the devices can afford. Therefore, low latency and less memory models are desired. Due to the size of these devices, they usually do not have the computational power similar to that of a high-performance computational machine (HPC) or even a 'consumer-level' high-end computer. Hence, for such applications, the CNN model is trained in the HPC or high-end computers and later deployed into the small devices for inference purposes. This makes sense as there is only a need to compute the feed-forward portion. However, even as an inference model, it might still be too computationally intensive for these small devices. In addition, most of these devices run on standalone power sources, and energy conservation is paramount for the devices. Hence, there is a need and use case to compress the model or inference model into a 'light-weight' model, known as 'model compression' (Cheng et al., 2017).

While model compression is one way to reduce model complexity, another option is to reduce the dimensionality of the input representation. The larger the input, the more matrix-multiplication and additive operations are required to cover the entire input and a reduction in the input representation can help save time complexity.

In summary, the proposed approach reduces both time and size complexity of the model in three folds:

- With the adaptation of genetic algorithm for feature selection, the high-frequency resolution of WS can be reduced. (See Section 4.2). By lowering the input representation dimension, it has a direct impact to time complexity of the model. It is noted on the heavy computational cost of using GA, similar to how 'pretraining' the model on a very large audio dataset such as Audioset, has a heavy computational overhead prior to the actual training of the model. While both techniques improve accuracy of the model, the outcome is different. The outcome of GA is an 'optimal' feature subset and potentially lowering of the time complexity, while 'pretraining' provides a model with 'trained' weights that allows the model to start learning with a boarder knowledge. As it requires tremendous resource and time to analysis the heavy computational cost of possible pre-computational overhead described above, in this thesis, time complexity is only calculated on the model training and the pre-computational overhead is excluded.

- However, for the application of a low complexity model, 'multi-time scaling is not a great option as the additional resolution will increase model size complexity. Hence, this thesis investigated a simple mixing of different timescales of the first and a single second-order to enrich the multi-timescale capability of WS. By coupling just, the first and second-order together, the number of input representation in the first stage remain similar to how WS is being handled, while providing a soothing relief on the timescale limitation. In this thesis, it is termed as 'Interleaved' wavelet scattering (IWS).
- While the first two folds help reduce the time complexity of the model, the final fold is to reduce the CNN model size complexity. By using model compression techniques, being convolution factorization and quantization as discussed in Chapter 4, this thesis developed a small-sized CNN model that can be deployed into low computational powered devices.

On a site note, with the advancement of network infrastructure to 5G, one can tap into cloud computing to perform training, reinforced learning, and even inference. The 5G enables the high-speed transfer of large data files, which can offload heavy computational power and offer energy conservation. It provides more options and flexibility on deploying CNN or DL-capable devices. Unfortunately, this thesis will not dwell further into the topic of CNN deployment as it will be another domain in internet of things (IoT).

### 1.4. Datasets

As briefly mentioned in Section 1.1, one can tap onto open datasets to evaluate and benchmark the proposed model. As such, this thesis trained and experimented the proposed models and approaches mainly on DCASE 2020/2021 dataset (Heittola et al., 2020). In addition, we also evaluated our models on two other popular ASC datasets, being Environmental Sound Classification (ESC-50) (Piczak, 2015) and Urban Sound Dataset (US8K) (Salamon et al., 2014). The details of the datasets are presented in Table 1.1 and The column "Fold" describes the number of predefined cross-validation folds provided by the authors who curated the dataset. As DCASE 2021 has only provided a single training/testing setup, instead of cross-validation, the setup is being iterated over five times and the classification result is the average of the five tests.

#### ESC-50

ESC-50 curated by (Piczak, 2015) consists of 2000 environmental audio recordings, each with a 5s duration. The recordings are organized into 50 classes and are equally balanced. While the dataset is categorized as ASC, the recordings consist of classes of sound events and are more

DETAILS OF THE DATASETS

| No | Dataset | No of Data | #Classes | Duration (s) | Fold(s) |
|---|---|---|---|---|---|
| *1* | ESC-50 | 2000 | 50 | 5 | 5 |
| *2* | US8K | 8732 | 10 | <=4 | 10 |
| *3* | DCASE 2021 | 23040 | 10 | 10 | 1 |

**Table 1.1.** Details of the datasets. This Table presents the details of the datasets used to evaluate our proposed models.

related to Sound Event Recognition (SER) for this case. Hence, this dataset provides a variety of SER and ASC recordings which can be broadly categorized into animal sounds, natural soundscapes, human non-speech sounds, interior/domestic sounds, and exterior/urban noises.

**US8K**

As for US8K, it consists of 8732 audio recordings, and each recording has a duration of less or equal to 4s. The recordings are organized into 10 classes: “air conditioner”, “car horn”, “children playing”, “dog barking”, “drilling”, “engine idling”, “gunshot”, “jackhammer”, “siren”, “street music”. Based on the classes, US8K is deemed as a pure SER task as the recordings are mainly events rather than the scene.

**DCASE 2021**

While US8K is mainly Sound Event Recognition (SER), DCASE 2021 is an example of pure ASC recordings, recorded from 10 European cities in 10 different acoustic scenes. Each recording has a duration of 10s and are given a label based on the following classes: ‘airport’, ‘shopping mall’, ‘metro station’, ‘street pedestrian’, ‘public square’, ‘street traffic’, ‘tram’, ‘bus’, ‘metro’, ‘park’. In a real-world scenario, most of the time, the audio devices are recorded in various devices such as smartphones, consumer video cameras, surveillance cameras, and IoTs, and are usually recorded in different audio resolutions. Hence, to increase the robustness of the DCASE 2021, (Heittola et al., 2020) included recordings from 9 devices. The breakdown of the devices is as follow and given names of (A, B, C, S1-S6): Device A forms the main bulk of the dataset and is recorded using either Soundman OKM II Klassik/studio A3 or Zoom F8 audio recorder using 48kHz sampling rate and 24-bit resolution. Device B is recorded by a smartphone, Samsung Galaxy S7. Device C also a smartphone, iPhone SE. Lastly, S1 to S6 are synthetic data.

In summary, this thesis evaluated on both ASC and SER in relation to the urban sound dataset. One of the reasons why these set of datasets are being selected is that it provides a well-rounded

mix of ASC and SER. In addition, it also provides insight on how the model reacts to different dataset sizes, 2000, 8732, and 23040. Though the largest draw in selecting these datasets is that it is heavily used by researchers to benchmark their ASC models (Heittola et al., 2020, Gao and McDonnell, 2020, McDonnell and Gao, 2020, Gharib et al., 2018, Mesaros et al., 2018, Barchiesi et al., 2015, Guzhov et al., 2022, Koutini et al., 2019, Ren et al., 2018, Zhang et al., 2019, Suh et al., 2020, DCASE, Mohaimenuzzaman et al., 2021, Abeßer, 2020), providing a solid platform to gauge the competitiveness of the proposed models. Hence, these provide a well-rounded empirical study on the proposed methods described in Chapter 4.

### 1.5. Contributions and Structure of the Dissertation

Lastly, in this section, a summary of the contribution of this thesis is being listed out, follow by a description of how the thesis is being organized. In addition, the chapters in the thesis comprise published works from the following publications:

1. X. Y. Kek, C. S. Chin and Y. Li, "Acoustic Scene Classification Using Bilinear Pooling on Time-liked and Frequency-liked Convolution Neural Network." *in IEEE Symposium Series on Computational Intelligence (SSCI 2019)*, Xiamen, China, 2019.
2. X. Y. Kek, C. S. Chin, and Y. Li, "An Investigation on Multiscale Normalised Deep Scattering Spectrum with Deep Residual Network for Acoustic Scene Classification," *22nd IEEE/ACIS International Fall Virtual Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPDI),* pp. 29-36, 2021.
3. X. Y. Kek, C. S. Chin and Y. Li, "Multi-Timescale Wavelet Scattering with Genetic Algorithm Feature Selection for Acoustic Scene Classification," in *IEEE Access*, vol. 10, pp. 25987-26001, 2022.
4. X. Y. Kek, C. S. Chin, Y. Li, "An Intelligent Low-complexity Computing Interleaving Wavelet Scattering based Mobile Shuffling Network for Acoustic Scene Classification" in in *IEEE Access*, vol. 10, pp. 82185-82201, 2022.

In summary, the main contributions in the thesis are as follows.

- In light of the magnitude disparity problem, a two-stage CNN architecture framework is being proposed such that the initial convolution layers explicitly extract features from first and second-order coefficients from WS.
- Typically, 'renormalization' technique described in Section 2.2.3 is used to mitigate the magnitude disparity problem between first and second-order coefficients in WS. This thesis proposed to tap into the normalization technique that is integrated into the

convolution neural network (such as batch normalization) to relieve the problem. The outcome has shown that any type of normalization techniques helps with relieving the problem. In addition, the proposal of using Batch Normalization turns out to perform better than renormalization technique.

- WS is identified to still be affected by Heisenberg Uncertainty Principle due to the averaging operation performed on the first and second-order. Hence, a multi-timescale WS is developed to improve the timescale resolution.
- Prior to constructing multi-timescale WS, an investigation is being put up to find an optimal range for the timescale as it is prudent to haphazardly combine WS of various timescales. As a result, the proposed multi-timescale WS is constructed with first-order with a timescale of 46ms, and second-order(s) with timescale of 92ms, 185ms, and 371ms.
- A comparative analysis of log Mel-spectrogram and wavelet scattering with different timescales is being studied. This study shows that log Mel-spectrogram does not perform well when timescale is greater than 46ms. In addition, the optimal timescale for wavelet scattering for ASC is 92ms.
- In regard to multi-timescale WS, two challenges have been identified when combining various WS with different timescales. The first is that the second-order coefficient has a large frequency dimension. The second is that the first and second-order coefficients are governed by an energy dispersion theorem. This theorem states that as the timescale get larger, more energy is dispersed into the higher-order, in this case, more energy is dispersed from the first-order to the second-order.
- Hence, this thesis tackles the problems in two parts. The first part involves analyzing the optimal timescale range and the impact of energy dispersion for each timescale to orchestral the proposed multi-timescale WS. The second part uses the genetic algorithm for feature selection to remove redundancy and to reduce the size of second-order coefficients.
- Building towards low complexity modelling, the current two-stage CNN convolution layers are being redesigned into a ‘light-weights’ version adapted from mobile network described in Chapter 4. In addition, this paper proposed the incorporation of shuffling modules into the new two-stage CNN architecture.
- The proposed shuffling modules differ from existing works where shuffling is performed either channel-wise or spatial-wise. The primary purpose is to create a more dedicated time-frequency representation. Hence, the shuffling is performed on the

feature map ‘frequency-wise’ and ‘temporal-wise’. Frequency-wise shuffling shuffles the feature map along the frequency axis with concept of binning. Each bin is being shuffled such that the high-frequency spectrum is shuffled to low-frequency spectrum position allowing the model to learn the general acoustic profile of a scene rather than memorizing what is happening at the low-frequency or high-frequency spectrum, which is erratic for ASC. As for temporal-wise shuffling, the shuffling is applied along the time axis of the feature map.

- As for the input features, a simple ‘mixed’ wavelet scattering termed ‘Interleaved’ wavelet scattering (IWS) is being introduced.
- the low complexity model is being evaluated on three different ASC datasets and has a rather competitive results, scoring 82.15% for ESC-50, 81% for US8K, and 70.4% for DCASE 2021. In addition, the inclusion of the shuffling modules has shown to improve the model ability to generalize based on the observation of improved logloss.
- Various combination of IWS is being evaluated to understand the impact of the energy dispersion theorem. The investigation affirmed that the larger the timescale, the more information is being transferred to the higher order. In this case, energy is transferred from first-order to the second-order. Taking note of this finding, a rule is set when configuring IWS such that the timescale of the first-order should always be smaller than the timescale of the second-order.

The thesis is organized into the following Chapters: Chapter 2 discussed on wavelet scattering and the construction of IWS. In chapter 3, the CNN architecture is first being described in detail and the discussion leads to the development of the two-stage CNN architecture framework. As there is a trend toward low complexity computing, the topic is shifted towards dimensionality reduction using genetic algorithm for feature selection on the carefully constructed multi-timescale WS, which is described in chapter 4. While also in chapter 4, the new ‘light-weights’ two-stage CNN model, termed two-stage mobile shuffling network is being discussed. The proposed models and approaches are being experimented on the ASC datasets presented in Section 1.4., the results and findings from the experiment is being evaluated and discussed in chapter 5. Lastly, in chapter 6, this thesis is being concluded with possible future directions.

# Chapter 2: Wavelet Scattering

## 2.1. Introduction

Time-frequency representation is a signal processing technique that retains both time and frequency resolution. The raw signal waveform by itself is a time representation and the maximum time resolution you can get. The representation is a 1-Dimensional vector of time-amplitude of the signal. Plotting it usually looks like a sinusoid (sine wave). Unfortunately, the raw waveform has very high resolution and tends to be noisy (Barchiesi et al., 2015, Chandrakala and Jayalakshmi, 2019). Especially for acoustic scenes where multiple sound sources interleaved and overlapped, constructing a new single time-amplitude representation. However, more recent research is trending toward an End-to-End system, where 1-Dimensional CNN is employed to learn the modulation of the raw waveform instead of using a predefined signal processing technique to extract the features (Aytar et al., 2016).

Understanding the time representation is complex and somewhat useful in signal processing related applications. The signal is usually pre-processed into the frequency domain (Jaitly and Hinton, 2011, Barchiesi et al., 2015). One of the most popular and common approaches is the Fourier Transform. The Cooley-Tukey algorithm (Cooley and Tukey, 1965) is the most widely used due to the efficiency of computing the frequency of a signal. The algorithm is typically known as fast Fourier transform (FFT). This algorithm decomposes time resolution to frequency components (spectral components) based on Fourier's theorem that every signal can be decomposed into sine and cosine waves. The frequency spectrum shows what frequencies exist in the signal and is measured in cycles/second, commonly known as 'Hertz'.

However, by just analyzing the signal in the frequency domain, we fully lose the temporal information. Temporal information is essential to capture the occurrence of transient or sound events. As aforementioned, it consists of quasi-stationary and non-stationary acoustic profile. In addition, with just frequency representation, we assume that the signals in a scene are stationary, which is not the case. Hence, there is a need to include both temporal and spectral

information to distinguish ASC better and combine them to provide the time-frequency representation.

As per Section 1.1, the most popular time-frequency representation used in ASC is log mel-spectrogram. The log mel-spectrogram is constructed by applying mel-scale filters onto a spectrogram, and then rescale it with a logarithm. The spectrogram is built from "Short-Time Fourier Transform" (STFT) (Mallat, 2009, Polikar, 2006), which means converting time domain to the frequency domain with Discrete-Time Fourier Transform (DFT) with a given time window, instead of the entire signal. Hence, STFT can be seen as an action to segment a signal into a sequence of DFT of a windowed signal.

Next, applying mel-scale filterbanks: for two reasons, one is to reduce the high resolution of STFT, next is to provide a certain degree of stability to deformation (Andén and Mallat, 2014). The Mel-scale is a transformation that converts signal frequency to how a human perceives various frequency ranges. Base on the Mel-scale equation below:

$$m = 1127.\log\left(1 + \frac{f}{700}\right) \quad (1)$$

the mel-scale rescales the frequency components in a logarithmic fashion; hence, low frequency bandwidths have better resolution and high frequency bandwidths have lower resolution, as depicted in Figure 2.1.

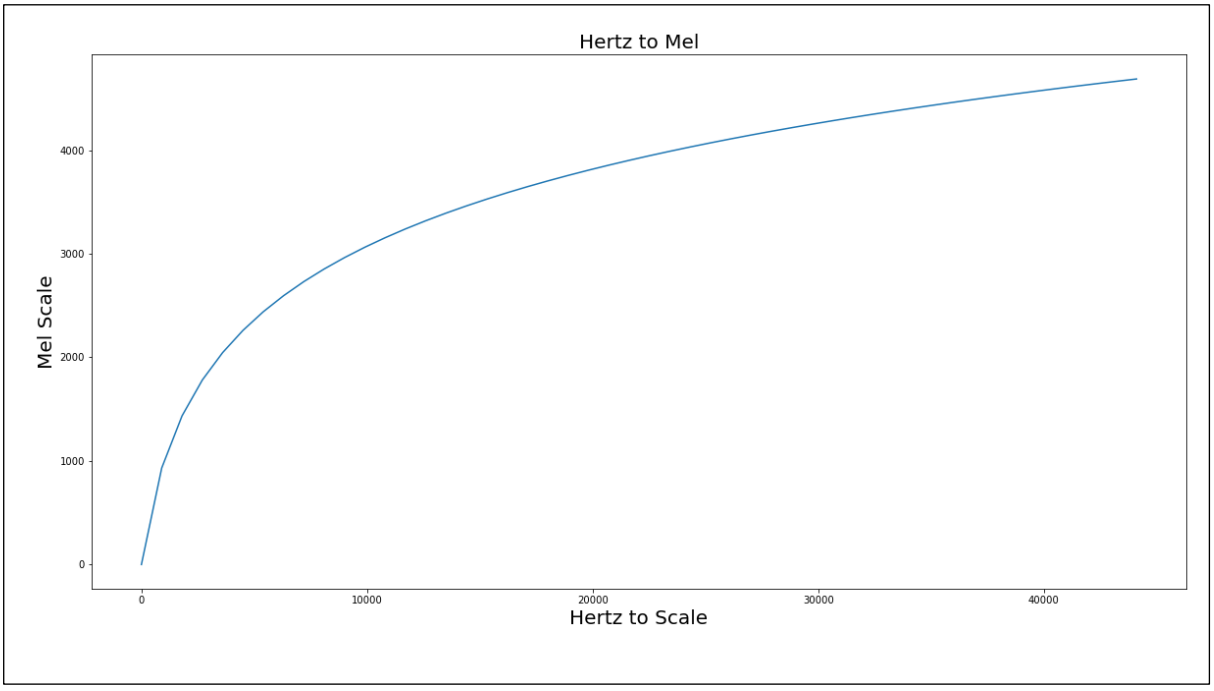



**Figure 2.1.** Graph of mel-scale algorithm based on 44.1kHz

The most common way to convert a spectrogram into a mel-scale is by applying mel-scale filterbanks. The filterbanks can be described as a set of bandpass filters and each filterbank consists of a range of frequency (e.g., between 100Hz to 250Hz). Hence, a mel-scale filterbank divides the signal into multiple bins based on the mel-scale as illustrated in Figure 2.2.

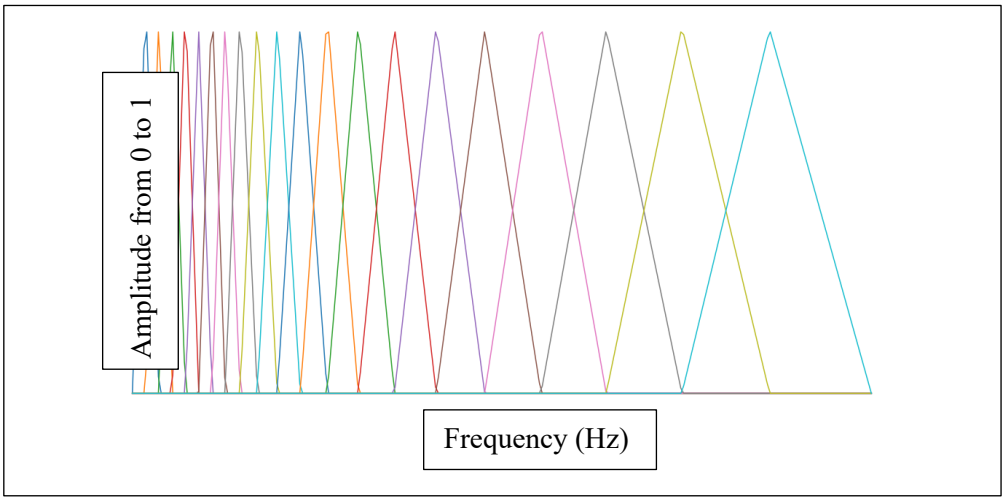



**Figure 2.2.** Example of 20 triangular mel-scale filterbanks using Librosa (McFee et al., 2015). The x-axis is the frequency in Hz, while the y-axis is the amplitude scaled between 0 and 1.

Previously, the more common approach was to adapt signal processing techniques from the speech domain as it is one of the most popular topics and the more matured acoustic domain. Researchers from the speech domain will continue to apply a Discrete Cosine Transform (DCT) onto log mel-spectrogram, producing Mel-frequency cepstral coefficients (MFCC).

Initially, MFCC has been used as the signal processing technique for ASC tasks (Barchiesi et al., 2015), but the growth of CNN has pushed the popularity back to log mel-spectrogram. This can be attributed to the capability of learning more complex structures by the CNN, especially for ASC, where both the ambient and foreground sounds can be used to define an acoustic scene. Thus, further decomposition of the feature is not required.

While log mel-spectrogram is the most popular time-frequency representation for ASC, their backbone algorithm, the STFT only computes based on fixed window size. Having a fixed window size limits the profiling of high-frequency resolution. The window function is a trade-off between high frequency and temporal resolution. Based on the Heinsberg Uncertainty Principle, one does not know the best resolution or, in this case, window size for a given acoustic signal. Hence, having a fixed window size will result in missing some sound characteristics (Mallat, 2009, Polikar, 2006).

This resulted in the development of the wavelet transform (Mallat, 2009, Polikar, 2006) that uses a set of wavelet, which is a wave-like oscillation, to convolute on the signal, resulting with the correlation coefficients between the wavelet and the signal.

However, wavelet transform is stable to deformation but not translation invariant (Mallat, 2012, Bruna and Mallat, 2013), which resulted in the development of Wavelet Scattering (WS). WS is a good option for ASC. However, based on the review on various ASC challenges described in Section 1.1, their usage is limited compared to the vastly used log mel-spectrogram. As such, in this thesis, WS is being choosen as the signal processing technique to convert the raw signal into time-frequency representation before setting it as an input for the CNN model.

WS is the focus of this thesis for the pre-processing phase. Hence, the following section will provide a detailed explanation of WS and is organized as follow. In Section 2.2. an explanation on the development of WS is provided, leading to the identification of its limitation. In Section 2.3, this thesis proposed the solution to resolve the limitations of WS specified in Subsection 2.2.3.

## 2.2. Wavelet Scattering

In Section 1.1, it is reviewed that log mel-spectrogram is the most popular time-frequency representation for ASC. However, one of the reasons for their popularity can be linked to the enormous research in using log mel-spectrogram to solve a myriad of acoustic domain challenges (Heittola et al., 2020, McDonnell and Gao, 2020, Gharib et al., 2018, Mesaros et al., 2018, Tchorz et al., 2017, Valenti et al., 2017, Hu et al., 2020, Chandrakala and Jayalakshmi, 2019). The other part resides in its algorithm, which has the property of being translation invariants and stable to deformation. However, log mel-spectrogram suffers from having a fixed timescale as explained in Section 2.1, which means that there are affected by Heisenberg Uncertainty Principle. In addition, (Andén and Mallat, 2014) observed that log mel-spectrogram is only stable to deformation when their timescale is limited to < 25ms. In light of this two disadvantages, (Mallat, 2012) developed Wavelet Scattering.

The intuition on how WS is being developed can start from the basic construction knowledge of log mel-spectrogram. Hence, in Section 2.2.1, how log mel-spectrogram is being constructed and how it leads to the development of wavelet scattering is being explained. Subsequently, in Section 2.2.2, the working mechanism of the scattering transform is further elaborated. Lastly, with the established knowledge of WS, in Section 2.2.3, the limitation of WS is being discussed.

Notably, the discussion of Section 2.2. wavelet scattering is heavily adapted from (X. Y. Kek, C. S. Chin, and Y. Li, “An Investigation on Multiscale Normalised Deep Scattering Spectrum

with Deep Residual Network for Acoustic Scene Classification," *22nd IEEE/ACIS International Fall Virtual Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPDI)*, pp. 29-36, 2021)

### 2.2.1. From log-mel spectrogram to the development of wavelet scattering

Recalled that Log mel-spectrogram start with the Fourier transform, which as discussed in Section 2.1, is the backbone algorithm and has an essential attribute of providing translation-invariant transformation. Following the explanation of (Andén and Mallat, 2014), the Fourier transform of a given signal *x* can be written as:

$$\hat{x}(w) = \int x(u) e^{-iwu} du \tag{2}$$

and if $x_c(t) = x(t - c)$ then $\hat{x}_c(w) = e^{-icw}\hat{x}(w)$, thus, proofing Fourier transform algorithm is indeed invariant to translation.

$$|\hat{x}_c(w)| = |\hat{x}(w)| \tag{3}$$

Since the algorithm to construct STFT is basically a windowed DFT, STFT has also inherited the property of being invariant to translation but at a localized level:

$$\hat{x}_{t,TS}(w) = \int x_{t,TS}(u) e^{-iwu} du \tag{4}$$

If $|c| \ll TS$ then proofing $|\hat{x}_c(ts, w)| \approx |\hat{x}(ts, w)|$, thus STFT is locally time-shift invariant based on a window size/timescale, *TS* (Andén and Mallat, 2014, Andén and Mallat, 2011).

While the above equation (1) to (3) has demonstrated that STFT is locally time-shift invariant, it is still 1 characteristic lacking from being a desirable feature representation. As mentioned in Section 1.2, being stable to time-wrapping deformation is also important. Stable to deformation means that for a representation $\Phi(x)$, a small change to the signal *x* should reflect a small deformation to the representation such that the transformation abides to Lipschitz continuity property. This can be expressed as:

$$\|\Phi(x) - \Phi(x_r)\| \leq C \, sup_t |r'(ts)| \|x\| \tag{5}$$

where $x_r(ts) = x(t - r(t))$ with $|r'(ts)| < 1$ represent a time-warped in signal *x*, and the calculation of the deformation, the Euclidean norm of the representation denotes as $\|\Phi(x) - \Phi(x_r)\|$, is small and there exists a Constant $C > 0$ such that for $x(ts)$ and all *r* with $sup_t |r'(ts)| < 1$. $sup_t |r'(ts)|$ denotes the deformation size with $sup_t$ a time support equal to *TS*, and the constant C > 0 is a measure of stability. However, the Fourier transform is not stable

to deformation due to the dilation shifts at the frequency component, resulting in distortion at high frequencies when small deformation occurs.

As such, to make STFT stable to deformation, mel-scale filters $\hat{\psi}_\lambda$, where $\lambda$ is the centre frequency of each $\hat{\psi}_\lambda$(w), averages the spectrogram along the frequency component to get log Mel-spectrogram. The result of mel-frequency averaging centres the frequency support at $\lambda$ with a bandwidth of the order of $\lambda/Q$ at high frequencies spectrum, similar to constant-Q and $2\pi/TS$ at lower frequencies spectrum. Hence, removing deformation instability that occurs at high frequencies due to dilations and satisfying Lipschitz deformation stability condition (Andén and Mallat, 2014, Andén and Mallat, 2011).

However, Mel-scale averaging will result in the loss of fine-scale information such as vibratos and transient attack, and it gets more severe when the window duration *TS* >25ms as explained by (Andén and Mallat, 2014, Mallat, 2012, Bruna and Mallat, 2013). Hence, log mel-spectrogram is usually optimal with a timescale of about 25ms, resulting in the poor understanding of large-scale structures.

Hence, (Andén and Mallat, 2014, Mallat, 2012, Bruna and Mallat, 2013) developed a strategy to recover the lost information when *TS* is increased. They presented wavelet scattering, which is locally translation invariant and stable to deformation while reducing information loss when T is large (e.g., T > 25ms).

### 2.2.2. Wavelet Scattering Algorithm

Wavelet Scattering, an algorithm developed by (Mallat, 2012), has a similar computational architecture as CNN, depicted in Figure 2.3. In this illustration, *X* is the input signal (waveform) which is being convolute with 'morlet' wavelets computed in a different scale. By cascading the convolution process, higher-order coefficients is being calculated. The final output is the concatenation of all the orders after averaging each order over a given time interval.

Instead of a set of filters that tries to learn the modulation of the signal *x* through convolution operations and backpropagation, wavelet scattering uses a set of filters with a fixed coefficient constructed from a 'mother' wavelet, particularly 'Morlet' wavelet. These filters are termed as wavelet filterbank, symbolized as $\psi_\lambda$ and this convolution operation is also called constant-Q transform, where $\lambda$ is the centre frequency of the wavelet, and $Q$ denotes the number of wavelet filters per octave for a wavelet filterbank.

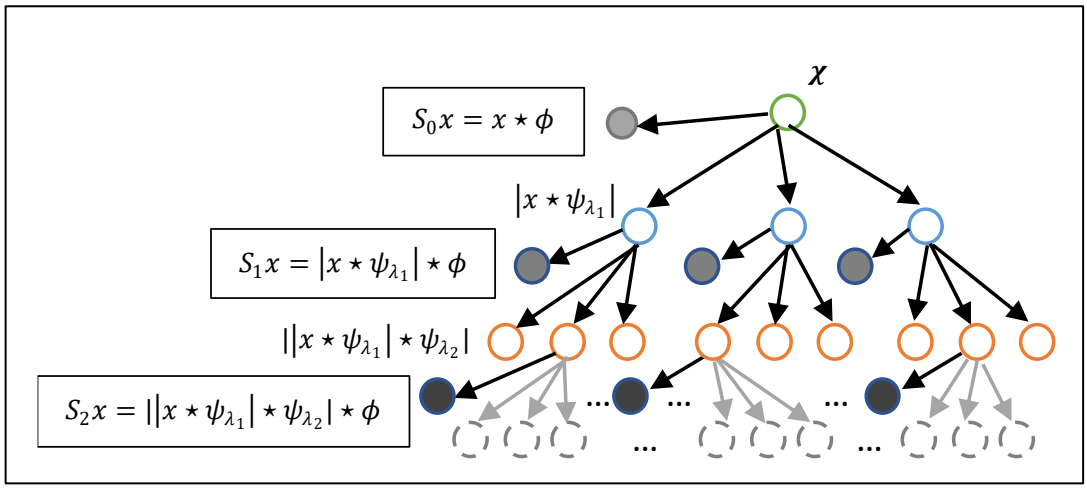



**Figure 2.3.** Wavelet Scattering transforms an adaptation from (Andén and Mallat, 2014, Mallat, 2012, Bruna and Mallat, 2013).

Wavelet Transform is translation covariant (Mallat, 2012). Hence, an averaging function is required to make WS invariant to translation at a local level, similar to how STFT does it. In addition, a complex modulus function $|\star|$ is chosen to optimize stability to deformation [9], where $\star$ denotes convolution operation. The averaging function is in fact a lowpass filter, typically Gaussian filter, represented as $\phi$ is being applied on the output of each layer to ensure translation invariant. The first layer (or order) is expressed as follow:

$$S_1x(ts, \lambda_1) = |x \star \psi_{\lambda_1}| \star \phi(ts) \tag{6}$$

Although the averaging of the modulus wavelet coefficient will result in translation invariant, it has a negative effect of causing information loss from the higher frequency spectrum. Hence, to recover this loss, another convolution, complex modulus, and average filtering is being performed on the modulus wavelet coefficient $|x \star \psi_{\lambda_1}|$ to form the second layer:

$$S_2x(ts, \lambda_1, \lambda_2) = ||x \star \psi_{\lambda_1}| \star \psi_{\lambda_2}| \star \phi(ts). \tag{7}$$

Similar to CNN architecture, DSS can be computed by m-th layers and is written as:

$$S_mx(ts, \lambda_1, \dots, \lambda_m) = |||x \star \psi_1| \star \dots| \star \psi_m| \star \phi(ts) \tag{8}$$

where $S_m$ denotes the m-th order(layer) coefficients, and for the rest of the thesis, the order will be used instead of the layer when describing WS ('layer' is used to illustrate how similar is WS as compared to CNN). $\psi_m$ denotes the m-th wavelet filterbank, and *ts* represents a window length for the averaging filter.

The studies by (Andén and Mallat, 2014, Peddinti et al., 2014, Mallat, 2012, Bruna and Mallat, 2013, Andén and Mallat, 2011) indicate that just computing the first and second-order coefficients will be sufficient to capture most of the signal energy. Consequently, this thesis

will only use first-order scattering coefficients, denote as $S_1x$, and second-order scattering coefficients, denote as $S_2x$, as such:

$$WS = \begin{pmatrix} S_1x(ts, \lambda_1) \\ S_2x(ts, \lambda_1, \lambda_2) \end{pmatrix}. \quad (9)$$

Hence, this thesis also redefined $Q$ that denotes the number of wavelet filters per octave for a wavelet filterbank, into $Q_1$ and $Q_2$. $Q_1$ is a quality factor that denotes the number of wavelet filters per octave for $S_1x(ts, \lambda_1)$, based on (6). Meaning that $S_1x$ has a time support of $\psi_{\lambda_1}$ centred in $\lambda_1$ with a frequency bandwidth of $\lambda_1/Q_1$ for $\lambda_1 \geq 2\pi Q_1/TS$ and $2\pi/TS$ for $\lambda_1 < 2\pi Q_1/TS$. Likewise, $Q_2$ is used for $S_2x$, $\lambda_2$ in (7).

### 2.2.3 Limitation of Wavelet Scattering

**Magnitude Disparity Problem**

In Section 1.2, the first and second-order coefficients have a huge difference in magnitude (see Figure 2.4). The large disparity of magnitude is mainly due to the cascading effect. To resolve the issue, (Andén and Mallat, 2014) proposed 'renormalization' to increase their invariance and decorrelates the coefficients between the lower order to the higher order. This can be achieved by performing a division of a current order against an earlier order, such that $S_1x$ becomes:

$$\widetilde{S_1}x(ts, \lambda_1) = \frac{S_1x(t,\lambda_1)}{|x| \star \phi(ts)} \quad (10)$$

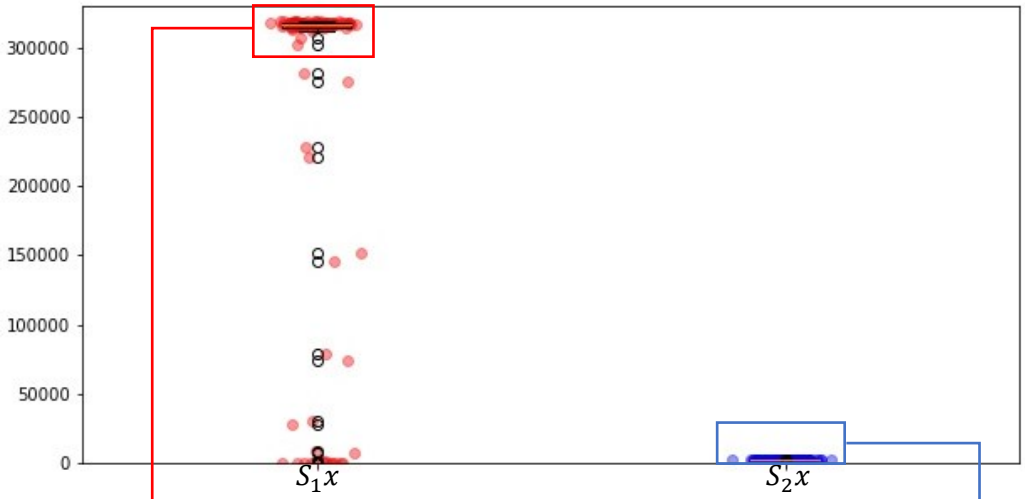
where $S_0x = |x| \star \phi(ts)$ and $S_2x$ becomes:

$$\widetilde{S_2}x(ts, \lambda_1, \lambda_2) = \frac{S_2x(ts,\lambda_1,\lambda_2)}{S_1x(ts,\lambda_1)}. \quad (11)$$

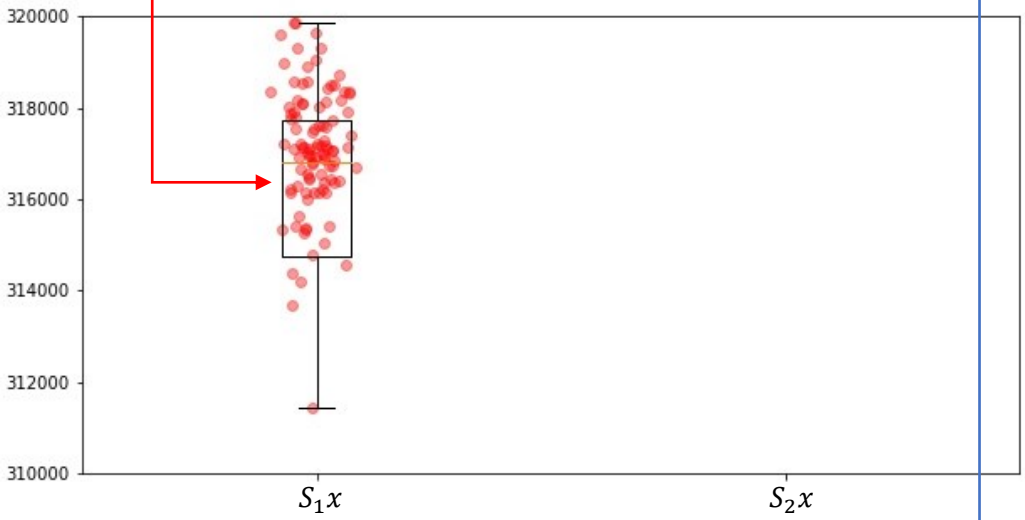
Instead of adapting 'renormalization', this thesis investigates on two-stage CNN architecture and 'batch normalization' described in Chapter 3 to tackle this problem. Based on quantitative analysis by evaluating the DCASE 2021 dataset, the findings are then presented in Sections 5.2 and 5.3. Notably, the two-stage deep neural network (DNN) architecture approach to tackle the magnitude disparity problem has been explored (Peddinti et al., 2014).

**Heisenberg Uncertainty Principle**

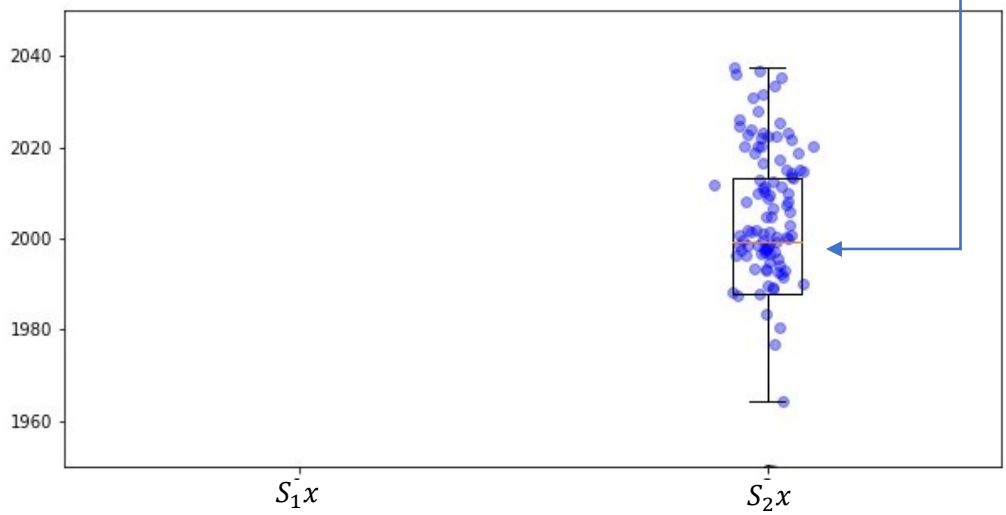
Next, WS is still affected by Heisenberg Uncertainty Principle. It is much more apparent in the case of log mel-spectrogram due to the use of a fixed timescale, while WS is much less obvious. As the building block of WS is the wavelet transform, WS itself has a scaled timescale based on the scaling functions in the theory of wavelets. However, due to an averaging operation on the wavelet transform coefficients observed in (6), (7) and (8), make the representation locally invariant to translation. The timescale (TS) that determines the size of the averaging function




a) Box plot comparison of $S_1x$ against $S_2x$ where the magnitude of the coefficients is between 0 to 350000.




b) Box plot comparison of $S_1x$ where the magnitude of the coefficients is between 310000 to 320000.




c) Box plot comparison of $S_2x$ where the magnitude of the coefficients is between 1950 to 2050.

**Figure 2.4.** Box plot of the coefficients of $S_1x$ and $S_2x$ demonstrating the magnitude disparity problem. (The box plot depicts the total averaged coefficients of $S_1x$ and $S_2x$ using DCASE 2021 Task1a dataset.)

limits the maximum scaling factor of the wavelet. Hence, the reason for multi-timescales/ multi-resolution will be discussed in Section 2.3.

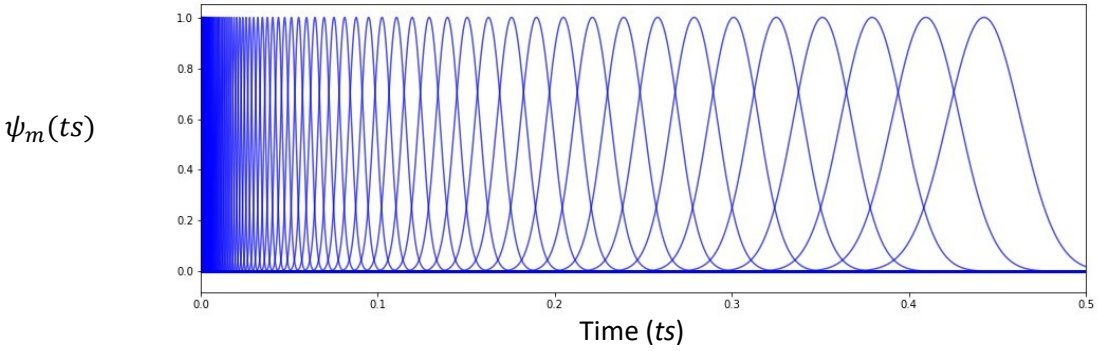
**Large Frequency Dimension**

The second-order has a large frequency dimension (e.g., > 500 rows). The large frequency is due to the addition of fine-scale resolution to capture high-frequency acoustic profiles. While aforementioned that to relieve the effect of Heisenberg Uncertainty Principle is to have multiscale, this solution can be a double-edged sword as more scales give rise to the possibility that certain wavelets transform do not capture any acoustic characteristics. Thus, these wavelet transformed coefficients are redundant. To make matters worse, the large frequency dimension of second-order coupled with redundancy can be detrimental to WS for a low complexity model.

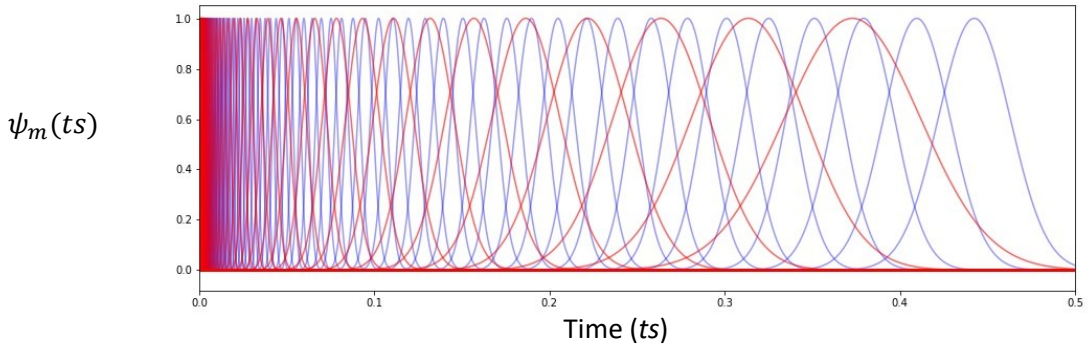
### 2.3. Towards Multi-timescales and the construction of Interleaved Wavelet Scattering

Notably, the problem with fixed timescale is not just limited to time-frequency representation. The studies of scaling properties have also been used in the image domain (Huan et al., 2021, Shahrezaei and Kim, 2020). The work of (Shahrezaei and Kim, 2020) involves the exploitation of discrete wavelet transform scaling properties to decompose synthetic aperture radar sec-ice texture providing multiscale/ multiresolution. While (Huan et al., 2021) uses the concept of multi-scaling on the convolution filter size (M×N) and dilation rate to create multiple feature maps with different receptive fields to improve the reconstruction of super-resolution remote-sensing images. The dilation rate is a control for dilated convolution filter, where the dilation is 1, it is like a standard convolution filter. When the dilation rate increase to above 1, the size of the convolution filter (M×N) will increase except the number of weights. Hence, dilation can be understood as the distance between each weight. Likewise, the concept of 'multi-convolution filters' is relatable to InceptionNet (Szegedy et al., 2015) design, where the problem they address is similar to Heisenberg Uncertainty Principle on convolution filter size. As for making a multi-timescale/multiresolution for WS, (Peddinti et al., 2014, Li et al., 2019) investigated the various $Q_1$. $Q_1$ is the quality factor that determines the number of wavelet filters per octave and also the centred frequency and the frequency bandwidth. Coupling with the observation in Figure 2.5b, having various configurations of $Q_1$ does increase the number of scaling

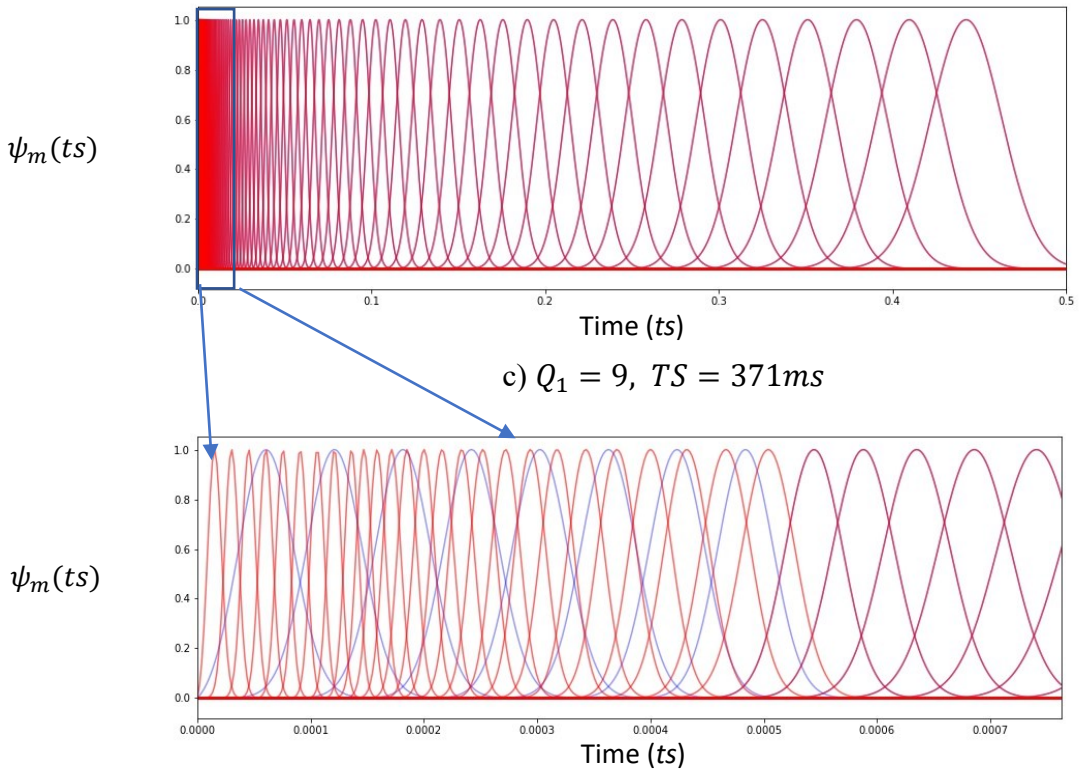
However, the maximum scaling of WS is still limited by TS, where the coefficients from the above wavelet transforms will undergo gaussian filtering illustrated in Figure 2.6, further expressed in (6) and (7). Notably, the increase in TS on the number of wavelet scale is only visible in the fine-scale resolution illustrated in Figure 2.5d, which is in accordance to (Andén and Mallat, 2014, Mallat, 2012, Bruna and Mallat, 2013) analysis of increasing *TS*.




a) $Q_1 = 9,\ TS = 92ms$




b) $Q_1 = 4,\ TS = 92ms$




c) $Q_1 = 9,\ TS = 371ms$

d) Zoom-in scale of $Q_1 = 9, TS = 371ms$

**Figure 2.5.** 'Morlet' Filter banks with different $Q_1$and TS. (The morlet Wavelets in graph (a) is the main wavelet filterbank used for comparison in (b), (c). While (d) is the magnified fine-scale wavelets of (c).)

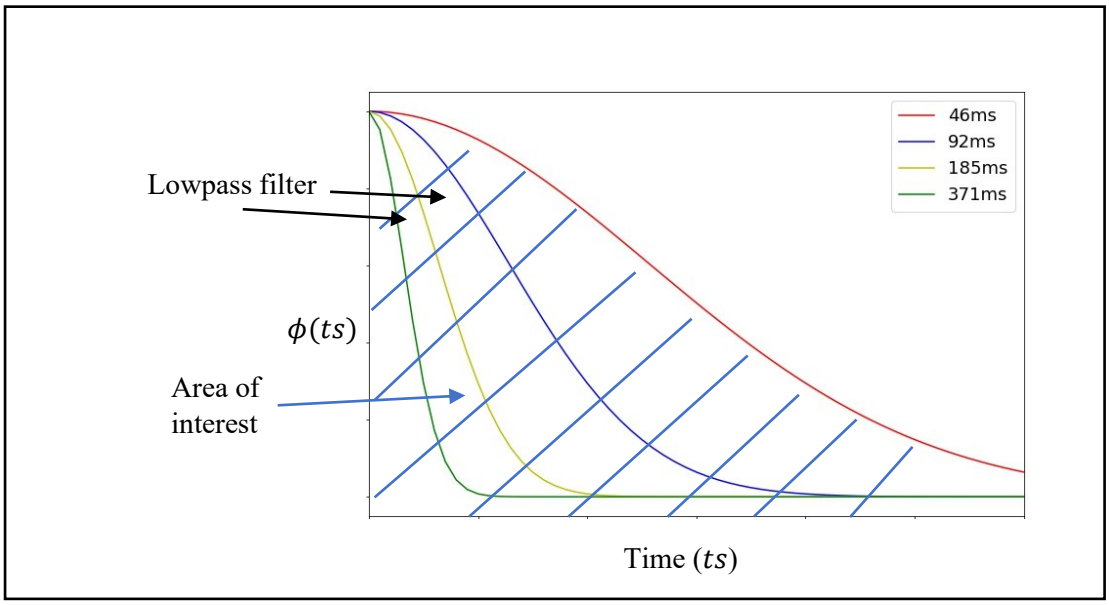


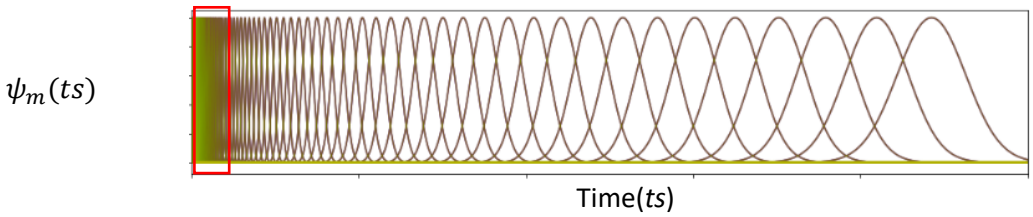

**Figure 2.6.** Gaussian filter, $\phi(ts)$, where $TS = \{46,92,185,371\}ms$. Each line represents the gaussian filter and the shaded area is the area that is being kept.

It is important to investigate TS instead of $Q_1$. In this thesis, an exhaustive evaluation of TS prior to creating a proper multi-timescale WS is being setup. Notably, the investigation of TS has been performed (Andén and Mallat, 2014, Andén and Mallat, 2011). However, their studies are driven towards the analysis of energy dispersion between the first and second-order as TS change, instead of multi-timescale approach.

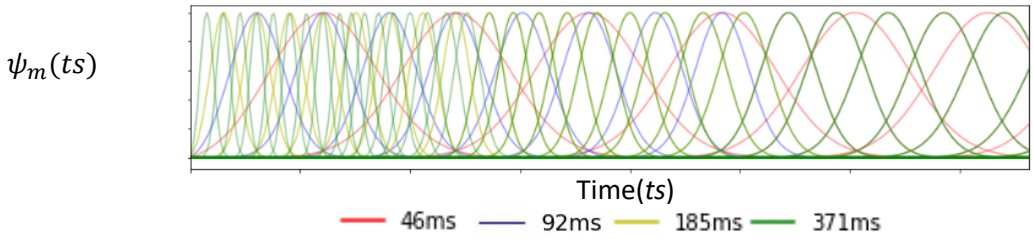
Indeed, by relating Figure 2.6 with Figure 2.7, an observation can be made such that the smaller the timescale, TS, the wider the time support, results in lesser high frequency acoustic profiles captured by the WS. In addition, the wider gaussian filter also corresponds to less energy captured in the second order coefficient and the study of (Andén and Mallat, 2014, Andén and Mallat, 2011) shed lights to the need to strategically choose an ideal group of TS for multi-timescale.

Hence, there is a need to investigate how TS affects the classification model. As such, this thesis has performed an exhaustive evaluation of TS and the discussion is presented in Section 5.4. The evaluation of TS provides a stage to choosing the range of timescales for multi-timescale WS.

As aforementioned in Section 1.3, the application of multi-timescale WS for low complexity model is not ideal due to the nature of higher resolution on the input representation usually means higher computational cost. Hence, an investigation on a simple mixing of different timescales of the first and a single second-order to enrich the multi-timescale capability of WS is being conducted.




a) Full Morlet wavelet filters with 4 different timescales




b) Zoom-in and focuses on fine-scale Morlet filters.

**Figure 2.7.** Morlet wavelet filters with different timescale before averaging. Figure b is the zoom-in view of the red box markup in Figure a.

By coupling just the first and second-order together, the number of input representations in the first stage remains similar to how WS is being handled with a soft relief on the timescale limitation. The idea is to bring diversity to the feature representation without overcompensating on time complexity and size complexity. This is termed as 'Interleaving' WS (IWS). IWS can be expressed as:

$$IWS = \begin{pmatrix} S_1x(ts_1, \lambda_{11}) \\ S_2x(ts_2, \lambda_{21}, \lambda_{22}) \end{pmatrix} \tag{12}$$

where $ts_1$ is the timescale for the first order, and $ts_2$ is the timescale for second-order. As seen in Figure 2.3, WS is being redrawn to provide a better linkage to IWS in Figure 2.8 and 2.9, respectively.

The investigation of various TS provides an appropriate range of timescale for IWS namely: $TS_1, TS_2 \in \{92ms,\ 185ms, 371ms\}$. Tentatively, this also extend the study of energy dispersion theorem (Andén and Mallat, 2014, Andén and Mallat, 2011) through exhaustively combining various first and second-order with different TS . Thus, providing an empirical study on energy dispersion theorem, in Section 5.5.

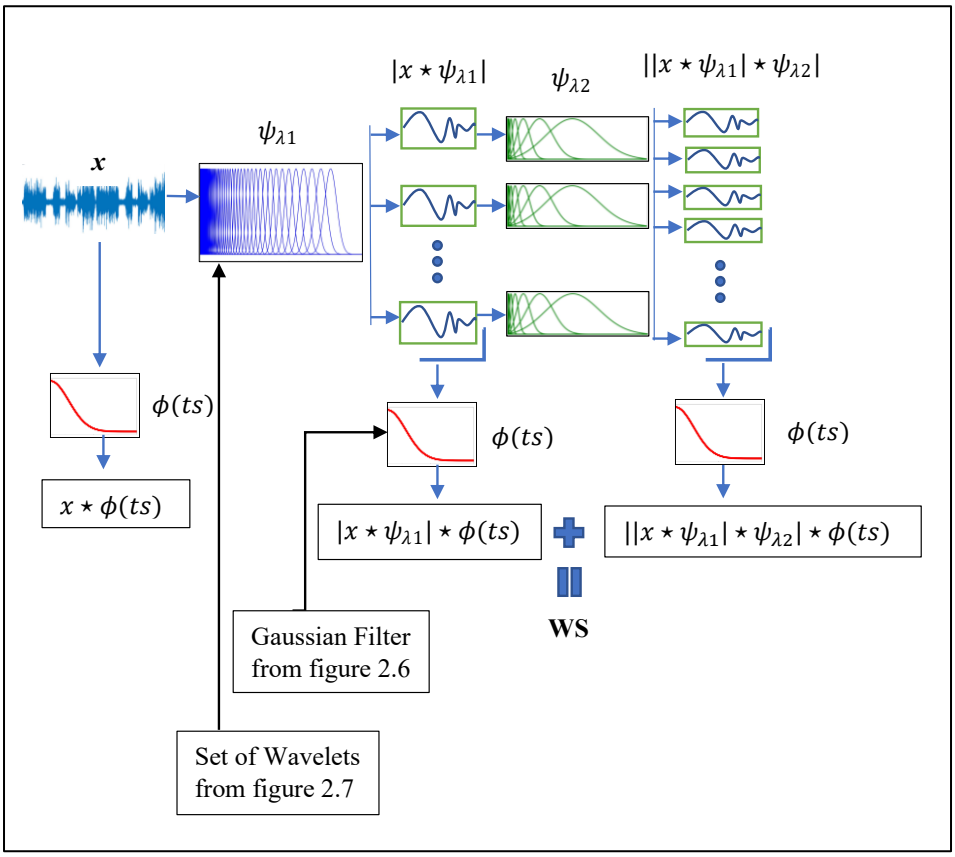



**Figure 2.8.** The construction of WS where $x$ is a given signal. WS is the result of concatenating (1) and (2). $\phi(t)$ is the gaussian filter illustrated in Figure 2.6 $\psi_{\lambda_1}$ and $\psi_{\lambda_2}$ are the set of wavelets with $Q_1 = 9$ and $Q_2 = 1$, where $Q$ denotes the number of wavelet per octave, for first and second-order, respectively.

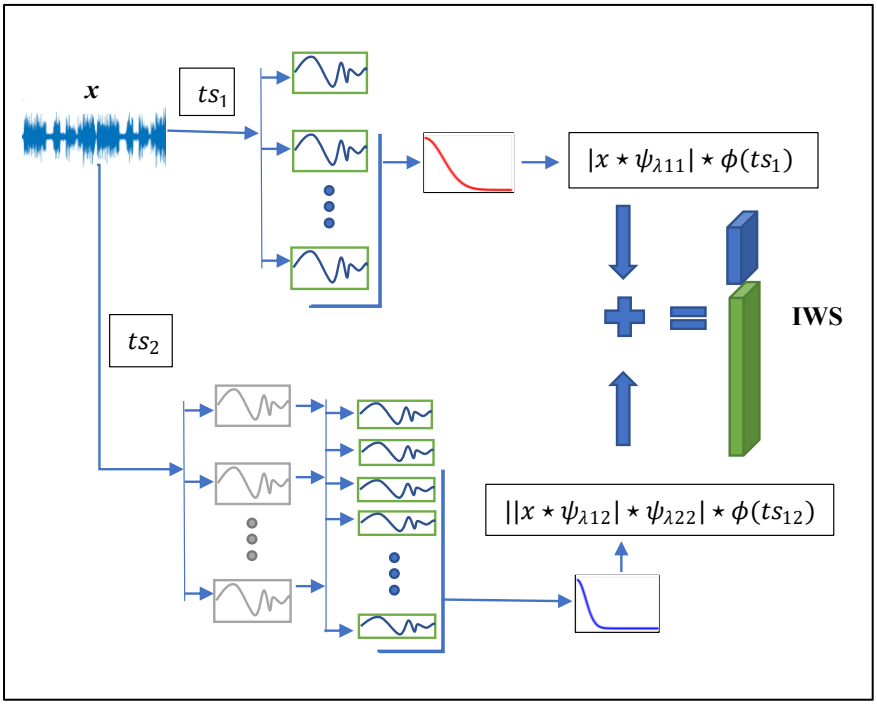



**Figure 2.9.** The construction of IWS where $x$ is a given signal. $ts_1$ is a time index based on $TS_1$ and $ts_2$ is a time index based on $TS_2$. In other word, $TS_1, TS_2 \in \{92ms,\ 185ms, 371ms\}$ and $TS_1 \,!= TS_2$.

## 2.4. Chapter Summary

This chapter introduced the Wavelet Scattering and touched on how it is developed from understanding the properties of Log Mel-Spectrogram. Through this understanding, this paper identified and addressed three weaknesses of Wavelet Scattering being, magnitude disparity problem, Heisenberg Uncertainty Principle, and large frequency dimension. Companying the weaknesses is a proposed solution(s).

In short, magnitude disparity problem suggests the awareness of choosing an appropriate algorithm or classifier when dealing with the first and second-order as they have a huge difference in magnitude. While renormalization is proposed by (Andén and Mallat, 2014), this paper proposed a two-stage CNN architecture and 'batch normalization'.

In this context, the Heisenberg Uncertainty Principle suggests that given a sound recording, we do not know the number of acoustic characteristics present. In order to capture various acoustic characteristics, one need to use multiple timescales. For the case of wavelet scattering, this paper identifies that the averaging function used to make wavelet scattering invariant to translation will limit the maximum scale of the wavelet. Thus, resulting in the possibility of not capturing all the acoustic characteristics. While multi-resolution (concatenation of WS with different $Q_1$) for wavelet scattering is proposed by (Peddinti et al., 2014, Li et al., 2019), this paper proposed multi-timescale coupled with genetic algorithm for feature selection.

Lastly, the large frequency dimension of the second-order becomes a problem when designing a low complexity model. A larger input representation means more convolution operations are required to convolute over the feature maps. This result in an increase in time complexity. Genetic algorithm for feature selection is being proposed again to reduce the frequency dimension. In addition, an enhanced wavelet scattering called Interleaved Wavelet Scattering is also proposed to slightly increase number of scaling of the WS.

# Chapter 3. Convolution Neural Network (CNN)

## 3.1. Introduction

While the development of the algorithm that drives CNN can be dated before 1990s (Krizhevsky et al., 2017, Khan et al., 2020), such as "LeNet" an earlier stage of convolution network (Lecun et al., 2000) for handwritten digit recognition. The fully developed 'framework' and architecture for CNN with practical usage only appeared in 2012, by (Krizhevsky et al., 2017). They combined various algorithms and concepts to create the CNN. The founding CNN model is known as 'AlexNet' for image classification and achieved SOTA for ILSVRC-2012 competition (Russakovsky et al., 2015). Taking the industry and academia by storm, CNN is now a renown deep learning techniques. CNN development is not only restricted to development in application on other domains, but they have also grown into more advanced architectures (Khan et al., 2020, Alzubaidi et al., 2021).

The CNN is not just limited to a 3-Dimensional input, there are expansion towards 1D-CNN which take in an input with 2-Dimensional, and 3D-CNN (Ahmed et al., 2018), which take in 4-Dimensionl input. Notably, 3D-CNN are for medical-image classification, chest CT Scans for COVID-19 (Morozov et al., 2020), CT scans for Tuberculosis prediction (Morozov et al., 2020) and real-time object recognition (Maturana and Scherer, 2015).

In this thesis, the focus is on CNN approaches and for the sake of brevity, CNN will stand for 2DCNN, while 1-Dimensional CNN is referred as 1DCNN and 3-Dimensional CNN as 3DCNN. In this section, the architecture of CNN is being described, followed by the orchestrating of the proposed CNN architecture for WS. The discussions are organized as Section 3.2 and Section 3.3, respectively.

Notably, the contents in Section 3.3 are fully adapted from the following papers.

- X. Y. Kek, C. S. Chin, and Y. Li, “An Investigation on Multiscale Normalised Deep Scattering Spectrum with Deep Residual Network for Acoustic Scene Classification,” *22nd IEEE/ACIS International Fall Virtual Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPDI)*, pp. 29-36), 2021
- X. Y. Kek, C. S. Chin and Y. Li, "Multi-Timescale Wavelet Scattering with Genetic Algorithm Feature Selection for Acoustic Scene Classification," in *IEEE Access*, vol. 10, pp. 25987-26001, 2022

## 3.2. The Skeleton of Convolution Neural Network Architecture

A more modern construct of CNN will consist of multiple components, and in this Subsection, the functionality of each element will be described. The Subsection will be organized as follows: Section 3.2.1 describes the most crucial component of CNN, or what defines a DL algorithm as CNN, which is the convolution layer. In Subsection 3.2.2, we described the pooling layer, which is an algorithm that is commonly used for down sampling the feature map. In Subsection 3.2.3, a discussion on the activation function, which is another essential component of CNN or other DLs, as it allows the network to learn features in multiple dimensional spaces or in other words, provide non-linearity to the networks. Other parts of the network include the loss function, optimizer for the backpropagation algorithm, and weight initialization and regularization, discussed in subsections 3.2.4, 3.2.5, and 3.2.6, respectively. Lastly, two additional components exclusive to the CNN design for ASC to improve the network's performance are included. They are namely: Data Augmentation and Normalization, in Subsection 3.2.7, and 3.2.8, respectively.

### 3.2.1. Convolution Layer

Following the explanation from (Goodfellow et al., 2016, Alzubaidi et al., 2021), a convolution operation on a 2D representation is the idea of convoluting a filter/kernel over it, hence producing a new output, or more commonly known as feature map. As the convolution layers are being stacked together to form the entire CNN architecture, the input and output of a convolution operation are in fact, the feature map (except the first layer, where the feature map is commonly named as input representation). The convolution operation usually works on a 3-dimensional representation. Hence, the feature map for the input and output is usually

represented as a matrix with 3-dimension, (*H* x *W* x *C*) denotes the height, weight, and channel, respectively. Similarly, for the filters, they also have a dimension of (*H* x *W* x *C*).

The properties of the convolution layer:

- The kernels/filters of the convolution layer are the ones that carry the weights of the network. In relation to an ANN architecture, the weights are the network's connections, which means that these are the learnable parameters, usually called the trainable parameters.
- For the case of CNN, it has 3-dimension, and there are cases where *C=1*.
- Currently, it works well on image-liked input representations or information that can be nicely represented in 2-dimension.
- As the filters are being convoluted around the input representation, the weights are being shared for the input feature map. Hence, it is less computationally complex as compared to the fully connected layers while providing the same level or higher level of discriminative power.
- The size of the filters is usually denoted as (*M* x *N*). The larger the size, the more connection it establishes with the feature map. Notably, (Simonyan and Zisserman, 2014) have discovered that stacking two (*3* x *3*) filters together, it provides the same discriminative power as a (7 x 7) filter. Hence, most of the standard CNNs design have fixated the filter size to (*3* x *3*) (Simonyan and Zisserman, 2014), though there are some exceptions to the case in Section 3.3.
- *C* denotes the number of filters. Hence, the more filters the network has, the more learnable parameters the network will have, and this constitute the learning capability of the network. The configuration for the number of filters has to be carefully considered for each stack as it will have an impact on both the complexity and performance of the model. While the more common configuration for the number of filters is to start off with a smaller *C* in the initial layers, then a gradual increment of *2* x *C* as the network goes deeper (e.g., C=32,64,128,256,512,1024), there is also other form of configuration discussed in Section 3.3.
- Stride determines the number of steps the convolution filter will move when convoluting around the feature map. As in this thesis, the focus is on 2DCNN, and the stride will have a tuple of values, (*m,n*). In addition, the default value is usually (1,1), which means that the filters will shift 1 pixel at a time.

- Padding is an option to retain the size of the output after convolution operation such that it has the exact size of the input through padding the edges of the input feature map with zero(s) evenly.
- Dilation controls the distance between each weight. It is determined by the dilation rate with an integer value $\geq 1$. Hence, if the dilation rate is set to 1, it is equivalent to standard convolution layer. An example for a filter of size (3,3), with dilation rate set to 2, the filter size will increase to (5,5). But the number of weights remain the same.

### 3.2.2. Pooling Layer

Pooling is an algorithm that reduces the output of the convolution layers, which is the feature map such that they achieve local translation invariance (Goodfellow et al., 2016). Translation invariance is important as overfitting may occurs if the model starts to learn fine-grain details of the features. We want the network to learn the structure of the features and focus on the essential features. Hence, average pooling (AVP) and max pooling (MaxP) are usually applied after a convolution operation or a series of convolution operations.

### 3.2.3. Activation Function

The activation function is also one of the most crucial components of a CNN. It decides whether this feature in the feature map is important for model learning or not. In other words, it makes the network understand the complex structure in different dimensional spaces or distinguish features that cannot be linearly separated. Of course, the most direct activation function is referred to as the linear function. However, they are only useful if the data can be separated linearly (or separated by a straight line). Hence, in the earlier development, the most common activation function is the logistic sigmoid function, and its extension is the hyperbolic tangent (tanh) function. However, the problem with this function is that stacking multiple Sigmoid or TanH functions can result in a vanishing gradient due to saturation, especially when the network goes deeper (Goodfellow et al., 2016, Krizhevsky et al., 2017).

Hence, to resolve this issue, (Krizhevsky et al., 2017) utilized Rectified Linear Units 'ReLU' that only 'fire' gradients with positive value, which was introduced by (Hahnloser et al., 2000). The activation function 'ReLU' can be expressed as follow:

$$\sigma(x_i) = \max(0, x_i) \begin{cases} x_i, & if\ x_i \geq 0 \\ 0, & if\ x_i < 0 \end{cases} \tag{13}$$

Doing so mitigates the gradient saturation problem, thus impeding the effect of vanishing gradient occurring as the network gets deeper. Of course, this comes with a caveat that the network becomes bias over positive values and negative values are not meaningful. Even

though ReLU has this biasness, it is still the most popular and standard activation function for CNN [cite all the papers that still use ReLU]. However, there is development in this area notably such as variants of ReLU (e.g., Leaky ReLU (Heittola et al., 2020), ELU (Clevert et al., 2016), and SELU (Klambauer et al., 2017)), Swish (Ramachandran et al., 2017), and Mish (Misra, 2020).

Apart from the activation function used in convolution layers, there is specified activation for the classification layer. The choice of activation function used in the classification layers depends on the type of solution or the objective of the task.

In this thesis, the ASC task is a multi-class classification. Hence, 'softmax' activation function is used for the 'final' layer or the classification layer. For ease of differentiating between the activation function used in convolution layers and the activation function used in classification layer, the latter will be referred as 'classification activation function'. Notably, the classification activation function is tightly connected to the loss function as the output is the logits calculated from the softmax function.

### 3.2.4. Loss Function

The loss function is simply the algorithm to calculate the model's performance. Before the network learns through backpropagation, it needs to know how accurate there are when predicting the given classes. Only with this information can the model start learning by correcting what is wrong. In the case of ASC, it is usually a multi-class classification problem. Hence, typically the loss is calculated using categorical cross-entropy. Other loss functions that are not being covered in this thesis are namely: binary cross-entropy, precision-recall area under the curve (PR-AUC) and mean square error (MSE).

### 3.2.5. Optimizer

Backpropagation is the mechanism behind the learning process of all neural networks or the backbone of the neural network, with earlier recognition of the work from 1986 to 1989 (Lecun et al., 2000, Rumelhart et al., 1986). The idea behind backpropagation is to teach computers how to understand a complex problem by providing feedback. As such there are strategies to optimize learning behaviour and the highly used approaches are RMSprop by Geoffrey Hinton, Adaptive Moment Estimation (Adam) (Kingma and Ba, 2015) and Stochastic Gradient Descent (SGD) (Loshchilov and Hutter, 2017). In particular, (Heittola et al., 2020, McDonnell and Gao, 2020, Phaye et al., 2019) used stochastic gradient descent (SGD) (Loshchilov and Hutter, 2017) with warm restart scheduled at a set of epoch indexes being (3, 7, 15, 31, 63, 126, 254) and learning rate covering from 0.1 (max) to 0.00001 (min). Hence, the learning rate will gradually

decrease at each step based on half a cosine curve called 'cosine annealing'. This provides adequate learning with a high learning rate to quickly approach local minimum, then a gradual decrease in learning rate to narrow into the local minimum. The learning rate is being restarted to the maximum learning rate to ensure the model does not get stuck at the local minimum.

### 3.2.6. Weight initialization and Regularization

L1 and L2 regularization are two weight regularization techniques commonly used in the CNN model (Goodfellow et al., 2016, Khan et al., 2020) and usually called weight decay. Like all regularization techniques, its main objective is to alleviate the problem of overfitting by applying a penalty on the weights through the loss function.

As the name suggests, weight initialization is the procedure of setting the starting point of the weight before the network start learning through backpropagation. It determines the ability of the model to converge (Goodfellow et al., 2016). The 'he' weight initialization strategy (He et al., 2015) will be used.

### 3.2.7. Data Augmentation

Data Augmentation is a technique that helps to prevent overfitting and helps the network to generalize better (Goodfellow et al., 2016, Abeßer, 2020, Alzubaidi et al., 2021). It is usually performed on the training dataset, and it modifies the data to prevent strong imprint on the network. In addition, data augmentation is also used to address the lack of data, especially for the case of ASC (refer to Section 5.8.1), where CNN tends to overfit for a small dataset. Notably, the curation of AudioSet dataset with roughly 2.1 million audio recordings and 527 sound event classes (Gemmeke et al., 2017), researchers have moved toward transfer learning exhibited in Section 5.8.2. However, the caveat of training a large dataset such as AudioSet, is the need for enormous computing power. Hence, data augmentation is still a very viable option.

The two most popular approaches used in most of the ASC papers (Heittola et al., 2020, McDonnell and Gao, 2020, Mesaros et al., 2018, Valenti et al., 2017, DCASE, Hu et al., 2020, Chandrakala and Jayalakshmi, 2019, Abeßer, 2020), especially for the 3 ASC datasets, are 'Mixup' and 'SpecAugment'. Mixup data augmentation, proposed by (Zhang et al., 2018a) is the mixing of data features and corresponding labels between one another. For example, if an 'airport' scene sound is mixed with a 'metro' scene sound, the new input representation will be a mix of predominately 'airport' and a small part of 'metro' sound. Mixup is also applicable between individual classes/labels. Hence, by mixing similar scenes, we stand a chance to understand the distinction of sound events for that scene and bring some balance to the dataset. (For example, a beach scene recording can have human's speech or no human's speech.)

Ultimately, providing better representation for the classes/labels. The formulation of Mixup can be seen as follows:

$$x' = \lambda x_i + (1 - \lambda)x_j, \tag{14}$$

$$y' = \lambda y_i + (1 - \lambda)y_j, \tag{15}$$

where $x_i, x_j$ are the raw data, and $x'$ is the new data output based on the mixing algorithm. Similarly, for $y_i, y_j$, and $y'$. The $\lambda$ values denote the amount of mixing using Beta distribution. Typically, $\lambda$ has a value range of [0,1]. Mixup for data augmentation is only used in all the experimentation discussed in Chapter 5.

SpecAugment is a technique adopted from the image domain and redesigned for audio domain. This technique performs a random masking in both the time axis and frequency axis, based on a predefined number of time and frequency axis to masked. The final output dimension after SpecAugment will still be the same as before. However, it was not used in the experimental setup as the preliminary examination has resulted in poorer classification performance with WS. Other data augmentation techniques include time stretching, pitch shifting, dynamic range compression, and random noise (Abeßer, 2020, Hu et al., 2020).

### 3.2.8. Normalization

Normalization is a common technique usually applied to the input features before feeding it to the network to improve convergence speed during the training phrase (Goodfellow et al., 2016). This normalization application is usually called 'Data Normalization', where the process ensures equal distribution of the features. Since, normalizing the features does speed up the convergence time, (Ioffe and Szegedy, 2015) incorporated the normalization technique into the CNN architecture and termed it 'Batch Normalization' (BN) as the normalization is seamlessly integrated into the mini-batch training setup. BN is applied on the feature map(s) of the network and is used to address the issue of internal covariate shift cause by the increase variation between the weights of the network during training. If a network suffers from internal covariate shift, other than slower convergence time, the network may experience vanishing or exploding gradients, which is the Achilles' heel of any neural network. Hence, BN has been used by several CNN networks described in Section 3.3.

In fact, BN is a feature regularization technique; unlike the usual normalization algorithm, it has additional learnable parameters. The formulation of BN is somewhat similar to the data normalization as:

$$y_i = \frac{\gamma(x_i - \mu)}{\sigma} + \beta \tag{16}$$

where $y$ is the output of the normalization, $x$ is the input representation, and $i$ represents a feature of $x$. The learning parameters $\gamma$ and $\beta$ are the scaling factor and offset factor, respectively. While $\mu$ and $\sigma$ are the mean and standard deviation (std) computed by:

$$\mu = \frac{1}{mb}\sum_{k \in S_i} x_k\,, \quad \sigma = \sqrt{\frac{1}{mb}\sum_{k \in S_i}(x_k - \mu)^2 + \epsilon}, \tag{17}$$

with $\epsilon$ as a small constant, $S_i$ is the set of pixels in which the mean and std are computed, and $mb$ is the size of this set which is the size of the mini-batch. In addition, $S_i$ is a 4D Vector with *(N, F, T, C)* which reflects a typical CNN feature output size when training by mini-batch. Lastly, $i$ is an index of the features $(i_N, i_F, i_T, i_C)$.

Besides applying BN on feature map, the BN algorithm can also be directly used onto the input representation, as exhibited in the works of (Chang et al., 2021, He et al., 2016b) . This technique of applying BN on feature map is known as '*input norm*'. Both BN and *input norm* are essential for WS, as there lies a huge disparity in magnitude between the first and second-order coefficients explained in Chapter 2. Hence, BN and *input norm* can be used to balance the distribution of the coefficients throughout the network, and an investigation on the effectiveness of *BN* and *input norm* is presented in Section 5.3. In addition, an exploration is being performed on other normalization techniques such as 'Layer Normalization' (Ba et al., 2016), 'Instance Normalization' (Ulyanov et al., 2016) and 'SubSpectral Normalization' (Chang et al., 2021). Their differences are illustrated in Figure 3.1.

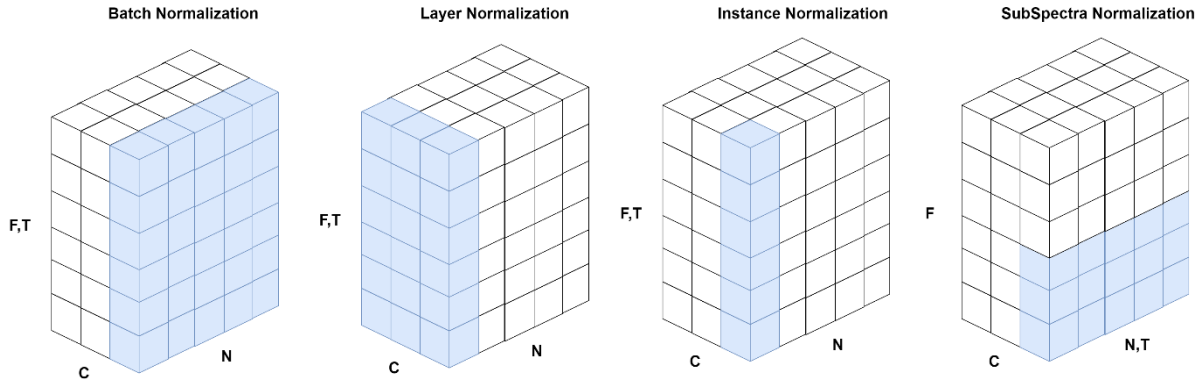



**Figure 3.1.** Illustration of the set $S_i$ for the input normalization, an adaptation from (Kek et al., 2021)

In summary, the success of CNN architecture is a precise combination of the algorithms presented above. In particular, an investigation on various normalization techniques on WS is conducted due to the vast disparity of magnitude between the first and second order, in addition to adopting a two-stage CNN architecture framework.

### 3.3. Two-stage Convolution Neural Network Architecture Framework

Before the introduction of the two-stage convolution neural network architecture approach, the combination of WS with CNN has already been well established (Ren et al., 2018, Zhang et al., 2019, Peddinti et al., 2014, Li et al., 2019). In addition, the framework of two-stage CNN is being featured (Peddinti et al., 2014, Li et al., 2019). In Simplicity, a two-stage CNN is theoretically split into two stages. The first stage analyzes different features individually, and the second combines them to build an overall understanding. In the case of (Peddinti et al., 2014), share a similar ideology about solving the magnitude disparity problem. Indifferent, they use a combination of CNN and Deep neural network (DNN) architecture. DNN in this context refers to stacks of fully-connected layers or neural network with more than one hidden layer. The CNN is used to learn the first-order coefficients, and the DNN is used to learn the second-order coefficients. They (Li et al., 2019) used the first stage to train two features: WS and frequency scattering. Frequency scattering is introduced in (Andén and Mallat, 2014) with the intention to make WS transposition invariance in frequency.

Inspired by the success of residual network design for ASC by (McDonnell and Gao, 2020, Phaye et al., 2019), the CNN topology is emulated onto WS to resolve the huge disparity in magnitude between the first and second-order coefficients that is believe to impede the learning process of CNN. This thesis termed it as two-stage CNN (TSCNN) architecture framework. The inituition of (McDonnell and Gao, 2020, Phaye et al., 2019) is to split the log mel-spectrogram into frequency bins. The binning process divides the log mel-spectrogram into large groups of low, mid, and high-frequency spectrum. Then, in the first stage, the CNN digests this set of frequency bins concurrently parallel, as illustrated in Figure 3.2. In the context of CNN architecture design, this process is called 'group convolution' (Krizhevsky et al., 2017).

In the first stage, instead of splitting the frequency into different bins, it is conveniently split into first and second-order. Furthermore, the first stage is named *'specialized learning'* as the purpose of the first stage is to allow the network to have independent learning of the first and second-order. The second stage is then named '*centralized learning*', where learning takes place after the concatenation of the two features illustrated in Figure 3.2. The concept of specialized learning and centralized learning is set up when explaining the configuration of TSCNN and

extending TSCNN as a framework instead of a CNN architecture. The ratio of specialized and centralized learning for TSCNN framework can be highly dependent on the feature representation such as log mel-spectrogram and wavelet scattering and the convolution block used (e.g., residual convolution block or mobileNet convolution block design). Notably, the essence of TSCNN framework is being used in all the proposed models. (The term 'TSCNN'

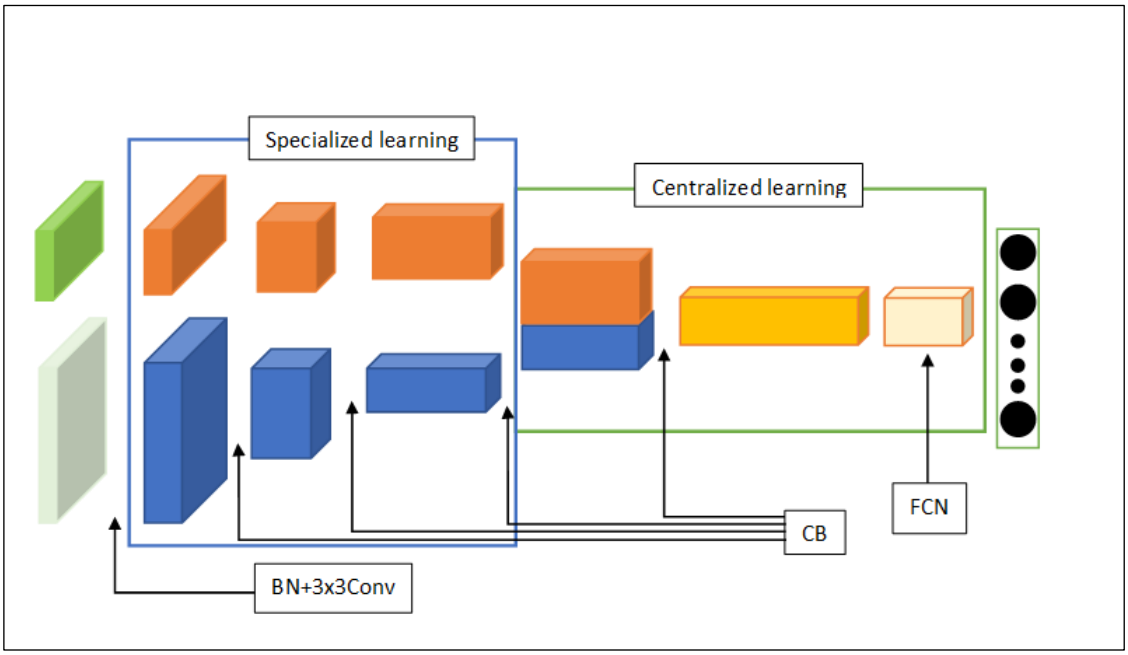



**Figure 3.2.** Two-stage CNN (TSCNN) architecture. The first two boxes are the input representation of the network and represent the first and second-order, respectively. The rest of the boxes reflect the feature maps after each convolutional layer or convolution block (CB), except for the last box being the feature map after FCN.

means the model, and 'TSCNN framework' means the two-stage CNN architecture framework) With the introduction of TSCNN architecture, let's deconstruct TSCNN and examine the convolution block (CB) design which is the fundamental building block of CNN. In other words, CNN is typically built with stacks of similar convolution blocks, except that the number of channels increases as the network goes deeper.

Following (McDonnell and Gao, 2020) designs, the convolution block is pieced together with the below components as shown in Figure 3.2:

**3-by-3 convolution layer (3x3Conv) design** has revolutionized most of the current CNN architecture (McDonnell and Gao, 2020, Huang et al., 2018, He et al., 2015, Srivastava et al., 2015, He et al., 2016b, Howard et al., 2017, Li et al., 2019, Howard et al., 2019, Ma et al., 2018, Zhang et al., 2018c, Iandola et al., 2016, Phaye et al., 2019, Khan et al., 2020), which usually follows the combination of stacking 2 3x3Conv. (Simonyan and Zisserman, 2014) developed 3x3Conv to reduce the computational complexity of the model. They investigated the

possibility of reducing the filter size from (7 × 7) or (5 × 5) to (3 × 3) and discovered that a stack of 2 3x3Conv has the same discriminative capability as a larger counterpart while reducing the computational complexity of the model. Hence, all CB uses a 3x3Conv as the convolution layer in Figure 3.2. Notably, zero padding is applied for all 3x3Conv(s).

**Average pooling (AvgP)** is mainly used to reduce the dimensionality of the feature map as the network goes deeper, as explained in Section 3.2.2. Notably, average pooling does not involve learning or trainable weights. An average pooling with a filter size of (3 × 3) is used.

**Residual Network** is another revolutionary design created by (He et al., 2016b). In simplicity, a residual network is a network where right after each convolution action, there is a propagation of gradient from the previous feature map to the new feature map. Let $x$ be a feature map, and the output of BN+ReLU+3x3Conv is $x'$, the result of a residual network will be depicted in Figure 3.3:

$$F(x) = x + x'. \tag{2}$$

While average pooling is abbreviated as AvgP and the product of average pooling is represented as *AvgP(x)*. By propagating the previous gradient forward, degradation no longer happens, even when the network goes extremely deep (e.g., 1000 layers).

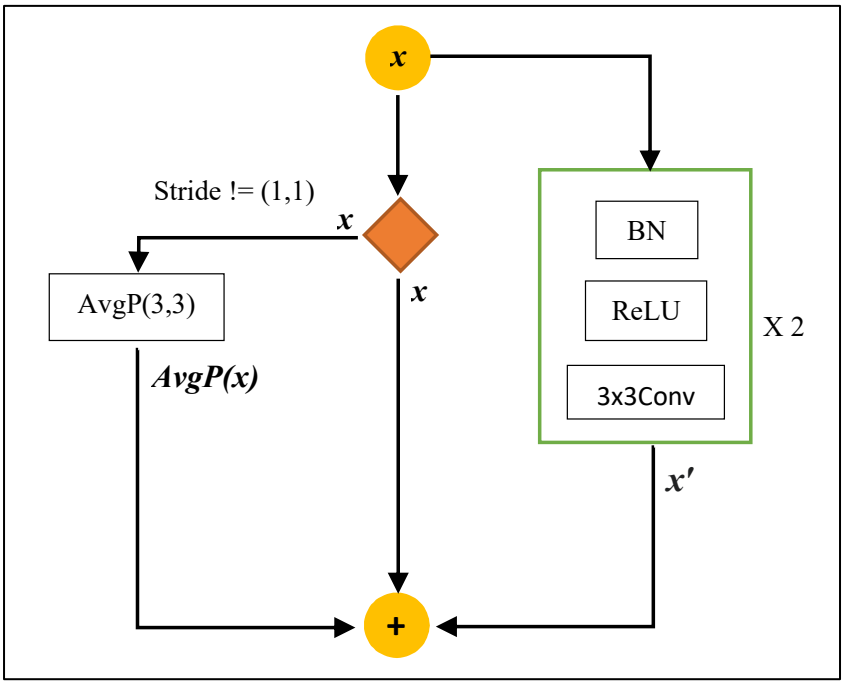



**Figure 3.3.** Convolution Block (CB) design following pre-Normalization residual module approach proposed by (He et al., 2016b).

**Batch normalization (BN)** to accelerate and smoothen the learning process. BN applies a standard normalization on a batch of feature maps using mean and standard deviation and is discussed in Chapter 3, Section 3.2.8.

**Fully convolution network**, better known as FCN (Shelhamer et al., 2017), is the last convolution block that graciously completes the CNN before the softmax classification layer. It earns its name as FCN as no fully connected or dense layer is used to complete the CNN, commonly found in older models (Krizhevsky et al., 2017, Simonyan and Zisserman, 2014, He et al., 2016b). FCN is constructed with a convolution layer with a filter size of (1×1), and the channels are compressed such that it is numerically similar to the number of classes. It is followed by a global average pooling (GAP) that flattened the feature map channel-wise into a vector as the softmax layer.

Lastly, TSCNN framework is also extended for multi-timescale WS and named it TSCNN-2 as illustrated in Figure 3.4. Subsequently, only the TSCNN-2 architecture is presented in a tabular format in Table 3.1 as it has rather similar architecture as TSCNN. In Table 3.1, $S_1x_1, S_2x_2, S_2x_3, S_2x_4$ represents first order with a $TS_1 = 46ms$, second-orders with $TS_2 = 92ms, TS_3 = 185ms, TS_4 = 371ms$ , respectively. The column "Strides" determine the number of strides is performed during the convolution operation and column "C" determine the number of channels. Lastly, column "output feature map" presents the dimension of feature maps after the convolution block. SCB stands for stacked Convolution Block or CB.

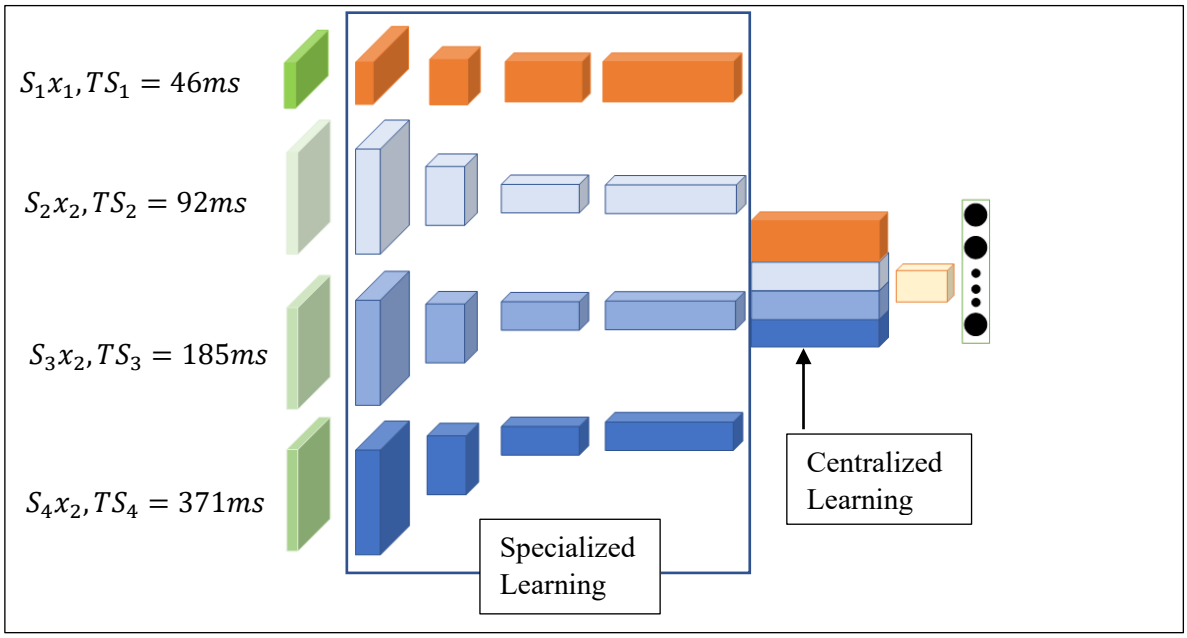



**Figure 3.4**. TSCNN-2 architecture extended from TSCNN. TSCNN-2 can accommodate 4 WS of different timescales.

TSCNN-2 ARCHITECTURE

| Convolution Block | | Strides | *C* | OUTPUT FEATURE MAP |
|---|---|---|---|---|
| Input: $S_1x_1, S_2x_2 S_2x_3, S_2x_3$ | | - | - | (87,248,3), (256,120,3), (306,120,3), (361,120,3) |
| BN+3x3Conv | | (1,2) for $S_1x_1$ (2,2) for $S_2x_2$, $S_2x_3$ & $S_2x_4$ | 32 | (87,124,32), (128,60,32), (153,60,32), (181,60,32) |
| SCB1 | 1a | (1,1) for all | 32 | (87,124,32), (128,60,32), (153,60,32), (181,60,32) |
| | 1b | (1,1) for all | 32 | (87,124,32), (128,60,32), (153,60,32), (181,60,32) |
| SCB2 | 2a | (1,2) for $S_1x_1$ (2,2) for $S_2x_2$, $S_2x_3$ & $S_2x_4$ | 64 | (87,62,64), (128,30,64), (153,30,64), (181,30,64) |
| | 2b | (1,1) for all | 64 | (87,62,64), (64,30,64), (77,30,64), (91,30,64) |
| SCB3 | | Same as SCB2 | 128 | (87,31,128), (32,15,128), (39,15,128), (46,15,128) |
| SCB4 | | Same as SCB2 | 256 | (87,16,256), (16,8,256), (20,8,256), (23,8,256) |
| AvgP for $S_1x_1$ | | (1,2) | - | (87,8,256) |
| Concat | | - | - | (87+16+20+23,8,256) |
| BN+ReLU +1x1Conv | | - | 512 | (146,8,512) |
| FCN+softmax | | - | 10 | 10 |

**Table 3.1.** TSCNN-2 Architecture. This Table presents the proposed TSCNN-2 architecture design. The first row is the input representation which is fed to TSCNN-2.

Each stack consists of 2 series of CBs with different strides, and each CB is applied onto each input feature, $S_1x_1, S_2x_2, S_2x_3, S_2x_4$. Other than SCB1, which has a different strides setup, the rest of the SCBs follow a stride of (1,2) for $S_1x_1$ and (2,2) for the rest, $S_2x_{2,3,4}$. For the first series and stride is equal to 1 for all features for the second series (refer to SCB2a,2b).

### 3.4. Chapter Summary

This chapter introduce the building blocks of Convolution Neural Network with an emphasis on the 'Batch Normalization' algorithm. In Chapter 2, it is discussed that BN can be used to tackle the magnitude disparity problem. The next part of this chapter described the proposed two-stage CNN architecture which consists of stacks of residual convolution block and end with a FCN. In this thesis, it is extended as a framework where the first stage is termed as specialized learning and the second stage is termed as centralized learning. In addition, an ablation is conducted to determine the optimal ratio between specialized and centralized learning for wavelet scattering. Two CNN architectures, being TSCNN and TSCNN-2 is built with the concept of the two-stage CNN framework. TSCNN supports wavelet scattering as the input representation, while TSCNN-2 supports multi-timescale wavelet scattering.

# Chapter 4. Low-Complexity Modelling

## 4.1. Introduction

In this section, the proposed approaches are further elaborated. In Section 4.2, a discussion on how genetic algorithm for feature selection is being used to tackle the large frequency dimension of WS, IWS, and multi-timescales WS. Subsequently, in Section 4.3, while presenting GA for Feature Selection 'wrapper' approach', the entire implementation of WS(s), GA, and CNN is incorporated. In Section 4.4, the construct of the new low complexity CNN which includes convolution factorization, stochastic manipulation, especially shuffling modules, and two-stage CNN architecture is being elaborated. In Section 4.5, a hypothetical design is drafted, suggesting that GAFS could be easily implemented with the mobile shuffling network.

Notably, the discussion of Section 4.2 and Section 4.3 are adapted from the following publication.

- X. Y. Kek, C. S. Chin and Y. Li, "Multi-Timescale Wavelet Scattering With Genetic Algorithm Feature Selection for Acoustic Scene Classification", in *IEEE Access*, vol. 10, pp. 25987-26001, 2022

The discussion of Section 4.4 is adapted from the submitted publication.

- X. Y. Kek, C. S. Chin, Y. Li, "An Intelligent Low-complexity Computing Interleaving Wavelet Scattering based Mobile Shuffling Network for Acoustic Scene Classification", in *IEEE Access*, vol. 10, pp. 82185-82201, 2022.

### 4.2. Genetic Algorithm for Feature Selection

In this thesis, feature selection (FS) (Liu and Zhao, 2014, Vergara and Estévez, 2014, Tang et al., 2014, Blum and Langley, 1997, Dash and Liu, 1997) is selected as the methodology to reduce the dimensionality of WS. Feature selection comes under a broader topic called dimensionality reduction, and the other common method under this topic is feature construction technique (e.g., Principal Component Analysis (PCA)). The difference between these two methods is that feature construction technique compute relationship among the features in a separate feature space, and the final output is usually the newly mapped features. In contrast, feature selection returns a subset of the original features.

The reason can be broken down into three factors: Firstly, WS is locally translation invariant and stable to deformation (Andén and Mallat, 2014, Mallat, 2012, Bruna and Mallat, 2013). These properties are paramount to acoustic classification tasks. Hence, it is vital to retain the feature as what it is. Secondly, WS consists of first and second-order coefficients, and the first and second-order coefficients have relatively huge differences in magnitude as explained in Section 1.2 (Peddinti et al., 2014, Kek et al., 2021). Thus, the translation algorithm might create a bias towards the first-order coefficients as their magnitude is larger. In addition, to suffering from internal covariant shift (Ioffe and Szegedy, 2015). Thirdly, the second-order presents a sparse representation (Mallat, 2012; Bruna and Mallat, 2013; Andén and Mallat, 2014), leading to carefully selecting the translation algorithm.

Feature selection is broadly studied and has been used in many domains (Liu and Zhao, 2014, Vergara and Estévez, 2014, Tang et al., 2014, Dash and Liu, 1997, Uysal, 2018). A traditional approach such as an exhaustive search, is practical for a small number of features but not feasible with many features. Exhaustive search requires $O(2^p)$ computational costs, where $p$ is the number of features. Hence, employing a heuristic technique (de la Iglesia, 2013) is more effective in finding the optimal feature subset without searching through every single combination or NP-Hard problem. Common Heuristic search techniques such as sequential forward/backward selection, "plus-l-take away-r" is an algorithm that combines sequential forward selection and sequential backward selection where l determines the number of steps going forward. The r determines the number of steps going backward after moving forward (XUE et al., 2013, Xue et al., 2016, Saeys et al., 2007). In addition, sequential forward/backward floating selection can be easily implemented. However, they can only provide local decisions without a globally optimal solution. Hence, to expand the search space into a global decision, genetic algorithm (GA) (Goldberg and Shakespeare, 2002, Holland, 1984, Goldberg, 1988, Xue et al., 2016) is being adapted, which is a type of evolutionary algorithm

(EA), and it emulates biological evolution to find the optimal feature subset. The application of GA for feature selection (GAFS) has been well studied (Uysal, 2018, Oreski and Oreski, 2014, XUE et al., 2013, Saeys et al., 2007, Xue et al., 2016, Sloss and Gustafson, 2020, Yang and Honavar, 1998, Jung and Zscheischler, 2013, Siedlecki and Sklansky, 1993, Hong and Cho, 2006, Zhang et al., 2018b, Ansari et al., 2021, Alexandre et al., 2007, Ibrahim et al., 2020, Nguyen et al., 2021).

Genetic algorithm (GA) is an adaptive heuristic search and optimization algorithm that is designed to influence Darwinism or Darwinian theory (Goldberg, 1988, Goldberg and Shakespeare, 2002, Holland, 1984, Duzdevich and Darwin, 2014) and is the most common form of EC. GA finds the best solution by a "survival of the fittest" concept, much like how nature evolves through mixing and selecting the best individuals for the reproduction of the next generation. The major draw to implementing GA is it simplicity of operation while yielding great results (Uysal, 2018, Oreski and Oreski, 2014, XUE et al., 2013, Saeys et al., 2007, Xue et al., 2016, Sloss and Gustafson, 2020, Yang and Honavar, 1998, Jung and Zscheischler, 2013, Siedlecki and Sklansky, 1993, Hong and Cho, 2006, Zhang et al., 2018b, Ansari et al., 2021, Alexandre et al., 2007, Ibrahim et al., 2020, Nguyen et al., 2021). Hence, GA can be applied for specified use cases, and in this thesis, GA for feature selection is the ideal framework to reduce the dimensionality of WS, which has been around since 1990s (Xue et al., 2016, Siedlecki and Sklansky, 1989).

GAFS has the same framework as GA, but with the specified objective to maximize or maintain classification accuracy while using a subset of the original features. In Chapter 1 and Chapter 2, it is observed that the second-order of a wavelet scattering has a relatively large frequency dimension (e.g., 500 to 1000 features), which translates to higher time complexity when coupled with the CNN model. In line with the direction toward building a lower complexity model, there is a need to perform dimensionality reduction in the second-order. In addition, a hypothesis is realized such that there is redundancy or irrelevancy in the second-order due to the 'fine-scale' resolution. The occurrence of redundancy or irrelevancy can impede the learning capability of the CNN, while also adding unnecessary time complexity. Hence, getting an optimal subset of the features has two advantages; improve classification accuracy and decrease computational time. Following (Goldberg and Shakespeare, 2002, Holland, 1984, Xue et al., 2016, Kek et al., 2022), the algorithm of GA for feature selection when applied on WS is presented in Table 4.1.

GA FOR FEATURE SELECTION (WRAPPER ALGORITHM

Initialized Population:
1. **For each** h in total_number_of_population **do**
   (a) WS = (F,T`) and T` /= *T*.
   (b) **For each** i in F **do**
      (i) chromosome[i] = random(0,1)
2. **For each** i in total_number_of_generations **do**
   (a) **For each** j in set_of_chromosomes **do**
      (i) Evaluate the classification accuracy (result is based on cross-validation)
   (b) **Sort** Population **by** classification accuracy
      **Delete** i number of chromosomes starting from the bottom. (i = number_of_offspring)
   (c) Crossover operation:
      **For each** k in number_of_offspring **do**
      (i) offspring[k] = parent[k]*0.5 + parent[k+1]*0.5
   (d) Mutation operation:
      **For each** k in number_of_offspring **do**
      (a) Randomly select genes in offspring[k] based on a given mutation rate.
      (b) **For each** q in selected_genes **do**
         (a) **if** gene == 1, gene = 0, **else** gene = 1
         (e) **Append** Population with offspring

**Table 4.1.** GA for Feature Selection (Wrapper) Algorithm. This Table presents the algorithm for GA for the Feature selection wrapper approach fully adapted from (Kek et al., 2022).

In Table 4.1, The proposed GA consists of two main loops. The first loop is the initialization of the Population. The chromosome is created based only on a binary mapping on the frequency axis. Hence, each row of WS is represented by a binary, 0 means not selected, and 1 means selected. (Refer to Figure 5). The second loop is an iterative process to find an optimal feature subset. The second loop consists of 5 steps: (2a) Evaluate the fitness value of the Population, (2b) Remove the weaker chromosomes from the Population, (2c) and (2d) are operations used to reproduce offspring. (2e) Add the offspring to the Population. The loop stops upon reaching the total number of generations. The implementation is very similar to a typical GA framework as GA naturally visualizes the features as a set of binary string encoding, where 1 form the final subsets and 0 for those not selected. In addition, a more detailed explanation of the GA processes is described below:

### 1) INITIALIZATION of population

The first step of GA is usually the creation of the population (Refer to Table 4.1, (1)), which is the pool of selected different combinations of features. Each set of features represents an input representation, and each feature represents an attribute of the input representation. Relating to GA and biology, a feature is termed a gene, and a set of features is termed as a chromosome.

The initial step involves the creation of a pool of chromosomes. The chromosomes are created through a random process of combining various genes. Thus, a population is formed containing variants of the chromosome. The Population will then undergo evolution, which is the continuous process of weeding out the weaker chromosomes while encouraging reproduction between the fittest chromosomes, aiming to achieve the chromosome with the 'best' genes. As such, the Population will evolve continuously from the initial Population until it meets an end condition.

**2) FITNESS FUNCTION**

The fitness function is the evaluation algorithm to determine the strength of the chromosomes. Hence, the product of the fitness function is the fitness value. In this case, the fitness function is based on the model classification accuracy, corresponding to Table 4.1, row 2.a.i.

**3) CROSSOVER**

Both crossover and mutation are techniques used to create new chromosomes based on the selected group of stronger chromosomes from the population. Hence, the concept of parent and offspring, where the offspring are the newly created chromosomes. Crossover is the technique of mixing the genes of a pair of parents to produce offspring.

**4) MUTATION**

Next, mutation is the changes on the offspring genes, such that given a chromosome represented in binary format, [0,0,0,1,1, 1], when mutation occurs, the mutated chromosome or offspring will become [1,0,0,0,1,1]. Mutation is crucial to prevent the GA from getting stuck at local optimal, however, high mutation rate might impede the GA from finding an optimal.

**5) END CONDITION**

Lastly, the end condition or termination criteria is the algorithm to determine the convergence of GA operation. This can be set as the number of iterations or generations. A generation is the process of step 2 to step 4.

While the algorithm for GAFS using wrapper approach is depicted in Table 4.1, it is noteworthy that there lies other GAFS approaches such as ‘filter’ approach (Blum and Langley, 1997), ‘embedded’, and ‘hybrid’ approaches (Tang et al., 2014, Yang and Honavar, 1998, Xue et al., 2016). While the hybrid technique utilized a combination of filter and wrapper approaches (Dash and Liu, 1997, Uysal, 2018), the embedding approach performs feature selection as part of the pattern recognition algorithm (Spolaôr et al., 2011).

The wrapper approach is commonly adopted (Xue et al., 2016) to evaluate the goodness of the selected features based on the classification model, usually the model's classification accuracy. Hence, in GA wrapper approach context, the fitness function directly relates to the choice of the classification algorithm.

The objective of the filter approach is to drastically reduce the computational cost of the wrapper approach caused by multiple training and evaluation of the classification model such as deep CNN models. Although it reduces the time taken to get the optimal feature subset, their accuracy performance is still inferior compared to the wrapper approach based on the comparative studies of (Saeys;Inza and Larrañaga, 2007; de la Iglesia, 2013; Xue et al., 2016). This can be attributed to the core algorithm that evaluates the chromosome's fitness. On the other hand, the filter approach uses an independent and less computationally intensive algorithm to evaluate the fitness of the features subset; only after finding the optimal features subset based on the filter-based algorithm will it be evaluated on the classification algorithm. Filter algorithms such as information theory, correlation-based approach, or distance measure approach might only provide limited interaction between the features (de la Iglesia, 2013). Furthermore, what is deemed insignificant to the filter approach algorithm might be considered useful by the wrapper approach.

In the case of WS, the first and second-order coefficients have a large disparity in magnitude. In addition, getting a viable filter-based algorithm to be sparse representation of the second-order coefficients is a challenge. Hence, driven by the disadvantage of using a filter approach, this thesis uses the wrapper approach, which provides a seamless framework to integrate WS with CNN, tapping into the discriminative prowess of CNN (Kek et al., 2021).

### 4.3. Genetic Algorithm with CNN architecture

Following the framework (Siedlecki and Sklansky, 1989), this thesis proposed the implementation of GA for WS, which has a meta-heuristic nature to find the global optimal feature subset. WS has a specified feature structure (Tang;Alelyani and Liu, 2014) (Time-frequency representation). Unlike image classification, where features can be flattened into a 1D array and selected pixel-wise, it does not work well for time-frequency representation such as wavelet scattering. Hence, the feature selection is being performed on the frequency axis.

In WS context, a subset of WS coefficients ‘frequency-wised’ is being selected as illustrated in Figure 4.3, where each modulation represents a single bit gene. Hence each gene is a vector with a single frequency axis and the entire temporal axis (1 x Time). Each bit has a value of 0

or 1, and 0 means not selected while 1 means selected. Only the selected genes will be fed to the Two-stage convolution neural network (TSCNN) and (TSCNN-2).

In the signal processing context, it is similar to performing bandpass filters at a very fine level. Usually, bandpass filters are used to cut-off signal based on a range (e.g., 0-30Hz, 10kHz – 20kHz, or more relatable in log mel-spectrogram context, 118 bins to 128 bins, assuming it has 128 bins) instead on the specified frequency in signal context or specified bin in log mel-spectrogram as exhibited in Figure 4.3. As for WS, GA is used to pinpoint the exact wavelet transformed coefficients based on a data-driven approach. In a way GA present a next level of bandpass filtering instead of depending on experts' domain knowledge. Even with expert knowledge, one might filter out important information. In the case of ASC, it is even more challenging to decide which frequency spectrums are vital as it consists of multiple sound sources and events with various frequency profiles.

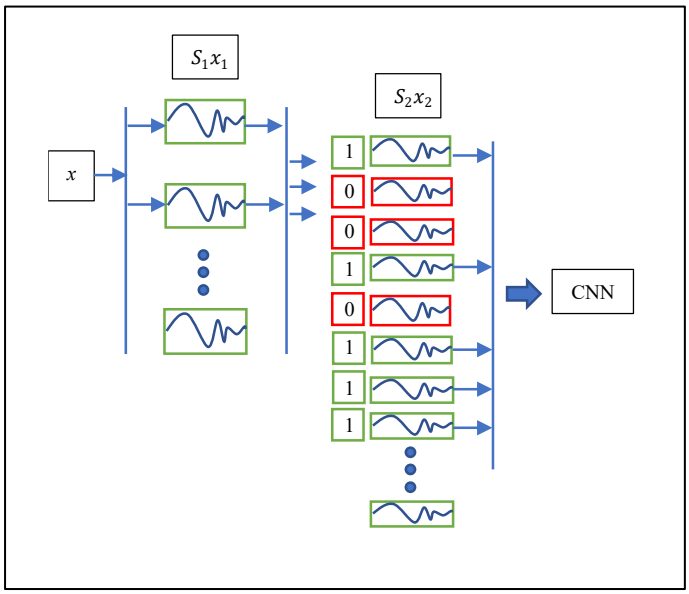



**Figure 4.3.** GA representation of WS second-order coefficients. GA visualized second-order coefficients as a binary string chromosome.

As aforementioned, the data-driven approach to select the optimal WS is the combination of GAFS with CNN as the fitness function based on the wrapper approach. As explained in Chapter 3, CNN is one of the current leading neural networks that can learn complex structures such as time-frequency representation and images, but not limited to 2-Dimension images (Alzubaidi et al., 2021, Khan et al., 2020). Hence, the collective combination of WS for feature extraction, GAFS wrapper approach or 'fine' bandpass filtering, and CNN as the classifier is depicted in Figure 4.4, and the entire process flow is illustrated in Figure 4.5. The proposed system is termed 'GATSCNN' for the first configuration and 'GATSCNN-2' for the second configuration.

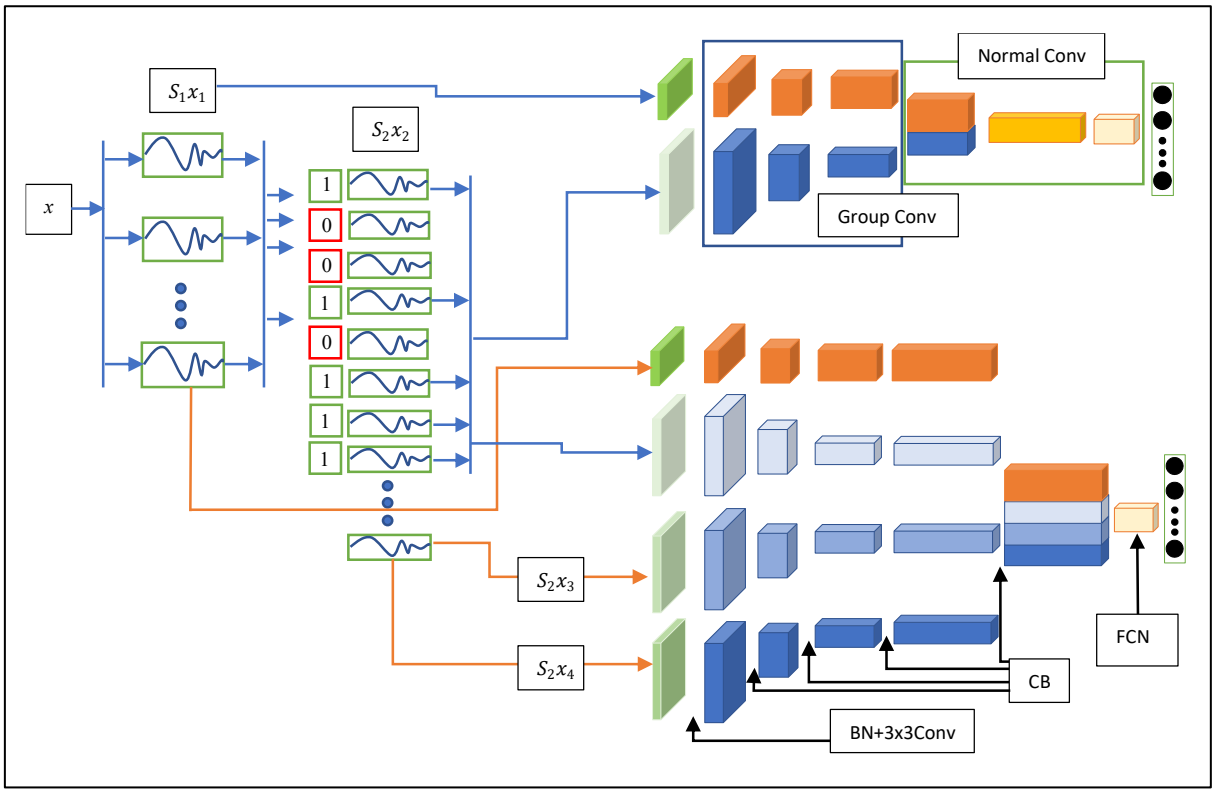



**Figure 4.4.** Integrated system of WS, GAFS with TSCNN and TSCNN-2.

Based on the illustration in Figure 4.5, the entire system starts with the initialization of the Population described in Table 4.1. Next, the fitness of all the chromosomes is being evaluated, and depending on the configuration, different CNN models will be selected. Subsequently, a checking mechanism is used to determine whether the system has already selected a set of parents. This argument caters to the first iteration, where no parents have chosen yet. Hence, the rest of the iteration will always flow to combining parents and offspring. Then, based on the classification accuracy, only the top-performing chromosomes is kept, and this batch of chromosomes will be the next parents. This leads to the next step of creating the offspring based on Table 4.1, step 2. Prior to continuing to the next iteration, a verification process is defined to check whether the stopping criterion is being satisfied. In this case, the stopping criteria depend on the total number of generations. If it is not met, another round of evaluation, selection of new parents, and breeding offspring will start. Upon reaching the stopping goal, the best optimal feature subset based on classification accuracy is being selected for training and evaluation with Mix Up. Finally, the classification result is presented in Section 5.6.

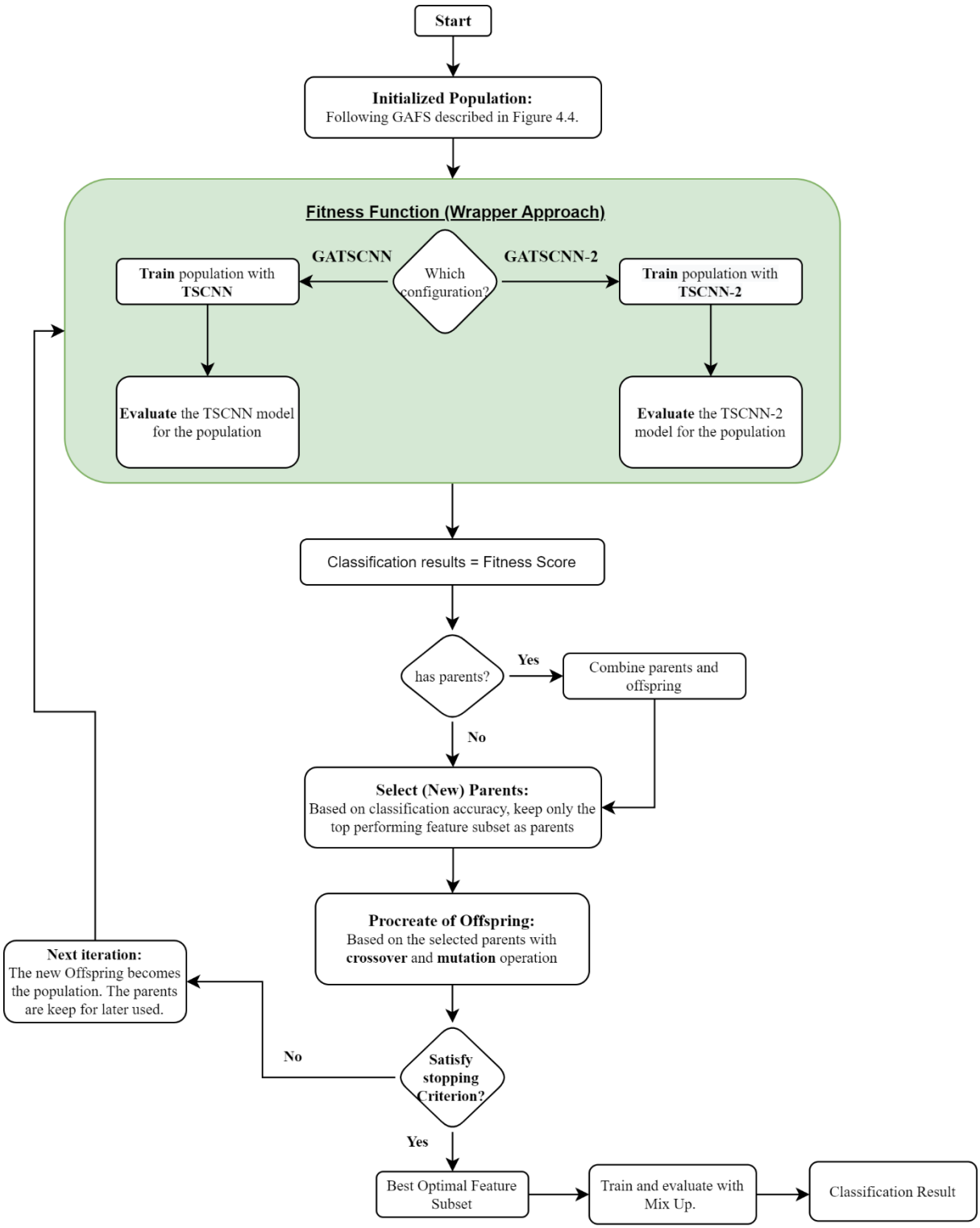



**Figure 4.5.** Process Flow of GATSCNN and GATSCNN-2 implementation.

The first configuration is the system design of GAFS for WS and IWS, which is rather straightforward as the inputs consist of single first-order coefficients and single second-order coefficients. Furthermore, based on evaluation of WS and IWS discussed in Section 5.5, the first order is fixated with a timescale of 46ms, while the second-order has a timescale of either *(92, 185, 371)ms* per GATSCNN run. Notably, during the runs for GATSCNN, both first and second-order coefficients undergo GAFS. However, during preliminary evaluation, it is

discovered that the first-order coefficients do not require feature selection. Hence, in the second configuration, GATSCNN-2, GAFS is only applied on the second-order(s). In GATSCNN-2, we proposed to resolve the problem of concatenating using GAFS. The result is that the input representation is a concatenated multi-timescale of WS(s) consisting of a 46ms first-order, 92ms second-order, 185ms second-order, and 371ms second-order, with only the second-order(s) undergoing GAFS. Based on the workflow described in Figure 4.5, the hyper-parameters for GA is listed in Table 4.2, where 8 parents or 8 fittest chromosomes is selected for the reproduction process. The parents will produce 12 offspring and 2 offspring have a mutation rate of 0.35 to alleviate the chances of reaching local optimal while preventing the network from the inability to converge. The number of generations is set at 10 due to limited resources and based on preliminary experiment with the no. of generations set at 20, the fitness score starts to plateau after the 10th run. Hence, 200 models are being trained for each GA implementation.

GA PARAMETERS

| PARAMTERS | *VALUE* |
|---|---|
| POPULATION SIZE | 20 |
| No. of PARENTS | 8 |
| CROSSOVER RATE | 0.5 |
| MUTATION RATE | 0.005 |
| MUTATION RATE for LAST 2 OFFSPRING | 0.35 |
| No. of GENERATIONS | 10 |

**Table 4.2.** GA Parameters. The GA parameters are used for both GATSCNN and GATSCNN-2 implementation. The stopping criterion here is by limiting the no. of generations to 10.

Setting the 'right' hyper-parameters for GA is crucial to achieve global optimal (Sloss and Gustafson, 2020, Yang and Honavar, 1998, Oreski and Oreski, 2014, Jung and Zscheischler, 2013, XUE et al., 2013, Siedlecki and Sklansky, 1989, Saeys et al., 2007, Xue et al., 2016, Goldberg, 2000, Goldberg, 1988, Duzdevich and Darwin, 2014). The first setting is to determine the population and the number of parents. The population initialization strategy is relatively straightforward, and the decision is based on observation from the preliminary experimentations and the limitation of hardware. Deriving from an earlier experimentation, it is possible to work with a population size of 20 and 8 parents. Based on the sentiment of (Uysal, 2018, Oreski and Oreski, 2014, XUE et al., 2013, Saeys et al., 2007, Xue et al., 2016, Sloss and Gustafson, 2020, Yang and Honavar, 1998, Jung and Zscheischler, 2013, Siedlecki and Sklansky, 1993, Hong and Cho, 2006, Zhang et al., 2018b, Ansari et al., 2021, Alexandre et al., 2007, Ibrahim et al., 2020, Nguyen et al., 2021), crossover rate is set as 0.5. This means that the offspring will consists of a mix of 50% of each parent.

A mutation operation can be described as changing the binary bit representation from 0 to 1, which means that the selected features inherited from parents will be flipped. For the mutation rate, (Siedlecki and Sklansky, 1989; Yang and Honavar, 1998; Jung and Zscheischler, 2013) cautioned that this has to be carefully selected. The higher the rate, the more indifferent the offspring are to their parents. Hence, if a high mutation rate is being applied for all the offspring, the offspring are more likely to lose important genes from their parents that made them fit in the first place. In other words, there lies a possibility of nullifying the crossover effect, where offspring are created based on the best-performing parents, thus, crippling the heuristic nature of GA. Hence, this thesis proposed using a high mutation rate of 0.35 only for the last two offspring to enhance the global search effect, based on the study of (Siedlecki and Sklansky, 1989; Yang and Honavar, 1998; Jung and Zscheischler, 2013). A mutation rate of 0.005 is applied for the rest of the offspring to provide stability to the GA process.

Lastly, the number of generations is set at 10 due to the resource limitation of evaluating over a combination of 1000 TSCNN and TSCNN-2 models. In addition, based on a separate experiment, with 20 iterations, the GA no longer demonstrates any gain after the 10th iteration or its 'plateau'.

While most of the parameters are being defined, there is another crucial decision for GAFS, which is to determine the size of the feature subset. Using the proposed GA parameters, an evaluation is conducted on the size of the feature subset by limiting the number of features selected for each independent run. 25%, 50%, 60%, and 70% is set as the size of the feature subset for each run, notably, the initial testing is only performed on GATSCNN configuration. The experimentation provided an insight on a trend towards an increase in the number of features selected for the case of 25%, 50% and 60%. Generally, at 70% of feature subset, the incremental of the number of features has plateau. Hence, for GATSCNN, a subset size consisting of 70% the total features is being selected.

While for the second configuration GATSCNN-2, based on the preliminary study of GATSCNN, on the hypothesis that combining multiple timescales will result in features overlapping. Hence, GATSCNN-2 is set with feature selection size to 50% for each second order.

### 4.4. Proposed Two-stage Mobile Shuffling Network

GAFS tackles the problem with large input representation and reducing the input size has shown to reduce time complexity, as proven in Section 5.6. Notably, GAFS or the feature selection process only affects the input representation that is being fed to the CNN and does not alter the

architecture of the CNN, such as the convolution block design. What it does is it reduce the number of convolution operations required to convolution around the feature maps. As one of the benefits of convolution neural network is weight sharing, additional convolution operations does not constitute to lesser weights. As such, another criterion for low complexity modelling is the size of the model. In terms of CNN, by counting the number of parameters of the network and based on the data type of the parameters, the memory size of the model can be calculated. Hence, the size of the model means the 'File size' measured in byte.

As mentioned in Chapter 1, the technique to downsize or build a low complexity CNN model is termed as 'Model Compression'. Notably, (Cheng et al., 2017) have provided a survey covering various strategies and techniques. The current strategy include parameter pruning (Molchanov et al., 2016), knowledge distillation (Hinton et al., 2015), quantization (Jacob et al., 2018), and convolution factorization (Sifre and Mallat, 2014).

Convolution factorization and quantization are the model compression chosen to 'down-size' the TSCNN model. Indeed, one could have utilized the current TSCNN model and performed parameter pruning and knowledge distillation. However, taking advantage of this problem, the thesis is able to demonstrate the flexibility of the two-stage CNN framework by integrating the framework with other possible convolution block.

Following the two-stage CNN framework proposed in Section 3.5, the proposed model T-MSNet, which is based on the compilation of MobileNet (Howard et al., 2019, Howard et al., 2017), ShuffleNet (Zhang et al., 2018c, Ma et al., 2018) and two-stage CNN is illustrated in Figure 4.6. The heavy adaptation of MobileNet is due to the depthwise separable convolution layer design (Sifre and Mallat, 2014). The computation cost analysis depicted in Table 4.3 shows that the depthwise separable convolution layer can deliver the lowest model cost. While shuffling modules presented by (Zhang et al., 2018c, Ma et al., 2018) account to zero model size cost. Lastly, following the two-stage CNN framework is unavoidable as we are dealing with WS with a large disparity in magnitude between the first and second-order.

Notably, the shuffling modules presented by (Ma et al., 2018, Fan et al., 2021, Zhang et al., 2018c) are not designed with time-frequency representation in mind. The shuffling modules proposed are channel shuffle (Zhang et al., 2018c), which shuffles channel-wise and spatial shuffle (Fan et al., 2021) that shuffles pixel-wise. Indeed, both shuffling modules have little interaction between frequency spectrum (low, mid, high). In addition, WS has a relatively large frequency dimension. Hence, to leverage this property, we proposed a new shuffling module

called 'sub-spectral shuffle' (SSS) that shuffles the feature maps 'frequency-wised'. Instinctively, we also studied temporal shuffling which shuffles the feature map ‘temporal-wised’ (TPS).

Subsequently, in Subsection 4.4.1, a brief discussion is made on various model compression techniques. In Subsection 4.4.2, the overall T-MSNet architecture and the possible configuration is being described. Then Subsection 4.4.3 entails a detailed elaboration on the mobileNet convolution block design. In addition, an analysis of the convolution operations complexity cost is presented. Lastly, in Subsection 4.4.4, the proposed ‘sub-spectral shuffling’ modules and the reason behind this design is being discussed.

### 4.4.1. Model Compression Technique

**Parameters pruning,** which deletes nodes deemed least important by the algorithms. The most popular approach is based on the magnitude of the nodes' weight or the gradients. The magnitude signifies the importance of the node to the model (Large magnitude means more important). This is a very popular technique in Section 5.7 for ASC.

**Knowledge distillation** tackles low complexity with a "teacher" and "student" network concept. In simplicity, the concept involves the teacher guiding the student, and corresponding to the neural network context, the teacher network calculated loss has an impact on the student final loss function and the level of influence is usually determined by a rate between 0 and 1. The student network is the desired low complexity model, and the teacher network is a larger network with the desired classification accuracy.

**Quantization** technique usually reduces computer memory size based on the data type. Commonly, the default data type for training deep networks is 32-bit floating points, which translate to each parameter in the network has memory upkeep of 4 bytes. Hence, by converting to a smaller memory data type (e.g., 16-bit floating points, 8-bit integer) will drastically reduce the overall memory size of deep networks. As exhibited in Section 5.7, this technique is prevalent as it can easily complement other model compression techniques. As mentioned in Chapter 3, the discriminative power of the network has a relation to the number of parameters when the CNN is constructed correctly. Hence, quantization is used to increase the number of parameters essential to building an effective CNN.

**Convolution factorization** is the concept of adapting matrix factorization to the convolution layer filters and is the focus of this thesis. Such design has been around in the image domain with state-of-the-art models, such as InceptionNet (Szegedy et al., 2015), MobileNet (Howard et al., 2017, Howard et al., 2019), and ShuffleNet (Ma et al., 2018, Zhang et al., 2018c). InceptionNet suggested that the convolution operations are a matrix multiplication between two

tensors and have the same properties as the matrix. Hence, it can be decomposed/factorized. The proposed ‘filter-wise’ factorization, which breakdown an M×N×C×D convolution layer (we can view this as the dimension of the convolution layer where M×N denotes the filter size, C and D denotes the number of filters for input and output, respectively), into two convolution layers, consisting of a M×1×C×D convolution layer followed by a 1×N×C×D convolution layer. This is similar to the 3×3 matrix = (3×1) × (1×3). Whereas the low complexity convolution layers used in both MobileNet (Howard et al., 2017, Howard et al., 2019) and shuffleNet (Ma et al., 2018, Zhang et al., 2018c) are mainly depthwise separable convolution layer (Sifre and Mallat, 2014), converting M×N×C×D into C+( M×N×1×D) follow by 1×1×C×D. In fact, 1×1×C×D is the development of (Lin et al., 2013) to improve cross-channel relations, initially termed as 'mlpconv'. In addition, shuffleNet integrates low complexity convolution layers with stochastic algorithm, which is channel shuffling, to improve model generalization at zero model size cost.

While the decomposition technique mentioned above works with the dimension of the convolution layers, SqueezeNet (Iandola et al., 2016) and ResNet (He et al., 2016a) leverage on mlpconv to construct their ‘fire module’ and ‘bottleneck module’, respectively. Their proposed ideas are more towards how to utilize mlpconv to expand or contract the number of output filters, corresponding to D of 1×1×C×D, without overly increasing the size of the model as the network gets deeper. In addition, this technique has also been used in CondenseNet (Huang et al., 2018) and ResNeXt (Xie et al., 2017). Moreover, (Xie et al., 2017, Huang et al., 2018, Iandola et al., 2016) adapted group convolution that divide the convolution layer operation channel-wise, corresponding to C and D of M×N×C×D. Let G denotes the factor to divide the convolution layer. The convolution layer will transform to G+(M×N×C/G×D/G).

Notably, the proposed model heavily adapts MobileNet (Howard et al., 2019, Howard et al., 2017) architecture design. However, the inclusion of shuffling modules (Ma et al., 2018; Zhang et al., 2018c) and TSCNN design is indifferent to their network.

It is challenging to distinguish which model compression method is superior as other factors will affect the model's accuracy, such as the CNN architecture design, applications, and requirements. Parameter pruning is the most popular approach as it can be directly applied to an existing network and when transfer learning is highly desired.

Knowledge distillation allows configuration of their networks (both teacher and student). However, they generally achieve less competitive performance as compared to pruning and convolutional factorization. The other broadly used technique is quantization, as it can be easily

implemented with other compression methods, as aforementioned. With prior knowledge of CNN architecture design, one can have the flexibility to construct the CNN using convolution factorization. Overall, all the model compression discussed above can be combined to maximize the model performance.

### 4.4.2. T-MSNet topology and configuration

The collective components of the MobileNet convolution block described in Subsection 4.4.3 and shuffling modules described in Subsection 4.4.4 present T-MSNet. In Section 3.5, it is explained that the two-stage approach is crucial in tackling the huge disparity in magnitude of the first and second-order coefficients. However, due to the computational complexity constraint setup by DCASE 2021, the first stage only consists of a single convolutional layer to separately process $S_1x$ and $S_2x$, with stride (1,1) and (2,1) respectively. Following the two-stage CNN architecture framework, downsampling is performed on $S_2x$, with a stride of 2, on the frequency axis as aforementioned. This is to reduce biases of the model on $S_2x$ caused by being a larger input representation (The larger the feature map the more convolution operations will be performed on the map, which will result in increased learning on the feature map). The second stage concatenates the features to enable cross-spectral learning, which is further enriched when SSS is applied. The base architecture of T-MSNet after the first convolution layer follows a 'lite' version of mobileNet. The model has 8 stacks of MobileNet convolution blocks (MBloc) and the channels configuration for each convolution block is stacked as follows: C = [16,24,24,32,32,64,64,96]. Next, the shuffling modules, which include a combination of SSS, TPS, and CS, or in singularity, are slotted after the MBloc before each incremental of C except for the first incrementation. Based on the findings of (Fan et al., 2021), there is a need to strategically position shuffling modules so that their maximum potential can unleash.

Hence, the location of shuffling modules is predetermined at C = [16,24,24,S1,32,32,S2,64, 64,S3,96,S4], where S1,S2,S3, & S4 are the indexed of their location as illustrated in Figure 4.6, and Table 4.5. While the location of the application of shuffling modules is set, it does not mean that shuffling will be applied. Inspired by (Fan et al., 2021), an analysis on the 'activation' of various combinations of shuffling modules at the predetermined location in Section 5.7 is being conducted. Finally, T-MSNet is completed with FCN described in Subsection 4.4.3. Apart from the location of the shuffling modules, the number of segments for SSS, TPS and CS is a set of hyperparameters. Based on the preliminary experiments, the configuration of the number of segments for SSS, TPS, and CS should be different. For ease of referencing, the

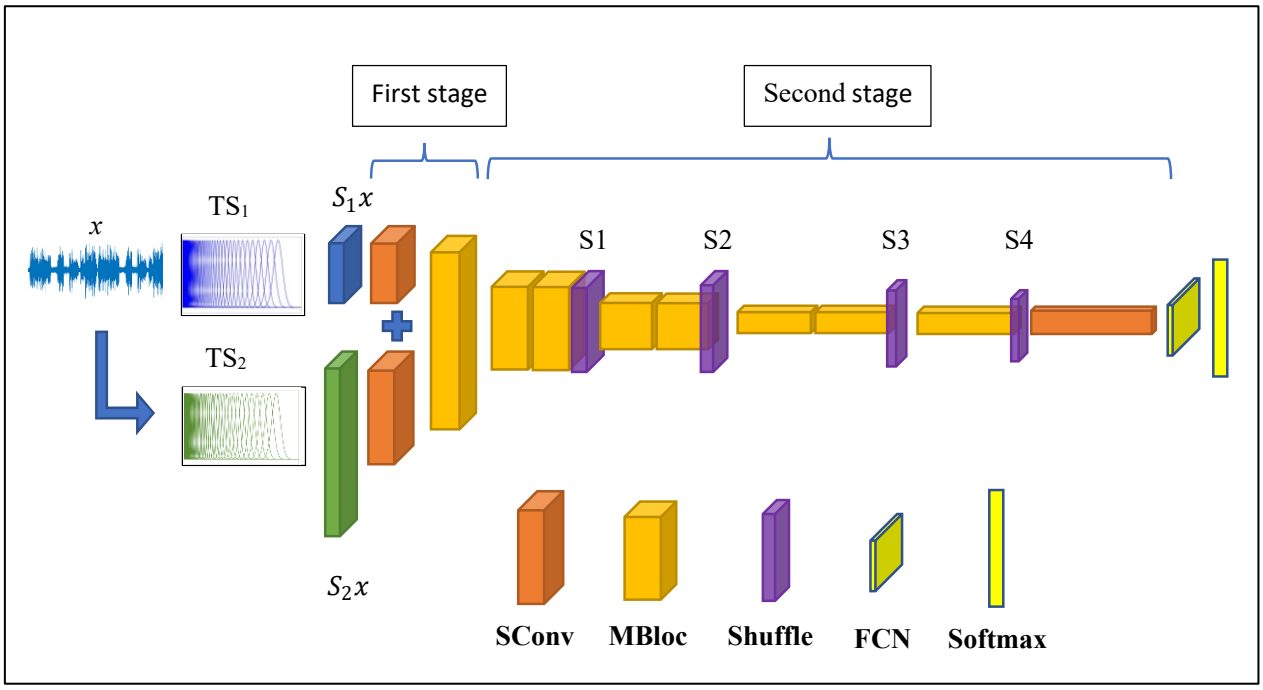



**Figure 4.6.** Overall, Two-stage Mobile Shuffle Network (T-MSNet) Architecture. SConv stands for Standard Convolution, MBloc stands for the convolution block of MobileNet. S1, S2, S3 and S4 are the predefined position of the shuffling modules. $TS_1$ and $TS_2$ are timescale used to construct WS first and second-order.

number of segments for SSS, TPS and CS are termed as *GBins*, *GTim* and *GChan*, respectively. The dimension of the time-frequency feature map is represented as ($F \times T \times$ C) where $F$ denotes the frequency axis, $T$ denotes the time axis, and $C$ denotes the number of channels. For the case of SSS, the number of segments used in S1 and S2 is more effective if it differs from S3 and S4.

The experimentation has shown a relationship that when *GBins* has the same size as $F$, it will result in poorer classification result. Having *GBins*=$F$ means that the entire spectral information is being distorted. Another relationship is that *GBins* are best set at 10 for both S1 and S2 and 5 for both S3 and S4, for the proposed configuration. As for TPS, we set *GTim*=$T$ as IWS has a small temporal dimension. Lastly, *GChan* is most effective at 8, which might be due to the number of channels of T-MSNet, which is relatively small to ensure the model size is within the given constraint. The configuration is tabulated in Table 4.3, where the parameters relate to the number of segments, and indexed relates to the predefined position of the shuffling modules. Lastly, the value is $T$ denotes the same size as the *Time* axis. Notably, the configuration in this Table will be used for the experimentation in Section 5.7. The overview of the system includes the pre-processing to IWS and T-MSNet is illustrated in Figure 5.6, and the T-MSNet architecture is described in Table 4.6.

CONFIGURATION OF THE NUMBER OF SEGMENTS FOR SSS, TPS, AND CS

| PARAMTERS | INDEXED | VALUE |
|---|---|---|
| *GBins* | S1,S2 | 10 |
| *GBins* | S3,S4 | 5 |
| *GTim* | S1,S2,S3,S4 | *T* |
| *GChan* | S1,S2,S3,S4 | 8 |

**Table 4.3.** Configuration of the number of segments for SSS, TPS, and CS. The Table shows the configuration of the number of segments for SSS, TPS, and CS.

### 4.4.3. Adapting from Mobile Network Convolution Block

The adaptation from MobileNet in this thesis is a combination of mobile network components that have carefully considered the extremely low computational cost. MobileNet convolution block (MBloc) is designed with convolution factorization in mind, and the adapted distinction can be summarized into three key components. The first component, which is the core convolution layer of MobileNet, uses depthwise separable convolution to reduce the complexity cost in two steps. The first step focuses on capturing spatial relationships using a single filter per channel concept, called depthwise convolution, depicted in Figure 4.7b. The second step employs a pointwise convolution as depthwise convolution does not capture a channel-wise relationship. To better understand how depthwise separable convolution help in the reduction in model complexity, a side-by-side comparison is illustrated in Figure 4.7 and a list of convolution layers and their computational analysis is compiled in Table 4.4.

For ease of referencing, a standard convolution layer is represented as a tensor of M×N×C×D where M×N represents the size of the filters, C is the number of filters, also called channels, and D is the number of output filters. The depthwise convolution layer is the first convolution layer of depthwise separable convolution design in Figure 4.7b. Instead of filter-wise factorization, they break down the convolution layer channel-wise, such that M×N×C×D becomes M×N×1×D and a condition of D=C. This considerably reduces model size as C is always larger than M and N, especially when the network goes deeper.

The output of the depthwise convolution layer results from overly focused feature-wised learning and lacks understanding of the cross-channel relationship. Hence, a 1×1×C×D is employed to cover this semantic. In a depthwise separable convolution setup, conv(1,1) is termed a pointwise convolution as it focuses on a single point of the feature map instead of a

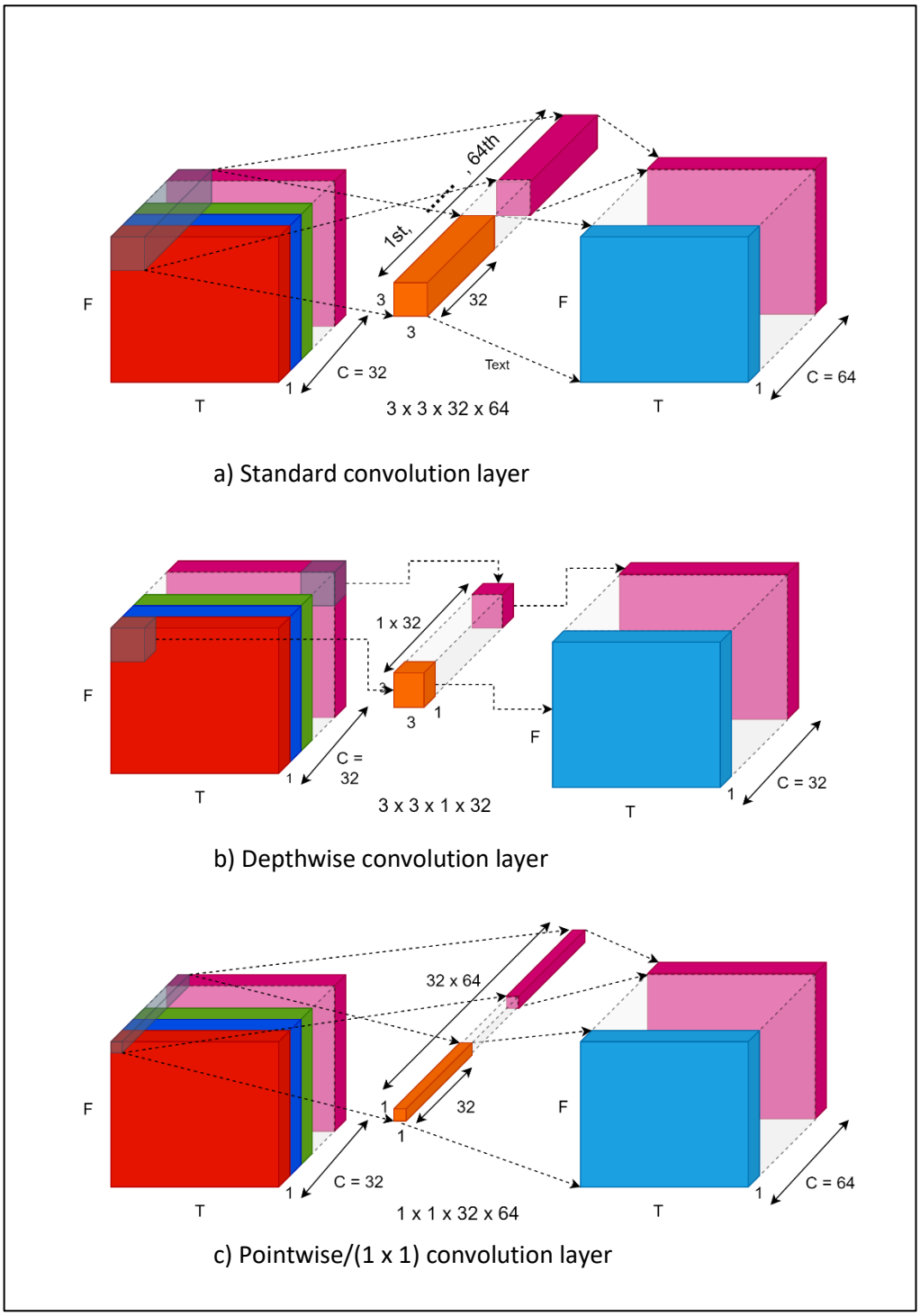



**Figure 4.7.** Comparison of standard convolution layer, depthwise convolution layer and pointwise convolution layer. (F x T x C) denotes the dimension of time-frequency representation with channels.

spatial relationship. The conv(1,1) was designed by (Lin et al., 2013), and they intend to enhance the cross-channel relationship. The design of Conv(1,1) is also depicted in Figure 4.7c.

Lastly, the comparison between standard convolution against depthwise convolution and pointwise convolution is depicted in Figure 4.7. The examples in Figure 4.7 are based on a feature map with a channel size of 32, undergoing convolution with a filter size of (3 x 3), and

the number of output channels is 64. Typically, Figure 4.7b is usually accompany with Figure 4.7c.

In addition to understanding the construct of depthwise separable convolution operations, it is crucial to understand the computational complexity of these convolution operations as they directly impact the model's size and the calculation of the number of parameters is reflected in Table 4.4. This Table presents the computational complexity of various convolution operations discussed in Section III, A, based on the number of parameters. The number of parameters translates to the size of the model. The examples used in Row 1 to 9 reflect a convolution operation on a feature map with 32 channels, and the number of output channels is also 32. While for Row 11 to 13, the output feature map size is similar to the proposed model. (Refer to Table 4.5). Based on the comparison analysis, depthwise separable convolution has shown to be the best candidate for low complexity modelling, especially when the limitation is set at 128Kb (Based on DCASE 2021 Task1a criteria).

The second component incorporates an inverted residual and linear bottleneck structure (Howard et al., 2017), which signified the MobileNet convolution block design (MBloc) as illustrated in Figure 4.8. The design follows an inverted residual and linear bottleneck structure. $x$ is a feature map, and the output after MBloc is $x'$. The first convolution layer of MBloc is called the 'expansion layer' and has a similar structure as the 1x1Conv = conv(1,1). A hyperparameter termed 'expand' determines the number of output channels, denoted as D. Batch Normalization (BN). Followed by an activation function "ReLU" is being used right after the expansion layer (1x1Conv) and depthwise convolution layer (3x3DWConv). No activation function is required after the pointwise convolution layer (1x1PWConv) as it shows to impede learning. If the stride of the Mbloc operation is equal to 2, there will be a residual connection between $x$ and $x'$.

The whole convolution block is defined by stacking expansion convolution, which is a conv(1,1), followed by depthwise separable convolution, then a projection layer, which is also a conv(1,1). As the name suggests, the expansion convolution is designed to increase the number of feature maps. It elevates the effectiveness of depthwise separable convolution. The expanded channels provide a higher-dimensional feature space, leading to more expressiveness of the non-linear spatial-wised learning delivered by the depthwise convolution layer. The pointwise convolution condenses this information channel-wised into a sparse representation, maintaining a compact model for the subsequent convolution operations (Howard et al., 2017, Howard et al., 2019). The projection layer reflects the actual number of output channel and act.

as both a consolidator and a dimensionality reduction layer, producing a sparse set of feature maps.

COMPUTATIONAL COMPLEXITY OF VARIOUS CONVOLUTION OPERATIONS

| No | Convolution layer | Filter Dim | Parameters |
|---|---|---|---|
| *1* | Standard | 3 x 3 x 32 x 32 | 9,216 |
| *2* | Filter-wise | 3 x 1 x 32 x 32 + 1 x 3 x 32 x 32 | 6,144 |
| *3* | Depthwise | 3 x 3 x 1 x 32 | 288 |
| *4* | 1x1Conv | 1 x 1 x 32 x 32 | 1,024 |
| *5* | Group(4) | 3 x 3 x 8 x 8 x 4 | 2,304 |
| *6* | Depthwise separable | (3)+(4) | 1,312 |
| 7 | Shuffle | *Permutation | 0 |
| | | Convolution Block | |
| *8* | VGGCB | 2 x (1) | 18,432 |
| *9* | 2 x MBloc | (4) = 1 x 1 x 32 x (32 x 6) (3) = 3 x 3 x 1 x 192 (4) = 1 x 1 x 32 x 32 | 2 x 8,896 = 17,792 |
| | | Classifier Layers | |
| *11* | Flatten+ Dense | Flatten(30,4,96)+ Dense(96) Dense(96) | 1.1(mil) |
| *12* | GAP+ Dense | GAP(30,4,96)+ Dense(96) Dense(96) | 19,594 |
| *13* | FCN | (4) = 1 x 1 x 96 x 96 (4) = 1 x 1 x 96 x 10 GAP(30,4,10) | 10,176 |
| | | Proposed Model | |
| *14* | *TSCNN* | | |
| *15* | *TSCNN-2* | | |
| *16* | *T-MSNet* | Appendix, Table I | 64,788 |

**Table 4.4.** Computational complexity of various convolution operations.

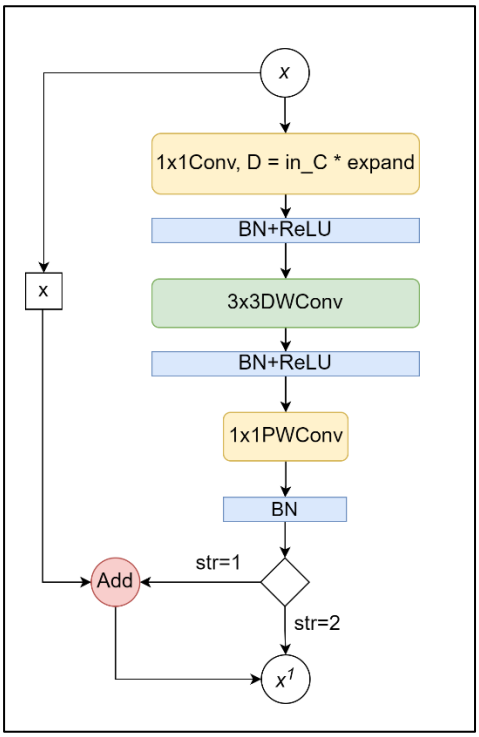



**Figure 4.8**. MobileNet Convolution Block (MBloc) design adapted from (Howard et al., 2017, Howard et al., 2019).

Lastly, to reduce the computational cost caused by classifying with a fully connected layer, (Shelhamer et al., 2017) proposed Fully-Convolution (FCN). Before FCN is developed, the CNN model typically ends with a fully-connected layer(s) or dense layer(s) before the softmax layer. For this dense layer(s) to be effective, it requires a sizable number of hidden nodes, resulting in extremely high computational costs (e.g., D2). Hence, (Shelhamer et al., 2017) redesigned this portion by eliminating all dense layer(s) and instead of connecting the final convolution layer with a conv(1,1) followed by a 1×1×C×D, where D, in this case, is the number of classes. Global average pooling (Lin et al., 2013) is applied before the softmax layer. The result is a significant reduction in computational cost, as shown in Table 4.4.

The classification layer used for the low complexity model is undisputedly FCN as seen in Table 4.4. The FCN has the best computational complexity compared to the other 2 classification layer designs described in Chapter 3, Section 3.3.2.

Apart from the three components, to provide greater flexibility, mobileNet offers two hyperparameters to throttle the complexity: alpha and expansion ratio. The alpha value controls the "width" of the T-MSNet architecture, affecting the number of channels of every Mbloc. On the other hand, the expansion ratio is a hyperparameter setting for each Mbloc, and it controls only the size of the expansion layer. Based on an initial empirical study, alpha is set to 0.5 and all the expansion ratios is set to be 6.

### 4.4.4. From Channel and Spatial Shuffle to Sub-spectral and Temporal Shuffle for Time-Frequency Representation

To further exploit IWS, sub-spectral shuffling (SSS) module inspired by channel shuffle (CS) designed by (Zhang et al., 2018c) and spatial shuffle (SS) by (Fan et al., 2021) is being proposed. In Chapter 3, Section 3.3.2, shuffling is categorized as a stochastic manipulation approach and this type of convolution block design tends to improve the network performance without increasing the number of model parameters. Hence, it is very applicable to designing a model with size restrictions. Shuffling channel-wise, pixel-wise, or sub-spectral-wise encourages the filters to encompass information of different features. Temporal shuffling (TPS), which is the shuffling along the time axis is also being investigated. The difference between CS, SS, SSS, and TPS is illustrated in Figure 4.9, where the feature maps represented in the figure have a dimension of (*F* x *T* x *C*) (time-frequency representation with channels).

Given an input representation with dimension size of ($F \times T \times C$), where ($F \times T$) represents the feature map which is a time-frequency representation. *F* is the frequency axis, and *T* being the time axis. *C* is the number of feature maps and usually termed as a channel. CS performed shuffling on-axis *C*, allowing cross-learning of feature maps. SS visualized feature map, ( $F \times T$ ), pixel-wise and randomly scrambled the individual pixels to destroy spatial memorization by the network. Thus, allowing the network to generalize more effectively. Although this makes sense for image classification, it has an adverse effect on acoustic scene classification. The poorer performance can be attributed to time-frequency representation, where a degree of temporal sequence and frequency locality are essential. Only SSS will be used for examples and explanations. In addition, the algorithm is applicable to TPS and CS.

Using SSS as an example, the feature maps are grouped into bins of feature maps with (*F/GBins* x *T* x *C*) dimensions. During SSS operations, all the bins will undergo a random process of repositioning while satisfying this condition $1 < GBins \leq F$, as illustrated in Figure 4.9. As when *GBins*=1, it means that no SSS is used, while *GBins* cannot be larger than the original input dimension *F*. The combination of Mbloc with shuffling modules is illustrated in Figure 4.10d.

In addition, Figure 4.10 also illustrates the difference between the proposed convolution block over shuffleNet and MobileNet. T-MSNet is constructed by adding of shuffling modules into mobileNetV2 convolution block design. The shuffling modules are accentuated to illustrate their position for different convolution block designs. While ShuffleNet and ShuffleNetV2 use channel shuffle (CS), we proposed the inclusion of various shuffling modules, being SSS, TPS, and CS. 'GConv' denotes group convolution. 'EConv' denotes the expansion layer explained in

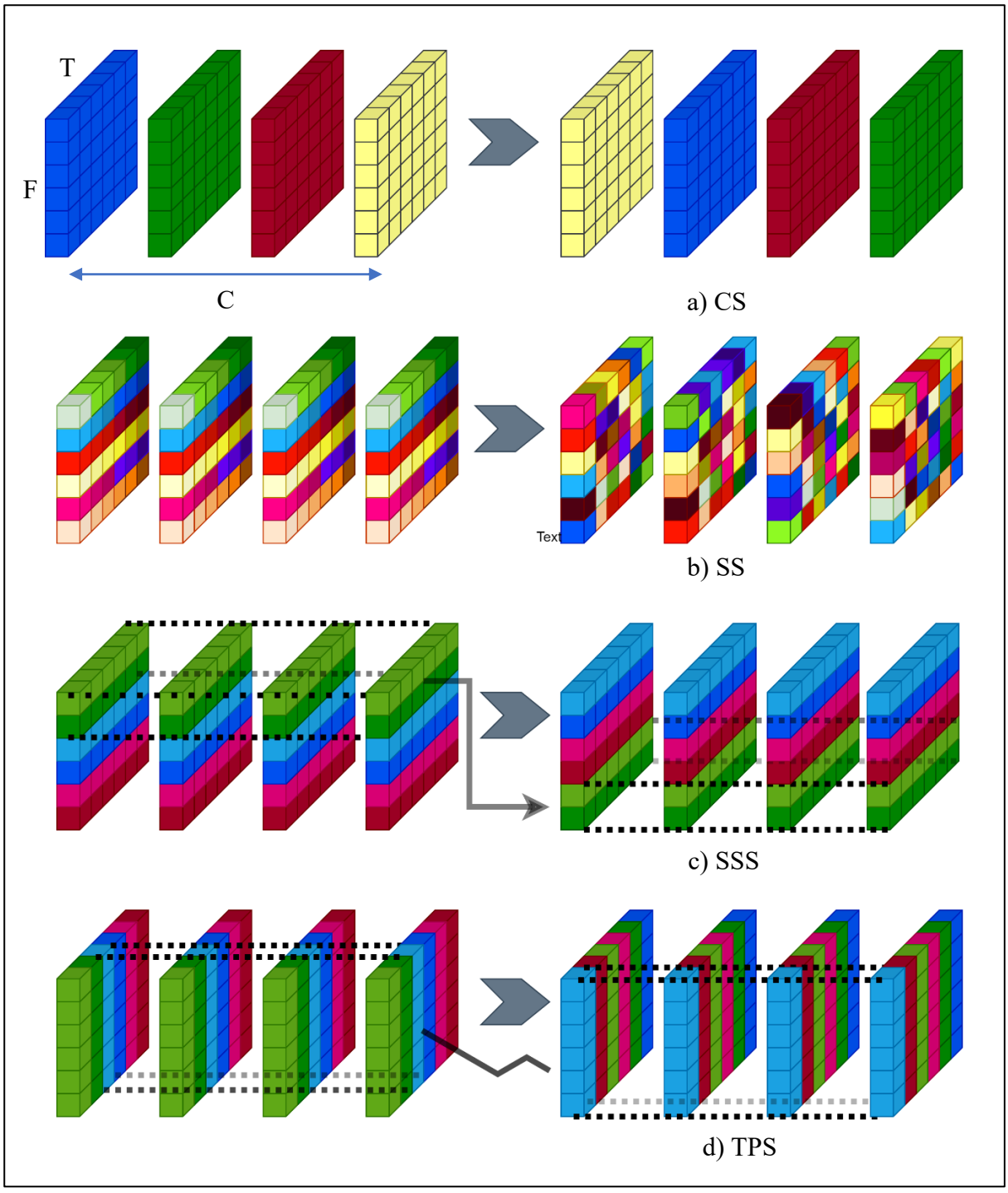



**Figure 4.9.** Variants of shuffle module. With (a) being channel shuffle (CS), (b) being spatial shuffle (SS), (c) being sub-spectral shuffling (SSS), and lastly, (d) being temporal shuffling (TPS).

Figure 4.8. 'DWConv' denotes Depthwise convolution, and 'str' denotes the stride value of the convolution layers. 'Channel Split' breaks down the feature maps into two groups.

Finally, we present the detailed T-MSNet architecture in Table 4.5, where the first convolution layer is a standard convolution operation. The rest of the Table are divided into Mbloc(s), shuffling modules of either TPS, SSS, CS or a combination of them, and the last layer is FCN. Mbloc is represented by 1x1Conv, Depthwise convolution (DWConv) and pointwise

convolution (PWConv). The BN and ReLU for Mblock are omitted from this Table as they are illustrated in Figure 4.8.

### 4.5. Hypothetical combination of GAFS with T-MSNet

While the setup of GAFS and T-MSNet is isolated, it is possible to combine both together. We can easily swap out the TSCNN with T-MSNet as both possess similar two-stage CNN framework. Just with this minor change, the entire workflow for GAFS becomes operatable, thus demonstrating the flexibility and robustness of the GAFS implementation. In addition, the proposed GAFS for WS, should not just be limited to WS. This approach can be extended to other time-frequency representations such as log mel-spectrogram or STFT.

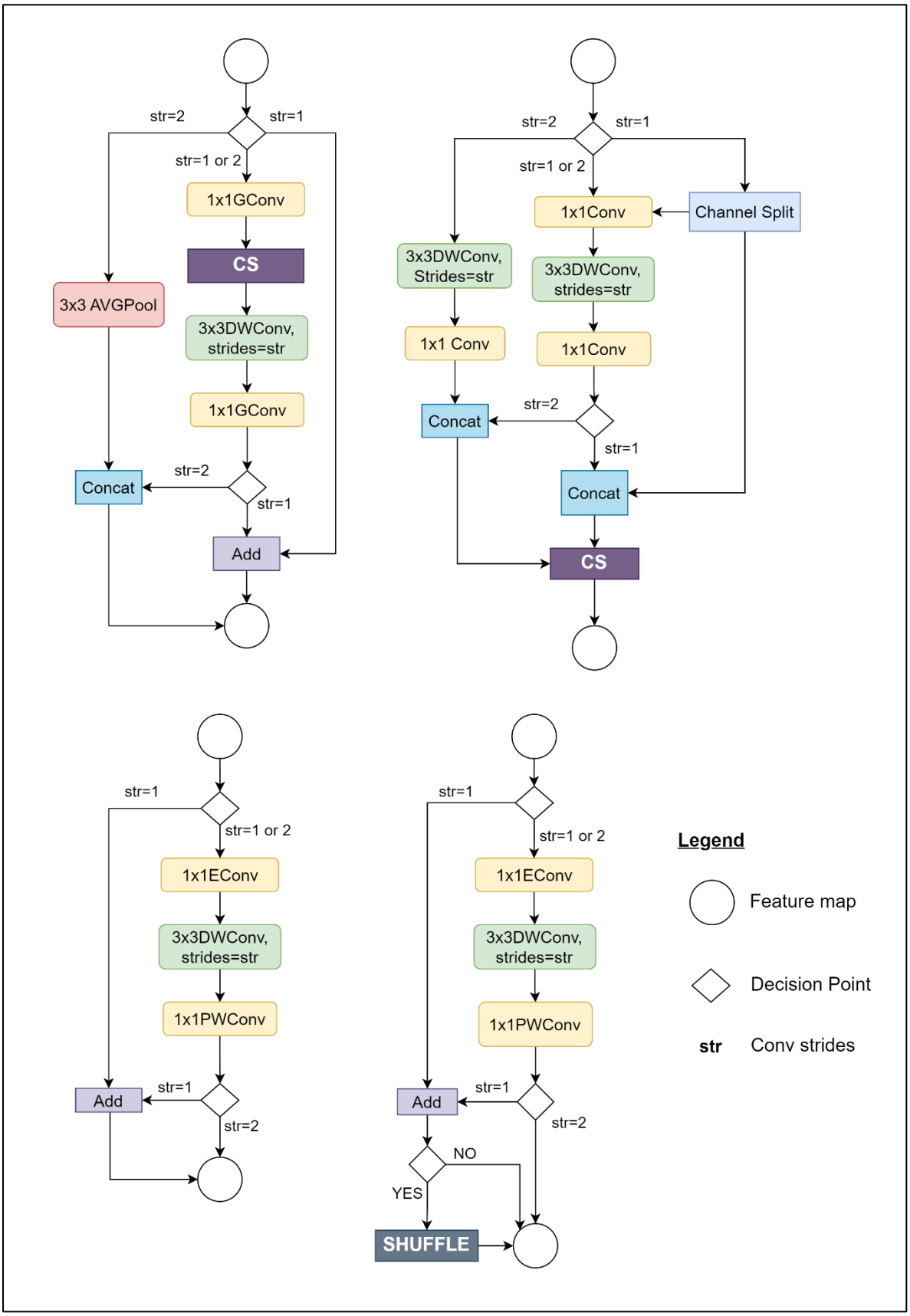



**Figure 4.10.** Comparison of convolutional blocks of ShuffleNet (Zhang et al., 2018c), ShufflNetV2 (Ma et al., 2018), mobileNet (Howard et al., 2019), and the proposed T-MSNet.

T-MSNET ARCHITECTURE DESIGN

| Ops | Stride | Filter Shape | Input Size |
|---|---|---|---|
| Conv | (1,1), (2,1) | 3 x 3 x 16 | 105 x 56 x 3, 724 x 56 x 3 |
| Concat+BN | | | 105 x 56 x 16, 362 x 56 x 16 |
| 1x1Conv | (1,1) | 1 x 1 x 16 | 467 x 56 x 16 |
| DWConv | (1,1) | 3 x 3 x 16 | 467 x 56 x 16 |
| PWConv | (1,1) | 1 x 1 x 8 | 467 x 56 x 16 |
| 1x1Conv | (1,1) | 1 x 1 x 48 | 467 x 56 x 8 |
| DWConv | (2,2) | 3 x 3 x 48 | 467 x 56 x 48 |
| PWConv | (1,1) | 1 x 1 x 16 | 234 x 28 x 48 |
| 1x1Conv | (1,1) | 1 x 1 x 96 | 234 x 28 x 16 |
| DWConv | (1,1) | 3 x 3 x 96 | 234 x 28 x 96 |
| PWConv | (1,1) | 1 x 1 x 16 | 234 x 28 x 96 |
| S1 | | | 234 x 28 x 16 |
| 1x1Conv | (1,1) | 1 x 1 x 96 | 234 x 28 x 16 |
| DWConv | (2,2) | 3 x 3 x 96 | 234 x 28 x 96 |
| PWConv | (1,1) | 1 x 1 x 16 | 117 x 14 x 96 |
| 1x1Conv | (1,1) | 1 x 1 x 96 | 117 x 14 x 16 |
| DWConv | (1,1) | 3 x 3 x 96 | 117 x 14 x 96 |
| PWConv | (1,1) | 1 x 1 x 16 | 117 x 14 x 96 |
| S2 | | | 117 x 14 x 16 |
| 1x1Conv | (1,1) | 1 x 1 x 96 | 117 x 14 x 16 |
| DWConv | (2,2) | 3 x 3 x 96 | 117 x 14 x 96 |
| PWConv | (1,1) | 1 x 1 x 32 | 59 x 7 x 96 |
| 1x1Conv | (1,1) | 1 x 1 x 192 | 59 x 7 x 32 |
| DWConv | (1,1) | 3 x 3 x 192 | 59 x 7 x 192 |
| PWConv | (1,1) | 1 x 1 x 32 | 59 x 7 x 192 |
| S3 | | | 59 x 7 x 32 |
| 1x1Conv | (1,1) | 1 x 1 x 192 | 59 x 7 x 32 |
| DWConv | (2,2) | 3 x 3 x 192 | 59 x 7 x 192 |
| PWConv | (1,1) | 1 x 1 x 48 | 30 x 4 x 192 |
| S4 | | | 30 x 4 x 48 |
| 1x1Conv+BN+ReLU | (1,1) | 1 x 1 x 96 | 30 x 4 x 48 |
| 1x1Conv | (1,1) | 1 x 1 x 10 | 30 x 4 x 96 |
| GAP | | | 30 x 4 x 10 |
| Softmax | | | 10 |

**Table 4.5.** T-MSNet architecture design.

## 4.6. Chapter Summary

This chapter topic revolve around tackling low complexity modelling such that the model can be deployed into a less computational powerful devices such as IoT and smartphone. Low complexity modelling can be broken down into two problems, being time complexity and size complexity. Time complexity is explained as the time taken for the model to predict, while size complexity deals with the size of the model in bytes.

One way to reduce the time complexity is by reducing the size of the input representation or dimensionality reduction. However, it is important to inherit or preserve the properties of wavelet scattering, hence, feature selection is preferred over feature construction. In addition, dealing with large set of features (over 500 features set) is rather challenging and this paper handled this problem by using genetic algorithm, which is has a meta-heuristic nature. Hence, in the first part of this chapter, an overview on genetic algorithm is delivered before drilling down into genetic algorithm for feature selection. Tapping onto the expressive power of convolution neural network and the ease of implementation, wrapper approach is selected to integrate WS, CNN, and GAFS together.

The second part of this chapter talks about resolving size complexity and the most popular technique to reduce the size of the CNN model is called model compression technique. Hence, in the second part, an overview is given to describe various model compression techniques before going deeper into convolution factorization. While this thesis selected convolution factorization and quantization as the choice for model compression, it is notable that all the model compression techniques can work together and compliment one another. In addition to model compression techniques, this paper also proposed the inclusion of shuffling modules which is a stochastic algorithm to improve the model ability to generalize at “zero-size” model cost. Particularly, this paper developed sub-spectral shuffling and temporal shuffling. Notably, the proposed sub-spectral shuffling work extremely well with wavelet scattering as compared to channel-wise shuffling and temporal shuffling.

# Chapter 5. Experiment and Results

## 5.1. Introduction

In this section, an evaluation is performed on all the proposed approaches over 3 ASC datasets described in Section 1.4. Notably, DCASE 2021 dataset is the primary dataset considered with all the proposed models. Specifically, ESC-50 and US8K is only evaluated in Section 5.5.2 to provide timescale analysis of WS and IWS, and when doing an ablation study of T-MSNet in Section 5.7.

In this section, the experiment discussions are adapted from the 3 papers:

1. X. Y. Kek, C. S. Chin, and Y. Li, “An Investigation on Multiscale Normalised Deep Scattering Spectrum with Deep Residual Network for Acoustic Scene Classification,” *22nd IEEE/ACIS International Fall Virtual Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPDI)*, pp. 29-36, 2021
    - for Section 5.2, 5.3, and 5.4
2. X. Y. Kek, C. S. Chin and Y. Li, "Multi-Timescale Wavelet Scattering with Genetic Algorithm Feature Selection for Acoustic Scene Classification," in *IEEE Access*, vol. 10, pp. 25987-26001, 2022
    - for Section 5.5.1, 5.6, and 5.8.1
3. X. Y. Kek, C. S. Chin, Y. Li, “An Intelligent Low-complexity Computing Interleaving Wavelet Scattering based Mobile Shuffling Network for Acoustic Scene Classification”, *Unpublished*. (Resubmitted to IEEE ACCESS)
    - for Section 5.5.2, 5.7 and 5.8.2

### 5.1.1. Hardware Specification and software packages

All the models are trained and tested on a Window Desktop with Intel i7-11700F Processor, 32 GB of RAM, and RTX 3090. Python is the programming language used to develop the models and implementation. Subsequently, Kymatio (Andreux et al., 2020) is used to perform scattering transform. Librosa (McFee et al., 2015) which is a python package for music and audio analysis. Lastly, the CNN network is built using Tensorflow 2.1 framework (Abadi et al., 2016).

### 5.1.2. Feature Extraction Implementation

In the entirety of the experimentation, the signal is only pre-processed into two time-frequency representations, being log mel-spectrogram and wavelet scattering. The two time-frequency is the point of interest in this thesis as log mel-spectrogram is the current SOTA features and wavelet scattering while not commonly used for ASC, provides a better representation. One major part of these studies is the setting of timescales. Hence, timescales, TS, is customized to fit the experiments described in the rest of the section, and the configuration for TS will be shared at the start of each Section. While TS is highly dependent on the experiment, the number of wavelets per octave for $Q_1$ $and$ $Q_2$ need to be predefined. Prior to defining $Q_1$ $and$ $Q_2$, it is notably that the mother wavelet used in this experimentation is 'Morlet' wavelet, as suggested by (Andén and Mallat, 2014).

For the value of $Q_1$, it was initially recommended by (Andén and Mallat, 2014) to set it as 8. Theoretically, they prove that it reflects a representation with a similar attribute as mel-scale spectrogram, which is essential for environmental sound classification. While based on the empirical studies (Salamon and Bello, 2015, Li et al., 2019, Ren et al., 2018, Zhang et al., 2019), where they exhaustively evaluated a series of $Q_1$, has demonstrated that $Q_1$ is in fact optimal when set to 9. This has led us to a preliminary study of choosing the value of $Q_1$ between 8 and 9, and based on that, $Q_1$ is set as 9 for both WS and IWS. As for $Q_2$, (Ren et al., 2018, Zhang et al., 2019, Andén and Mallat, 2014, Peddinti et al., 2014, Salamon and Bello, 2015, Li et al., 2019, Andén and Mallat, 2011) is in unison of setting it to 1. In addition, (Peddinti et al., 2014, Andén and Mallat, 2011) have found positive effect with the addition of the deltas and delta-deltas and concatenating them channel-wise, giving $C$=3.

For log mel-spectrogram, similar to WS setup, the timescales or window size, in this case, is not fixed. The predefined constant parameters are the number of mel-filter banks set to 128, 'hamming' window is used as the window function, and the hop length set to 50%. The hop length determines the number of overlaps between each window.

### 5.1.3. Hyper-parameters setup

**Learning Strategy:** Following (McDonnell and Gao, 2020, Chang et al., 2021) learning strategy, stochastic gradient descent (SGD) (Loshchilov and Hutter, 2017) with warm restart scheduled at a set of epoch indexes being (3, 7, 15, 31, 63, 126, 254) and learning rate covering from 0.1 (max) to 0.00001 (min) is adapted. Hence, the learning rate will gradually decrease at each step based on half a cosine curve called 'cosine annealing'. This provides adequate learning with a high learning rate to quickly approach local minimum, then a gradual decrease in learning rate to narrow into the local minimum. Based on the indexes above, the learning rate is being restarted to the maximum learning rate to ensure the model does not get stuck at the local minimum. (Loshchilov and Hutter, 2017) further proposed restarting the learning rate with shorter intervals at the beginning of the training phase to accelerate the learning process.

**Weight Initialization and Regularization:** 'He normalization' (He et al., 2015) is being used for weights initialization, and subsequently, L2 regularization is being applied.

**Batch size and Epoch:** The setup for batch size and epoch differs between GAFS runs and usual model training. For usual model training, a mini-batch size of 32 with 254 epochs is chosen. An epoch in this context is defined as a single cycle of learning the entire training data. As the learning strategy is SGD with warm restart, a save point is configured which only keeps the weights of the best model based on the logloss or classification accuracy. Hence, the models will be trained with a total of 254 epochs, but only the weights saved will be picked.

To speed up the process for GAFS runs, an investigation on the possibility of reducing the total number of epochs is conducted. Based on the observation, the best result can be achieved by the 62$^{nd}$ Epoch. The reason for this phenomenon can be described in two parts. The first part is the effective learning strategy of SGD with a warm restart. The second reason is that we are training without Mixup, which might have led to overfitting caused by a longer learning duration. Hence, for the GAFS run and in the essence of speed, we set a mini-batch size of 64 and total epochs of 63.

**Cross-Validation Setup:** For ESC-50, following the cross-validation setup proposed by (Piczak, 2015), the dataset is split into 5 predefined folds for cross-validation and ease of benchmarking. In addition, for each fold, we performed 3 additional training, which bring us to a total of 15 trainings and testing to evaluate a single model setup. Next, the average of all the 15 trainings tests classification results and logloss is tabulated as the metrics for comparison. For US8K, (Salamon et al., 2014) also provided a predefined 10 folds cross-validation setup. Instinctively, this thesis just follows the predefined setup without additional training for this

dataset. Lastly, for DCASE 2021, the organizers (DCASE) recommended a single fold with a predefined 70/30 (training/testing) split. As the data size for this dataset is a lot larger, a cross-validation setup is unnecessary. The model will be trained and evaluated at least 5 times over the same fold. The final score is calculated based on the average of the 5 training results.

**Data Augmentation:** Based on analyzing DCASE (Heittola et al., 2020, Mesaros et al., 2018) and suggestions from (McDonnell and Gao, 2020, Chang et al., 2021), this thesis adopted the use of Mixup (Zhang et al., 2018a) for data augmentation with an alpha of either 0.2 or 0.4 for evaluation on most of the models. For some models, the addition of Mixup has shown to impede the model's ability to learn in each section. Hence, for the case of GAFS run, which we deemed it unnecessary as the objective of GAFS is to find the optimal feature subset. Lastly, the value for the alpha is selected based on an exhaustive search where an evaluation is conducted on the alpha ranging from 0.1 to 1 for each model.

**Performance Metric:** In this thesis, two metrics is used to evaluate the models. The reason for the choice of metric is the ease of benchmarking with other models. The first metric is the classical way of measuring the performance of a model, which is based on classification accuracy. The next metric is logloss. Logloss is the measure of the confidence level of the model when making the prediction. (e.g., high confidence means that the logits on the 'true' label are nearing 1, and it has a logloss value approaching 0. Low confidence means that the logits on the 'true' label are far from 1, for example, 0.5, this will result in a higher logloss value)

### 5.2. Ablation study of the two-stage CNN Architecture Framework

The first investigation is the proposal of the two-stage CNN architecture framework for wavelet scattering. Inspired by (McDonnell and Gao, 2020, Chang et al., 2021), who use a TSCNN on log mel-spectrogram, this thesis emulates the CNN architecture design and applied it on WS and the rationale behind it was the challenge of effectively combining first and second-order coefficients described in Chapter 2. While (McDonnell and Gao, 2020, Chang et al., 2021) discussion is fixated on the effectiveness of binning/grouping process between low and high-frequency spectrum in convolution operations. A different perspective is being provided and is termed as two-stage CNN, where the first stage focus on specialized learning and the second stage focus on centralized learning (Refer to Chapter 3, Section 3.5).

With this new perspective, a study is setup to analyze the proportion of specialized and centralized learning illustrated in Figure 5.1. For Figure 5.1, the number index 1,2,3 & 4 correspond to the index number of the convolution block described in Table 3.1, also termed as 'stack'. The last stack is omitted. FCN is the classification layer. The tabulated result is

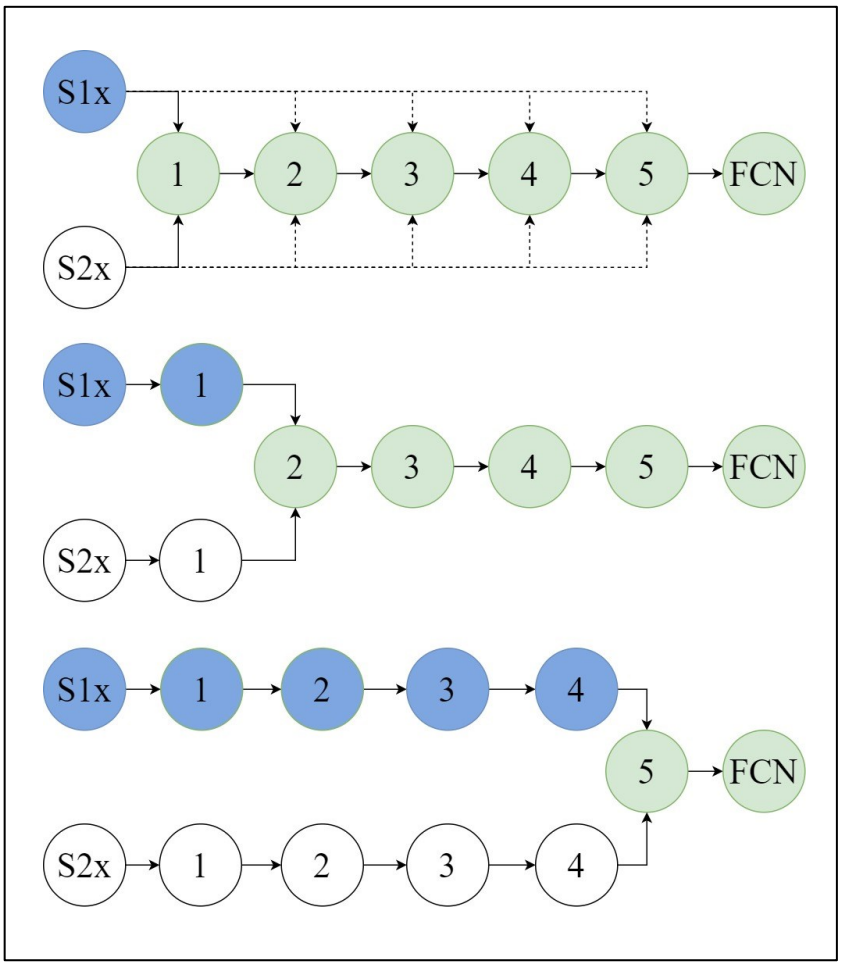



**Figure 5.1.** TSCNN setup for ablation study on the proportion of specialized and centralized learning. S1x and S2x denotes first and second-order of WS.

presented in Table 5.1 which shows the classification accuracy result of TSCNN based on DCASE 2021 dataset based on TS=92ms, and LMS is abbreviation of log mel-spectrogram with TS=46ms. Based on the result, the best setup for TSCNN differs between log mel-spectrogram and WS. It seems that different input representation will affect the proportion. In addition, the inclusion of T-MSNet has also demonstrated that the differences in CNN architecture will also affect the proportion of specialized and centralized learning. For the case of T-MSNet which is a combination of mobile convolution block and shuffling modules, an improvement in classification can be noticed right stack 1. However, this is not the case for TSCNN for both features, log mel-spectrogram and wavelet scattering. Combining both observation, both the input representation and the CNN architecture has shown to have an impact on the proportion of specialized and centralized learning. Hence, the suggestion to perform an ablation study on the proportion of specialized and centralized learning made sense.

While comparing a standard approach, the standard approach just means that there is no splitting between the first and second-order before feeding it to CNN, against a two-stage

approach, which has shown to improve the network by a slight margin. Based on this observation, two possibilities can be inferred. The first is that a deep CNN model can learn

CLASSIFICATION RESULT OF TSCNN WITH VARYING RATIO OF SPECIALIZED AND CENTRALIZED LEARNING

| No | Stack | LMS | WS | T-MSNet |
|---|---|---|---|---|
| *1* | Standard ResNet | 68.37% | 69.75% | 66.80% |
| *2* | Stack 1 | 68.00% | 69.65% | 67.10% |
| *3* | Stack 2 | 68.58% | 70.24% | - |
| *4* | Stack 3 | 68.78% | **70.36%** | - |
| *5* | Stack 4 | 69.40% | 69.29% | - |

**Table 5.1.** Classification result of TSCNN with varying ratio of specialized and centralized learning.

complex input representations such as the combined first and second-order coefficients with a huge disparity in magnitude. The next is that the inclusion of BN which performs a 'learning' normalization on the feature map during training, has replaced the need for a two-stage framework, which is further backed by the study on normalization in Section 5.2.

In summary, even though the improvement based on classification accuracy between standard and two-stage approaches is not very significant, it still does elevate the problem with the disparity of magnitude between first and second-order. The best configuration is stack 3 where there strikes a balance between specialized and centralized learning. Hence, a subsequent experiment for TSCNN is setup with this configuration unless otherwise stated differently.

### 5.3. The importance of Normalization

Other than employing a two-stage CNN framework to ease the learning process of first and second-order, the inclusion of input normalization demonstrated to be even more important than having a two-stage CNN design. In addition to the study on the importance of normalization, various normalization strategies have also being analysed and stated in Section 3.2.8. The result is tabulated in Table 5.2. The Table shows the classification accuracy result of TSCNN based on DCASE 2021 dataset based on TS=92ms. 'Renorm' is the shortform of renormalization, a technique introduced by (Andén and Mallat, 2014) to tackle the magnitude disparity problem. To reiterate, BN stands for Batch Normalization, LN stands for Layer Normalization, IN stands for Instance Normalization and SSN stands for Sub-Spectral Normalization.

Based on the observation, it undisputedly shows the need to perform any sort of normalization on WS, which is in accordance with (Ioffe and Szegedy, 2015), who stress the importance of having equally distributed coefficients such that it will improve the network ability to generalize more efficiently. Furthermore, the addition of input norm renormalization seems needless with

this setup. Hence, BN is a much more effective and robust option than renormalization designed just for WS.

CLASSIFICATION RESULT OF DIFFERENT NORMALIZATION TECHNIQUE

| No | Input Norm | ACC | Renorm |
|---|---|---|---|
| *1* | None | 67.31% | 68.01% |
| *2* | BN | **70.36%** | 70.35% |
| *3* | LN | 69.06% | 68.16% |
| *4* | IN | 68.19% | 68.08% |
| *5* | SSN | 70.19% | 70.28% |

**Table.5.2.** Classification result of different normalization technique.

While this investigation of various normalization techniques has shown that BN is still the best option, with the next runner up as SSN, there is a need to understand this phenomena. Base on how CNN process each mini-batch of data, the dimension of an output of any convolution layers will have the size of ($B \times F \times T \times C$), where $B$ is the number of data(e.g., 32 acoustic scene recordings in WS representation), ($F \times T$) relates to the time-frequency representation, and $C$ denotes the number of channels which accumulate as the network gets deeper. Indeed, their succession can be attributed to how each normalization works. The worst performing normalization is IN. IN is in fact a normalization based on normalizing each ($F \times T$), hence, it is rather similar to renormalization. Further supporting this claim is the relatively close classification score. If $C$ is included in the normalization process such that normalization is based on ($F \times T \times C$), it similar to LN. The channels, $C$, are the output of the feature maps based on the filters. Hence, in simplicity, the normalization is performed on many versions of a single WS. This has shown to be more effective than just normalizing a single data version. However, normalization based on various data has been demonstrated to be far superior in this case, where ($B \times F \times T$) is normalized together. The variety of data-based normalization is none other than BN and SSN. It makes total sense as the different acoustic scenes can exhibit very different levels of energy. In addition, this applies even to the similar acoustic scene, such as transient events. Hence, learning to get the correct scale to normalize this dataset will help with the model's ability to discriminate.

Furthermore, DCASE 2021 dataset are recorded with different devices, and other devices have different frequency response. Hence, how one device perceives sound can be different from another device. Normalization can equalize the data distribution, showing why BN and SSN work much better.

Another observation is that combining renormalization with SSN will improve the model performance. While this paper did not further study various combinations of normalization

techniques, the concept of combining normalization technique is not new. Such as combining batch and instance normalization by (Choi et al., 2021).

### 5.4. Timescale analysis between log mel-spectrogram and WS

The limitation of log mel-spectrogram (LMS) is that it is no longer stable to deformation when the timescale is larger than (25ms). Hence, log mel-spectrogram will perform worst given a large timescale. An investigation is established to see whether this is the case for ASC and whether WS can take advantage of having a larger timescale. In this experimentation, TS = [23,46,92,185,371,743,1480]ms and the result is presented in Figure 5.2. While it is noted that in the discussion of (Andén and Mallat, 2014, Mallat, 2012), Log mel-spectrogram is always emphasised to be configured best under 25ms, this thesis uses 23ms. This is due to the configuration of WS where the wavelet transform is built based on octave or $2^n$ and the sampling rate of the datasets.

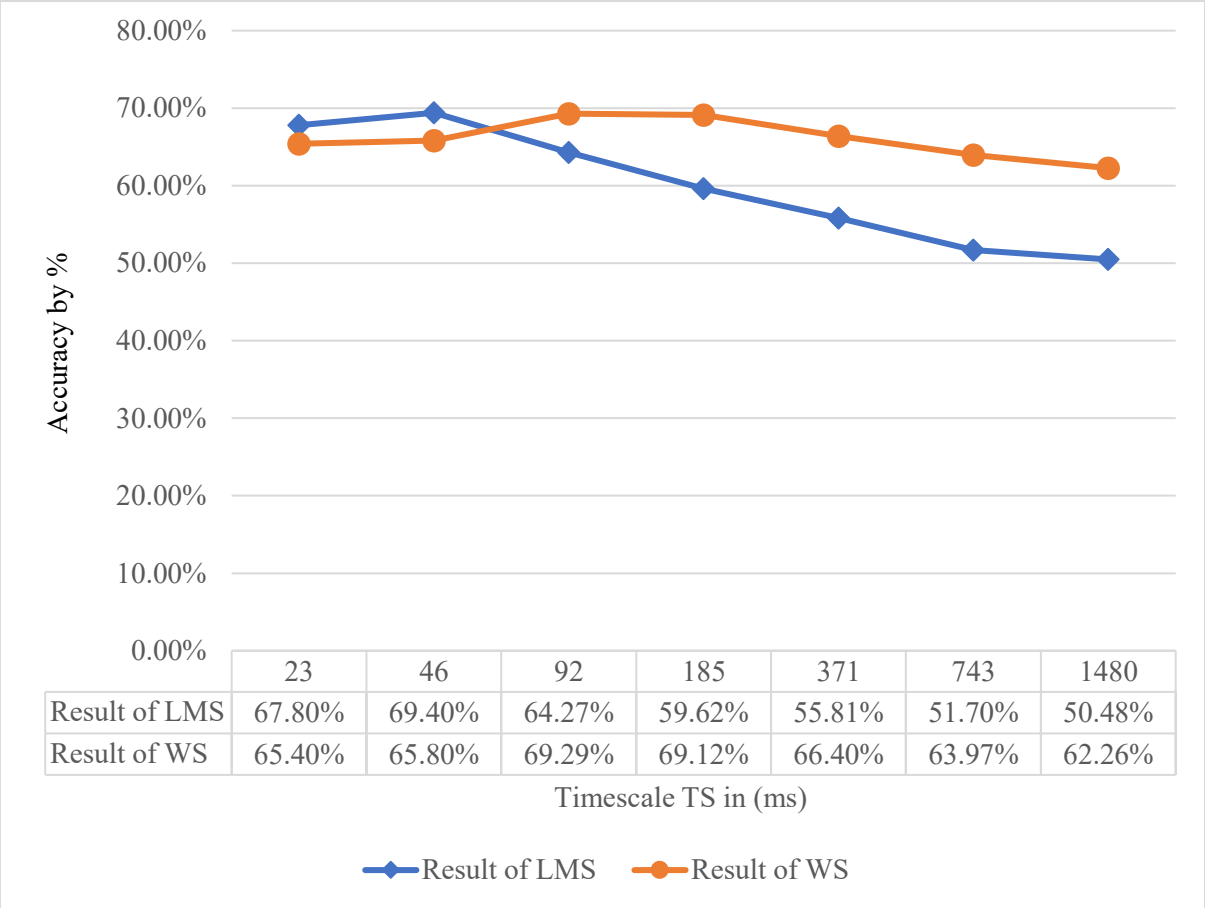
It is observed that LMS performs well with a smaller time scale of between 23ms and 46ms, while DSS strives with a larger time scale between 92ms and 185ms. This reaffirmed (Andén and Mallat, 2014, Mallat, 2012) theories that LMS is optimal given small window size, though for the case of ASC, it can be deduced that the TS of LMS could be stretched to at least 46ms instead of being conservative and capping it at 25ms. This study correspond to the competitors in DCASE ASC competition (Heittola et al., 2020, Gao and McDonnell, 2020, McDonnell and Gao, 2020, Gharib et al., 2018, Mesaros et al., 2018, DCASE) where they usually choose a timescale of >25ms for log mel-spectrogram.

This is also a reflection of the Heisenberg uncertainty principle. Different time scales are better at capturing different acoustic profiles, and the loss of information caused by mel-averaging does impede the classification result. Notably, LMS with the timescale of 46ms perform the best with accuracy of 69.4% while DSS come just in by 69.29% which is 0.11% poorer.

Perhaps for the case of ASC, LMS has a slight advantage over DSS due to the acoustic characteristic found in the scene, which is important. Another possibility is getting a suitable model for WS, as shown in Section 5.2.

The result of the timescale has also shown that a fixed time window using STFT works well with small timescale while a varying timescale using wavelet transform works well with a larger time scale. However, both features have poorer results when the time scale goes very large and time resolution diminishes greatly. This finding suggested that for both cases, a certain degree of temporal resolution is required for ASC. Hence, the CNN model might learn patterns of mid-

transient liked or quasi-stationary acoustic behaviors or events to help differentiate the scene instead of just the spectra component.




| | 23 | 46 | 92 | 185 | 371 | 743 | 1480 |
|---|---|---|---|---|---|---|---|
| Result of LMS | 67.80% | 69.40% | 64.27% | 59.62% | 55.81% | 51.70% | 50.48% |
| Result of WS | 65.40% | 65.80% | 69.29% | 69.12% | 66.40% | 63.97% | 62.26% |

**Figure 5.2.** Classification result of LMS and WS with different Timescale, TS. (For WS where TS=1480ms no delta and deltas and deltas-deltas is being calculated due to limited time dimension).

Based on the exhaustive search, the optimal range of TS for WS is between (46ms to 371ms). Hence, this range of TS will be used for multi-timescale WS.

## 5.5. Timescale analysis between WS and IWS

### 5.5.1 Timescale analysis between WS and IWS using TSCNN

This experimentation is conducted to exhaustively searched through a combination of the first and second-order of different timescales identified in Section 5.3. The classification results are tabulated in Table 5.3, where $TS_1$ & $TS_2$ denotes the timescale for the first and second-order, respectively. The first 4 rows are WS, and the rest of the tests are the combination of mixing first and second-order, IWS. The highlighted group of mixed first and second-orders have the best classification performance. However, we noticed that the classification result plateau at around ~70%. This finding solidifies the hypothesis that acoustic scene requires even more diversification of timescales, and just mixing two timescales is not enough.

Hence, in this experiment, we paired a set of first-order with timescales, $TS_1$ = {46,92,185,371}ms with a set of second-order with $TS_2$ having a similar value as $TS_1$. Based on the observation of Table 5.3, the best combination occurs when a small $TS_1$ (46ms) is combined with any $TS_2 > 46ms$. While combining the first and second-order the other way

CLASSIFICATION RESULT OF WS AND IWS WITH TSCNN

| No | $TS_1$ FOR $S_1x$ (ms) | $TS_2$ FOR $S_2x$ (ms) | ACCURACY |
|---|---|---|---|
| 1 | 46 | 46 | 65.80% |
| *2* | 92 | 92 | 70.36% |
| *3* | 185 | 185 | 70.15% |
| *4* | 371 | 371 | 69.14% |
| *5* | 46 | 92 | 70.72% |
| 6 | 46 | 185 | 70.42% |
| 7 | 46 | 371 | 70.51% |
| *8* | 92 | 46 | 69.14% |
| *9* | 92 | 185 | 70.13% |
| 10 | 92 | 371 | 69.53% |
| 11 | 185 | 46 | 68.50% |
| 12 | 185 | 92 | 70.03% |
| 13 | 185 | 371 | 69.52% |
| 14 | 371 | 46 | 68.35% |
| 15 | 371 | 92 | 68.42% |
| 16 | 371 | 185 | 69.33% |

**Table 5.3.** Classification result of WS and IWS with TSCNN. This is the classification result of an exhaustive search of the mixed first and second-order with TSCNN.

around, where $TS_1 > TS_2$, in most cases, it will result in poorer performance than not combining them (Table 5.3, rows 1, 2 & 3). This finding is consistent with the theory of energy dispersion from first to second-order as TS gets larger. In addition, the combination of timescales with the first order being the smallest TS, presents better result (Table 5.3 rows, 5, 6 & 7). While the combination of timescales with the first order having the largest TS, present poorer result (able 5.3 rows, 11, 14, 15 & 16). Hence, based on this observation, a general rule that $TS_1 < TS_2$ when constructing IWS is derived. Another observation is that mixing of the first or second-order of timescale of 46ms with larger timescale will result in a considerable boost of approximately ~4% in classification performance, as shown in Table 5.3, row 1, compared to Table 5.3, rows 5, 6, 7, 8, 11, 14. This phenomenon can be attributed to the acoustic profile of the acoustic scene, where important acoustic information is mainly captured by a larger timescale, TS>46ms. Contrary to the findings of (Andén and Mallat, 2014, Andén and Mallat, 2011), who state that the effective timescale of WS is around the range of ~300ms.

Perhaps this is highly dependent on the kind of acoustic profiles that will be tackled. Evidently so for ASC, as log mel-spectrogram with a timescale of 46ms has performed comparably as discussed in Section 5.4.

### 5.5.2 Timescale analysis between WS and IWS using T-MSNet

While Subsection 5.5.1 show the result of WS and IWS based on a single perspective, the inclusion of this experiment will further expand the quantitative analysis. This is done by evaluating IWS and WS with a new CNN model, T-MSNet, and other ASC datasets, being ESC-50 and US8K.

Similarly to Subsection 5.5.1, an exhaustive evaluation is being conducted on IWS where the timescale $TS_1, TS_2 \in \{92ms, 185ms, 371ms\}$. The result is featured in Table 5.4, where Rows 1,2 and 3 are the results of WS. Row 7 is chosen as the baseline for ease of concatenation and consistency in scoring second best for both shuffling and non-shuffling results. Column 'RMS' is the root mean square of the combined accuracy score of the three datasets (excluding with shuffling). Based on the RMS, having a timescale of 371ms seem to be the most optimal and the best IWS is combination 7. Based on the root means square (RMS) of the combined accuracy of DCASE 2021, ESC-50, and US8K, mixing of the first and second-order based on this rule $TS_1 < TS_2$ shown to improve the classification performance of WS. Hence, the finding is in line with the analogy of energy dispersion of the first and second-order based on the timescale discussed by (Andén and Mallat, 2014, Bruna and Mallat, 2013, Andén and Mallat, 2011).

As scattering transform is contractive, (Andén and Mallat, 2014, Bruna and Mallat, 2013, Andén and Mallat, 2011) calculated the energy of dispersion based on the fraction of the squared Euclidean norm of a vector of coefficients of the raw signal $x$ against both $S_1x$ and $S_2x$ (e.g., $\|S_1x\|^2/\|x\|^2$). Their calculation review that as TS gets larger, there will be a shift of energy dispersion from first order to second order. Considering this analysis, the log-scalogram of the first and second-order of the sound of bird chirping is plotted in Figure 5.3 and Figure 5.4. Furthermore, in Section 2.3, there is an illustration of the gaussian filters in Figure 2.6 to illustrate the reason for this phenomenon. TS affects the scale of the gaussian filter, which in this context is a low-pass filter.

Hence, building on this concept, simple mixing of first and second-order with different timescales is being proposed. The effectiveness of IWS has been demonstrated based on the experimental results of all the datasets, especially for ESC-50, which consists of both SER and ASC recordings. For DCASE 2021 and US8K, slight improvement is observed. In addition, the

CLASSIFICATION RESULT OF WS AND IWS WITH T-MSNET

| No | $S_1x$ | $S_2x$ | WO/ Shuffle | W/ Shuffle | ESC-50 | US8K | RMS |
|---|---|---|---|---|---|---|---|
| 1 | 92 | 92 | 67.10 | 68.52 | 64.69 | 78.72 | 70.4 |
| 2 | 185 | 185 | 67.47 | 69.36 | 68.40 | 78.39 | 71.6 |
| 3 | 371 | 371 | 66.73 | 69.19 | 72.57 | 78.36 | 72.7 |
| 4 | 92 | 185 | **68.45** | 69.6 | 68.15 | 78.05 | 71.7 |
| 5 | 92 | 371 | 67.74 | **70.57** | 73.02 | **79.97** | 73.7 |
| 6 | 185 | 92 | 68.05 | 68.75 | 62.98 | 78.74 | 70.2 |
| **7*** | **185** | **371** | 68.28 | 70.37 | **73.75** | 79.76 | **74.1** |
| 8 | 371 | 92 | 67.14 | 66.3 | 62.55 | 77.08 | 69.2 |
| 9 | 371 | 185 | 67.76 | 68.45 | 68.21 | 77.91 | 71.4 |

**Table 5.4.** Classification result of WS and IWS. The classification accuracy of IWS and WS on DCASE 2021, ESC-50 and US8K datasets.

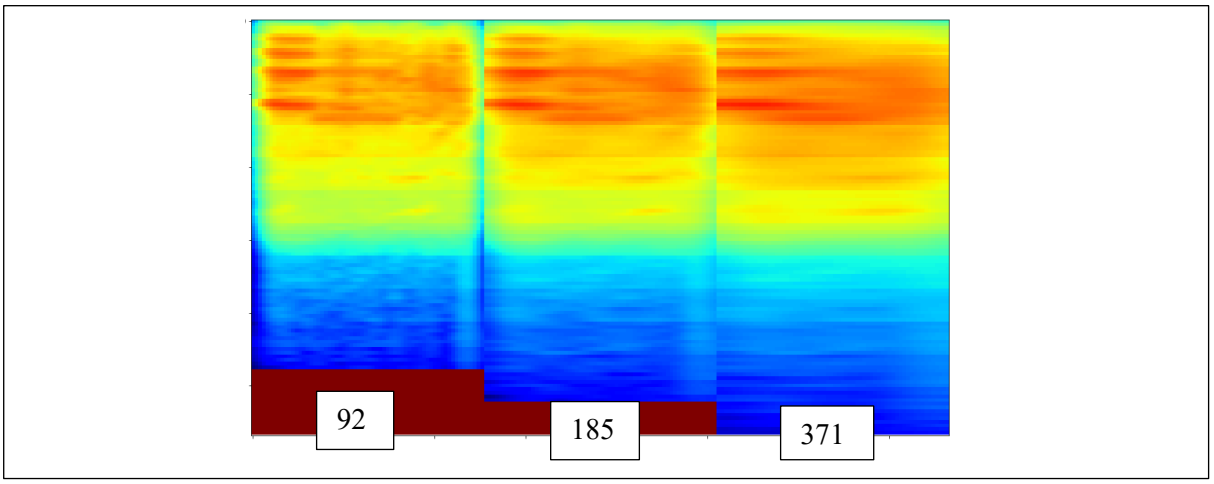



**Figure 5.3.** First-order coefficients presented in log-scalogram of different timescales of acoustic scene in a shopping mall.

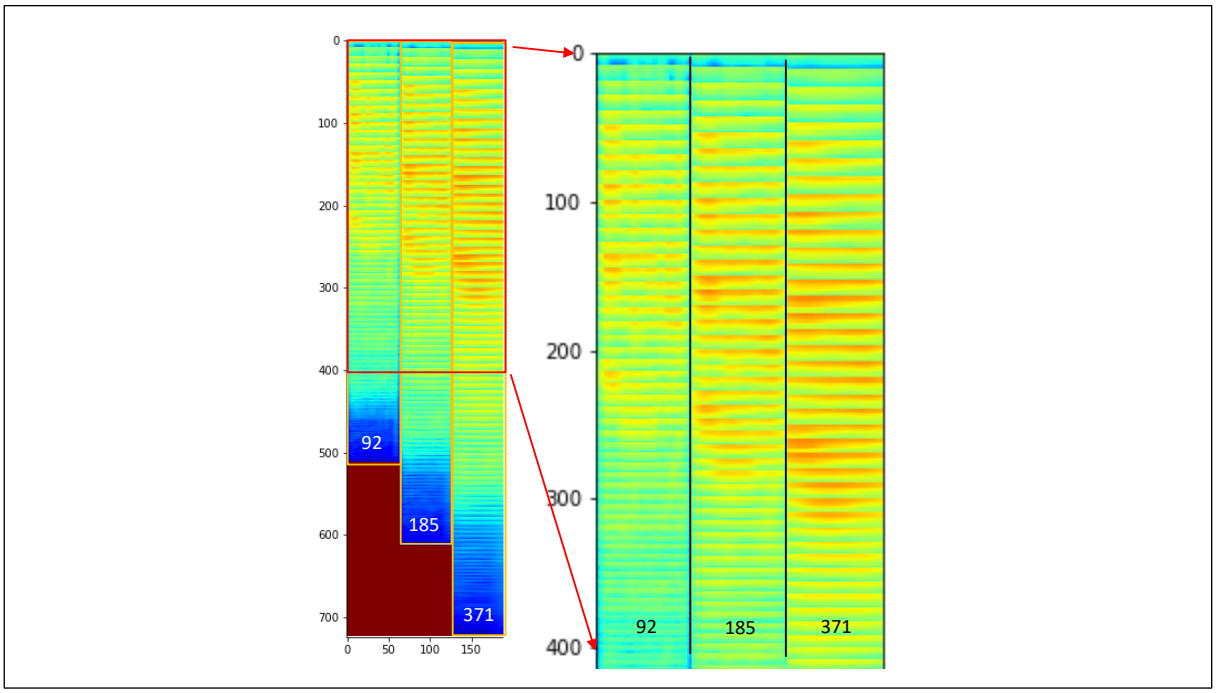



**Figure 5.4.** Second-order coefficients presented in log-scalogram of different timescales of bird chirping sound.

logic of $TS_1 < TS_2$ does increase the number of wavelet transforms, especially for fine-scale resolution. Figuratively, the second-order resolution increase as TS increase from 92ms to 371ms is by at least 100s over additional wavelet transform as illustrated in Figure 5.4. This tentatively means an increase in the wavelet filter's variation, which result in capturing more acoustic characteristic, as explained in Section 2.3. More wavelets with different scales can reduce the effect of Heisenberg Uncertainty Principle. In contrast, setting $TS_2 < TS_1$ does not work well. Typically, this pairing of IWS will always result in a poorer performance against their WS counterpart, as reflected in Section 5.5.1, Table 5.3, and Table 5.4, further backing the energy dispersion theorem.

However, a caveat appears when mixing first and second-order with different timescales. There is a possible occurrence of feature redundancy or irrelevancy. The possibility of redundancy is further grounded by the energy dispersion theorem (Andén and Mallat, 2014, Bruna and Mallat, 2013, Andén and Mallat, 2011), where scattering transform is contractive, i.e. $\|S_0x\|^2 + \|S_1x\|^2 + \|S_2x\|^2 + \cdots + \|S_mx\|^2 = \|x\|^2$ for $m = \infty$. Hence, there will be a possibility of overlapping where the signal is captured by both first and second orders. Coupling with CNN that thrive in complex representation might build a stronger relationship between the overlapped wavelet scattering coefficients.

Thus, IWS can seem to be a double-edged sword without thorough experimentation. For this setup, the best combination of IWS for this setup is the combination of $TS_1 = 185ms\ with\ TS_2 = 371ms$. While the timescales of the best IWS in this setup are different from Section 5.5.1, the rules of $TS_1 < TS_2$ still applies. Notably, GAFS is only evaluated on TSCNN.

## 5.6. Multi-Timescale with GA for Feature Selection Analysis

### 5.6.1 GA for Feature Section on IWS

The earlier findings of Subsections 5.5.1 and 5.5.2 perpetuate the study of GA for feature selection for IWS. The entire system is called GATSCNN, and the experimentation results are tabulated in Table 5.5 where the first to third row are the result from Table 5.3, row 5,6 & 7 and is used for comparison between with GA and without GA runs. Row 7,8 & 9 is the combination of models by averaging the final logits. In rows 4,5 & 6 under column "feature subset", the value in the bracket is the number of features not being selected. Notably, the experiments demonstrated that GA with the first order did not amount to any improvement. As such, GA for the first order is not being evaluated further. Hence, for experiments GATSCNN and GATSCNN-2, GA is only applied in the second-order.

CLASSIFICATION RESULT OF GATSCNN

| No | Features | FEATURE SUBSET | GA | ACCURACY |
|---|---|---|---|---|
| 1* | 46-92 | FULL 513 | No | 70.72% |
| 2* | 46-185 | FULL 614 | No | 70.42% |
| 3* | 46-371 | FULL 724 | No | 70.51% |
| 4 | (1) | 362 out of 513 | Yes | 70.41% |
| 5 | (2) | 445 out of 614 | Yes | 71.12% |
| 6 | (3) | 524 out of 724 | Yes | 70.14% |
| 7 | (4+5) | - | - | **73.00%** |
| 8 | (4+5+6) | - | - | 72.70% |
| 9* | (1+2+3) | - | - | 71.30% |

**Table 5.5.** Classification result of GATSCNN. This is the classification result of GATSCNN and only the second-order is of interest while the first order is left untouched for all the experiments.

The application of GA has shown optimistic results for Table 5.5, row 5, while for Table 5.5, rows 4 & 6, there is a slight decrease in accuracy of approximately 0.3%. However, for all cases, Table 5.5, rows 4, 5 & 6, GA has managed to reduce the dimension of the features by at least ~27%, which translates to a reduction in model training cost of 20s-25s per epoch presented in Table 5.7. Hence, reaffirming that the second-order coefficients of WS consist of redundancy and, for the case of Table 5.5, row 5, irrelevancy too.

Redundancy can likely occur in second-order coefficients as it is a more refined representation of first-order coefficients. Hence, there lies a possibility that a sound source is captured in multiple second-order coefficients. On the other hand, irrelevancy occurs when no important sound sources reside in that frequency spectrum, or a frequency spectrum shares similar second-order coefficients with the rest of the data. Thus, rendering the second-order coefficients not useful when discriminating different acoustic scenes.

Irrelevancy is evident when combining the models by averaging the final logits from the softmax layer, as shown in Table 5.5, rows 8 & 9. In the comparison of Table 5.5, rows 8 & 9 have shown an improvement of 1.30% classification accuracy, on top of a combined computational cost reduction of 30s to 45s per epoch. While combining just Table 5.5, rows 4 & 5, models have shown the best result for GATSCNN implementation.

### 5.6.2 GA for Feature Section on Multi-timescale WS

In this experiment, the Multi-timescale WS is created based on the findings of Section 5.4, being 46ms, 92ms, 185ms, and 371ms. While evaluating a single second-order coefficient in Section 5.6.1 has highlighted the existence of redundancy and irrelevancy, the effect is greater

when all the timescales is being combined, as shown in Table 5.6. This table provides a tabulated perspective on the comparative study of FS1 vs FS2 vs FSALL. FS1 is constructed by concatenating the feature subsets from Table 5.5, rows 4, 5 & 6. Following Table 5.5, the feature subset only reflects the total number of features for the second-order. The algorithm for the construction of FS1 is described in Section 4.2. FS2 is the best solution from GATSCNN-2. Lastly, FSALL is the full set of features from the three timescales.

Prior to evaluating GATSCNN-2, another feature is constructed with all the timescales based on the second-order feature subsets from Table 5.5, rows 4, 5 & 6, and termed it as 'FS1'. FS1 is constructed using a statistical-based approach on each timescale.

CLASSIFICATION RESULT OF GATSCNN-2

| No | Features | MODEL | FEATURE SUBSET | GA | ACCURACY |
|---|---|---|---|---|---|
| 1 | FS1 | TSCNN | 920 | - | 72.60% |
| 2 | FS2 | TSCNN | 923 | Yes | 72.89% |
| 3 | FSALL | TSCNN | 1851 | No | 72.32% |
| 4 | FS1 | TSCNN-2 | 920 | - | 71.55% |
| 5 | FS2 | TSCNN-2 | 923 | Yes | **73.32%** |
| 6 | FSALL | TSCNN-2 | 1851 | No | 72.36% |

**Table 5.6.** Classification result of GATSCNN-2. This is the classification result of GATSCNN-2 and a comparative study of FS1 vs FS2 vs FSALL.

Let use Table 5.5, row 4, as an example. The final product of GATSCNN provides the best solution and a set of parents. This set of parents has the ‘best solutions’ from the GA run, and the best solution can be expressed as ‘the best of the best’. With the best solutions or parents, a frequency model is built based on the number of occurrences each feature or gene is being selected. Hence, in this context, a highly desired feature will have a frequency count of 8, and the worst case will have a frequency count of 0. After calculating the number of occurrences, only the top 50 percentile of the features is being retained. This process is also emulated on Table 5.5, rows 5 & 6. Notably, a trail is conducted with various retention percentages and in this thesis, only the ‘retention policy’ with the best classification accuracy is being presented. Lastly, all 3 timescales are concatenated together along the frequency axis.

Next, FS1 is compared against 'FS2'. 'FS2' is the optimal feature subset output from running GATSCNN-2. The difference between FS1 and FS2 is that FS2 has feature interaction between the second-order of different timescales during the feature selection process. The selection based on the three timescales directly constitutes the classification accuracy. Hence, based on Table 5.6, FS2 has the best performance for all the outputs, whether it is evaluated on TSCNN or TSCNN-2. While comparing FS1 and FS2, this paper reconfirms the importance of feature

interaction (de la Iglesia, 2013, Oreski and Oreski, 2014). The lack of feature interaction is even more apparent in Table 5.6, row 4.

Indeed, the combination of the four timescales has produced features with high redundancy and irrelevancy, which resonate with the plot shown in Figure 2.7. Thus, the application of GA has removed redundancy and irrelevancy while achieving better performance in Table 5.6, row 5, yielding 73.32% accuracy, which is almost 1% gain in accuracy, while ~50% reduction in feature dimension, translating to saving ~40% of the computational time as shown in Table 5.7. The computational time is recorded based on the average time to train each epoch with a batch size of 32. (~80s per epoch faster of the usual ~200s per epoch, which can be expressed as $80s/200s * 100\% = 40\%$ )

COMPUTATIONAL TIME ANALYSIS

| No | Features | TIME(*S*) | FS TIME (*S*) |
|---|---|---|---|
| 1 | 46-92 | 75 | 53 |
| 2 | 46-185 | 84 | 57 |
| 3 | 46-371 | 98 | 77 |
| 4 | ALL | 200 | 120 |

**Table 5.7.** Computational time analysis. This Table presents the computational time required before and after feature selection, column TIME, and FS TIME, respectively.

### 5.7. Ablation study of the proposed Two-stage Mobile Shuffling Network

The main scope of this evaluation is targeted at the analysis of shuffling modules. In Subsection 5.7.1, this section performed an ablation study of SSS, TPS, and CS. The ablation study is broken down into two parts: the first part is an evaluation of the individual type of shuffling modules. In this experiment setup, an investigation on the relationship between the position of the shuffling module and the number of shuffling modules that will influence the performance of the T-MSNet is being performed. Subsection 5.7.2 involves the study of the synergy of collectively combining SSS, TPS, and CS. Lastly, the robustness of shuffling modules on all the ASC datasets is evaluated.

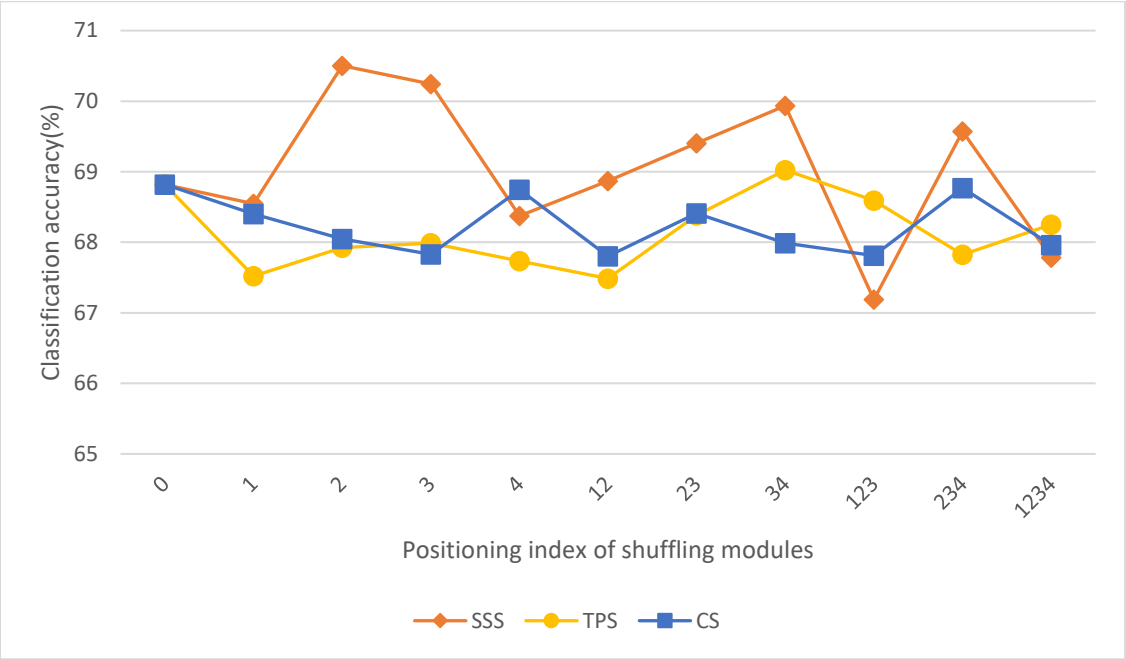
#### 5.7.1. Ablation study of SSS, TS amd CS

This experimentation investigates the effectiveness of the individual type of shuffling modules when 'activated' in different locations of T-MSNet, being S1, S2, S3 & S4 termed in Chapter 4, Section 4.4. Individual type of shuffling modules means that SSS, TPS, and CS are evaluated separately. Overall, SSS has the best performance in classification accuracy and logloss as illustrated in Figure 5.5. This exhibits that SSS is more effective for a time-frequency representation, especially IWS (due to its high-frequency dimension). The first operation of

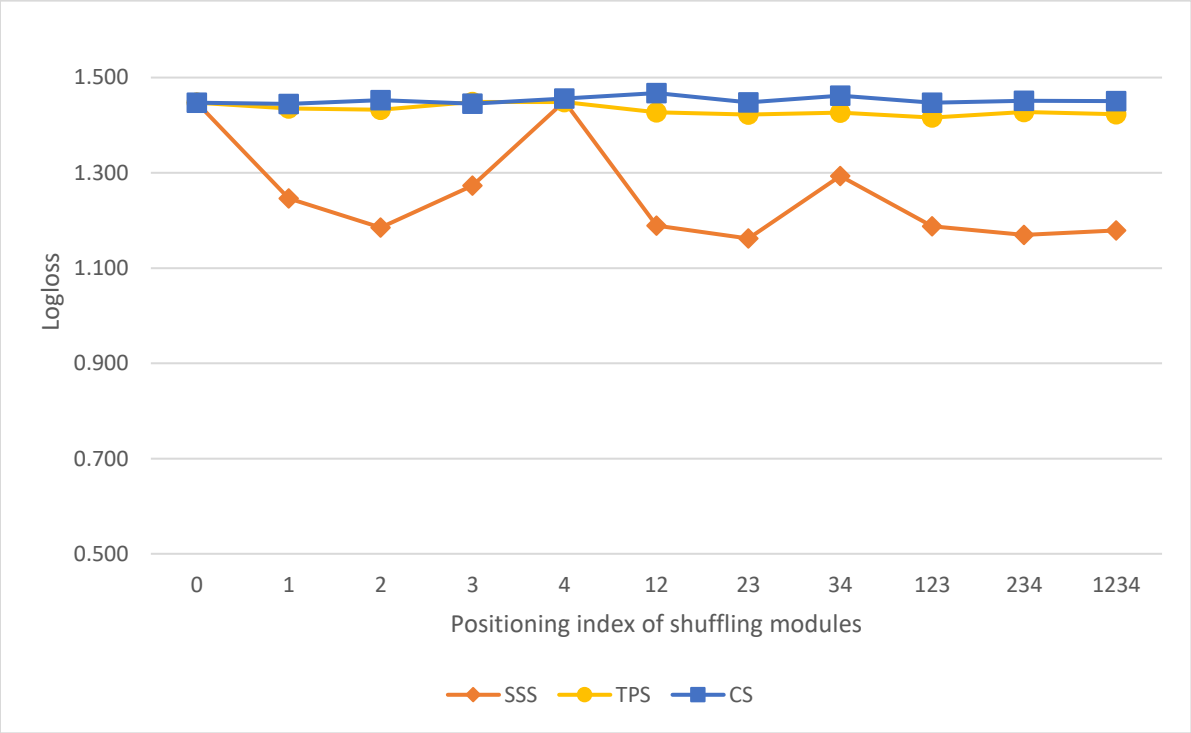
SSS, which splits the feature maps 'frequency-wise', corresponds to the signal processing technique of applying equal frequency binning or using linearly scaled filters on the spectrogram. (The more common practice is applying spectrogram with mel-scale filterbanks) Hence, a slice of the feature maps has similar characteristics to a frequency bin. Next, the shuffling operations mix the frequency bins around such that the feature maps are no longer ordered based on frequency magnitude (e.g., 0-100Hz, then 101-200Hz, then 201-300Hz, …). This forces the network to learn the general "shape" of the acoustic rather than the pitch. E.g., It no longer matters to the network whether it is a female voice or male voice since it just generalized that this is a speech. Hence, the general consistency in the improvement of logloss is demonstrated by SSS. The improvement of logloss is also reflected in other datasets. (Ref. Figure 5.5b)

However, the effect of improved model generalization is lost when applying SSS on S4. The inclusion of SSS on S4 has been demonstrated to result in a higher logloss. We attributed the cause of this observation to the relatively smaller dimensional size of *F*, which is 30. (Refer to Table 4.6) The poorer performance observed when shuffling with a low-dimension size is in unison with the findings (Fan et al., 2021). They concurred that channel shuffling is ineffective at earlier layers because of limited channels. Another observation made is that the classification accuracy dips when SSS is activated on S1. One possibility is that the network needs to learn initial spatial relationships and to shuffle better on feature maps with higher dimensional space. (a few layers of CNN with non-linear activation function) The same setup is also being performed on TPS and CS. However, their classification accuracy performances are erratic and do not improve the logloss. Furthermore, in some cases, the classification accuracy worsens with the inclusion of TPS or CS. This observation is again supported by the findings of (Fan et al., 2021), where shuffling does not work effectively on low-dimension sizes.

The proposed SSS works best on a low-complexity model with IWS with a high dimension size of *F*. Hence, there is a need to evaluate the effectiveness of GAFS with T-MSNet proposed in Chapter 4, Section 4.5 to analyze whether shuffling modules work with GAFS. On top of that, including a single SSS on S2 provides the best overall performance.




a) Classification accuracy.




b) Logloss

**Figure 5.5.** Result of various position of SSS, TPS and CS on T-MSNet. Index 1,2,3 & 4 relates to S1,S2,S3 & S4 termed in Chapter 4, Section 4.4.

### 5.7.2. Ablation study of combining SSS, TPS and CS

While the study of Subsection 5.7.1 provides an insight into the individual effectiveness of each shuffling module, this Subsection explored the possible synergy when combining all of them. It is incredibly time-consuming to evaluate all the types of combinations. Based on the observation from Subsection 5.7.1, some rules are being drawn to reduce the number of possible tests. The rules are as follows:

- SSS must always be included as it provides significant improvement to logloss.
- There should not be a combination where SSS is solely activated in S1, S4 and S34 (S34 stands for SSS activated on both S3 and S4), due to poor logloss performance.

Based on the rules, the focus is shifted to SSS being the lead shuffling module and how TPS and CS can enhance it. Even with the rules, there are still several combinations. Hence, a random search is used. (more than 20 combinations, the results are presented in Appendix ***A***) As no particular findings can be observed, this thesis presents the top 3 combinations in Table 5.8. With Row 1 having the best shuffling performance. The overall performance of the combinations is relatively close. Hence, other datasets is used to establish consistency and test the robustness of shuffling modules.

BEST RESULT OF VARIOUS COMBINATION OF SSS,TS AND CS

| No | Combination | Accuracy | Logloss |
|---|---|---|---|
| *1* | SSS2 | 70.50% | 1.185 |
| *2* | SSS234,CS234 | 70.39% | 1.148 |
| *3* | SSS234,CS4 | 70.25% | 1.149 |
| *4* | SSS2,TS34,CS3 | 70.60% | 1.162 |

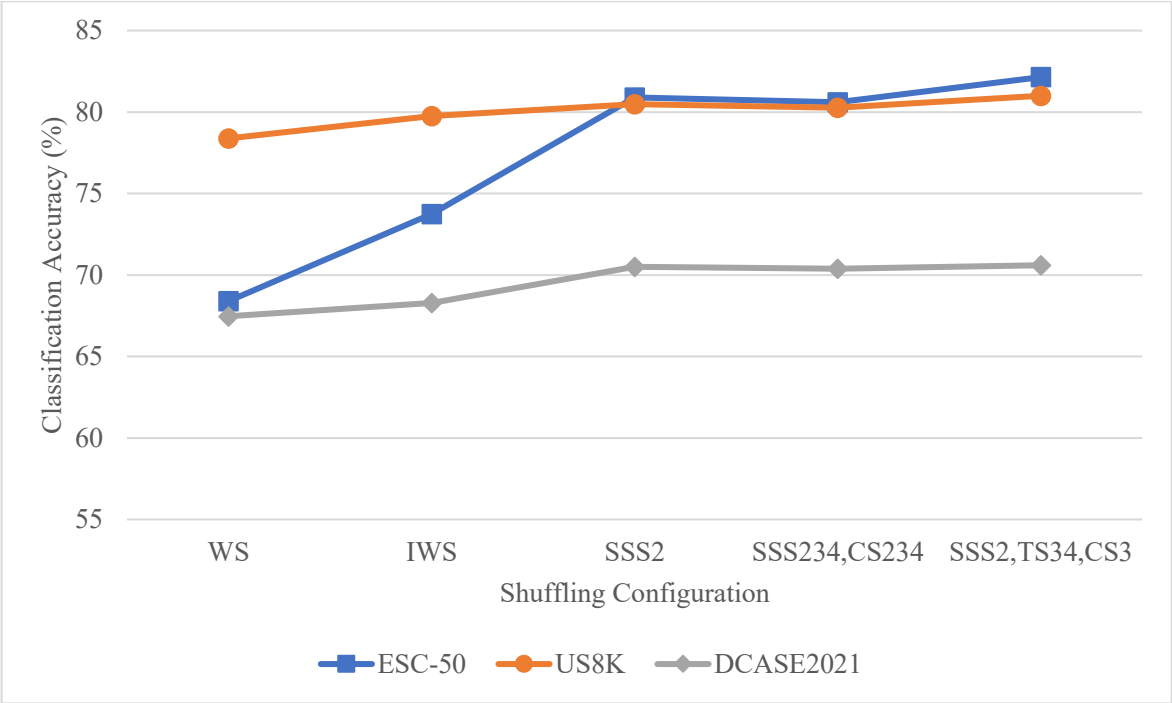
**Table 5.8.** Best result of various combination of SSS,TS and CS. This Table presents the top 3 best combinations of SS, TS and CS based on a set of rules and a random search.

### 5.7.3. Robustness of Shuffling Modules on ASC

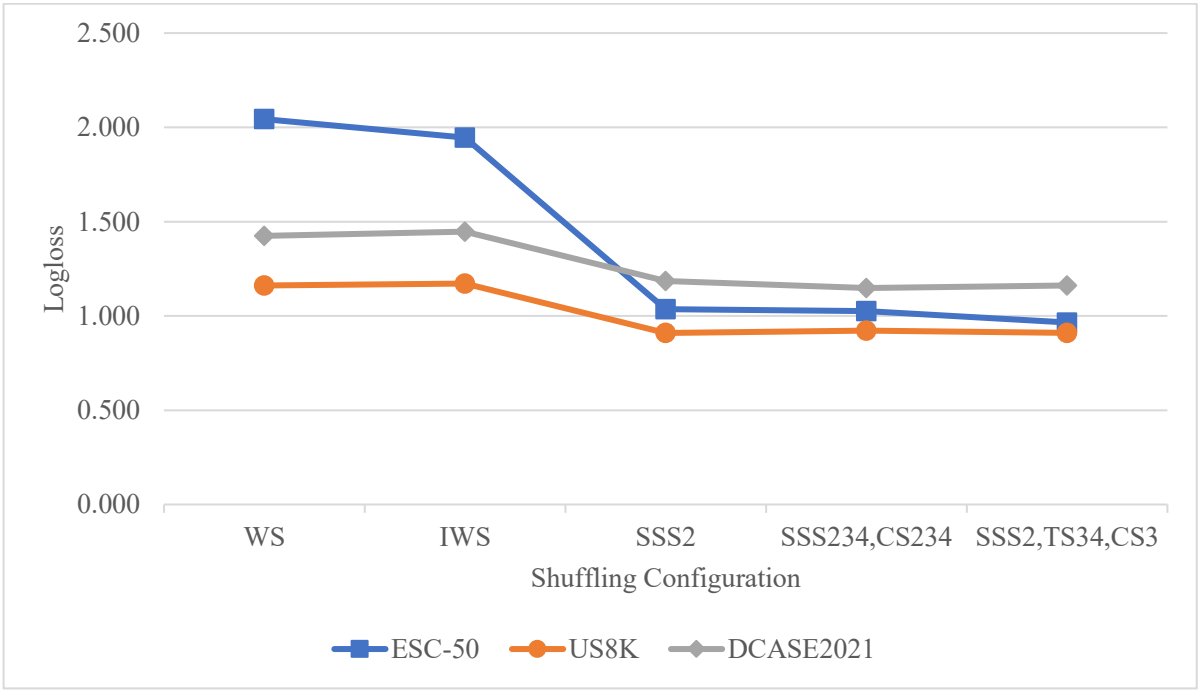
In this Subsection, to examine the robustness of shuffling modules, it is evaluated with additional datasets and the results can be seen in Figure 5.6. A slight progressive improvement is observed when we mixed first and second-order to form IWS for the US8K and DCASE 2021 datasets, while a significant improvement is observed for ESC-50. This can be attributed to the number of classes needed to be classified. The variety of sounds recorded for ESC-50 is larger than US8K and DCASE 2021 (the number of classes is 5 times larger). Thus, it has a richer acoustic characteristic, and by mixing different timescales, the model can distinguish the other acoustic profiles better.

Yet another observation is noticed when comparing the model performance on ESC-50 against US8K and DCASE 2021. A significant increase in performance is observed when IWS was applied and further alleviated when shuffling modules were activated on ESC-50. This can be attributed to the size of the dataset. ESC-50 is a small dataset, and CNN is known for overfitting when the sample size is small. Hence, this finding reaffirmed that the shuffling algorithm could improve the generalization of the model, as mentioned by (Fan et al., 2021, Ma et al., 2018, Zhang et al., 2018c). Another tell-tale of better generalization can be observed by the consistent improvement of logloss when shuffling modules are applied, as exhibited in Figure 5.6b.

Shuffling modules can improve classification accuracy and logloss performance for all the datasets. Generally, the top 3 T-MSNet configurations presented in Table 5.8 have very close performance rating. While the combination of SSS, TS and CS has been shown to provide the best classification accuracy, the activation of a single SSS offers a less complex model (complexity means that it has fewer hyperparameters to configure). In summary, sub-spectral shuffling is the most effective shuffling module for the proposed system setup.




a) Classification accuracy




b) Logloss

**Figure 5.6.** Result of the top 3 variant shuffling modules evaluated on other datasets. WS and IWS are models without shuffling modules. The tabulated result is in Appendix, Table IV (classification accuracy) and Table V (logloss).

## 5.8. Comparative Analysis

### 5.8.1. Comparing GATSCNN-2 with SOTA(s) models presented in DCASE 2020

Table 5.9 tabulated the top 3 models from DCASE 2020 Task1a, being 'Suh' relates to (Suh et al., 2020), 'Hu' relates to (Hu et al., 2020), and 'Gao' relates to (Gao and McDonnell, 2020). Starting after the index column, 'Ref' points to the reference of the paper. 'FEAT' stands for feature, and under feature, LMS stands for log mel-spectrogram. Next column, 'DATA AUG', stands for Data Augmentation and column 'EVAL/DEV ACC' stands for classification accuracy for the evaluation dataset and development dataset. Under column 'DATA AUG', 'SpecAug' stands for spectrum augmentation, which involves randomly masking of the frequency axis and time axis before training. The DRC stands for Dynamic Range Compression. We denote '*Etc' for (Hu et al., 2020), as they used several data augmentation techniques. Under column "MODEL", we included the result of a single model labelled 'SINGLE', VGG-liked refers to similar model as (Simonyan and Zisserman, 2014), and 'McDonnell' is the CNN model from (McDonnell and Gao, 2020). This is to provide another perspective when comparing the Genetic Algorithm with Two-stage Convolution Neural Network (GATSCNN-2) model.

COMPARISON WITH DCASE 2020 TASK1A MODELS

| No | Ref | FEAT | DATA AUG | MODEL | EVAL/DEV ACC |
|---|---|---|---|---|---|
| 1 | 'Suh' | LMS | Mix Up<br>Temporal Crop | Ensemble<br>TSCNN-liked<br>Frequency-wise dilated conv | 76.5%/74.2% |
| | | | | SINGLE | 75.5%/73.7% |
| 2 | 'Hu' | LMS | Reverberation + DRC<br>Mix Up<br>SpecAug<br>*Etc. | Ensemble of VGG-liked and TSCNN-liked | 76.2%/81.9% |
| | | | | SINGLE | -/74.6% |
| 3 | 'Gao' | LMS | Mix Up<br>Temporal Crop | Ensemble of 'McDonnell', 'McDonnell'+ focal loss and 'McDonnell'+ auxiliary binary classifier | 75.2%/72.5% |
| | | | | SINGLE | 75.0%/71.7% |
| 4 | Prop* | WS | Mix Up | GATSCNN-2 | -/73.3% |

**Table 5.9.** Comparison with DCASE 2020 Task1a models. This Table presents the top 3 models from DCASE 2020 Task1a.

Notably, most of the models adapted a Two-staged Convolution Neural Network (TSCNN)-liked or TSCNN variants approach, and all of them uses log mel-spectrogram. While not stated in the Table, all the models use the same learning strategy of SGD with warm restart. The models are ranked based on the evaluation dataset set classification result. Notably, all the authors (Hu et al., 2020, Suh et al., 2020, Gao and McDonnell, 2020) and this paper adopted a two-stage convolutional architecture. While their best performing models are all ensemble models, results on their single model are also provided to have a better perspective when comparing against the proposed model.

Based on Table 5.9, the proposed model has the edge over (Gao and McDonnell, 2020), regardless of comparing against a single model or the ensemble model. Adding the comparison of (Gao and McDonnell, 2020, Suh et al., 2020, Hu et al., 2020) to the analysis, GATSCNN-2 comes short of 0.4%. The combined comparison analysis between GATSCNN-2, (Suh et al., 2020), and (Gao and McDonnell, 2020), further established the need for higher frequency resolution. While (Gao and McDonnell, 2020), only uses 128 frequency bins, while (Suh et al., 2020) have double the frequency bins (256). Likewise, in the experiment, the need for multi-timescale is being observed, which also has the same effect as increasing the frequency resolution to improve the classification accuracy (Refer to Table 5.5, row 5 against Table 5.6, row 5).

In the case of (Hu et al., 2020), they approached this problem using a myriad of data augmentation techniques. Their single model architecture is quite similar (Gao and McDonnell, 2020), and the only difference is that the number of channels after each convolution has doubled. The application of various data augmentation techniques resulted in a dramatic classification accuracy of 74.6% and 81.6% on the development dataset (single and ensemble models, respectively). However, their accuracy dipped from 81.6% to 76.2% on the evaluation dataset. When compared with a single model (Hu et al., 2020) against (Gao and McDonnell, 2020), the model suppressed by 2.9% on the development dataset. The estimated gain can range from 0.7% to 1.2% as there is no classification accuracy recorded for the single model (Gao and McDonnell, 2020). By estimating using the available ensemble results, there is a possibility of the higher classification accuracy contributed by the multiple data augmentation techniques despite increasing the training time leading to higher computing resources.

In short, GATSCNN-2 fall short compared to the first and second models due to having an ensemble model. While comparing against single model, the gap become closer with 0.4% differences against the top model and 1.3% differences against the second place. The possible

reason is the change in convolution layers design expedited in 'Suh' model and the used of more advance data augmentation techniques such as 'SpecAug' observed in 'Hu' model.

Based on the comparative study, the best way to improve the classification accuracy is to enhance the frequency resolution. However, doing so will have a caveat, the increase in frequency resolution will directly translate to an increase in computational cost in Table 5.10, row 2 compared to row 1, and for WS, row 4 to row 6. While Table 5.10 presented a

COMPUTATIONAL TIME ANALYSIS WITH DIFFERENT FEATURE SIZE

| No | Ref | FEAT | FEAT SIZE (*C* IS EXCLUDED) | FLATTEN SIZE | TIME(S) |
|---|---|---|---|---|---|
| 1 | 'Suh' | LMS | 1. 64 x 423<br>2. 64 x 423<br>3. 128 x 423 | 108,288 | 160 |
| 2 | 'Hu', 'Gao' | LMS | 1. 64 x 423<br>2. 64 x 423 | 54,144 | 86 |
| 3 | GATSCNN | 46-185 | 1. 87 x 248<br>2. 445 x 120 | 74,976 | 57 |
| 4 | GATSCNN | 46-185 FULL | 1. 87 x 248<br>2. 614 x 120 | 95,256 | 84 |
| 5 | GATSCNN -2 | FS2 | 1. 87 x 248<br>2. 306 x 129<br>3. 256 x 120<br>4. 361 x 120 | 135,090 | 120 |
| 6 | GATSCNN -2 | FSALL | 1. 87 x 248<br>2. 614 x 120<br>3. 513 x 120<br>4. 724 x 120 | 243,696 | 200 |

**Table 5.10.** Computational time analysis with different feature size. This Table presents a computational time analysis of DCASE 2020 Task1a top 3 features shown in TABLE VIII and the proposed models.

computational time complexity comparison between the features, it is still not 'fair' to directly compare LMS against WS as the CNN architecture significantly contributes to the computational time complexity. Hence, the comparison is made based on the size of the input representation. As mentioned in Chapter 4, the size of the input representation does contribute to the time complexity. Presented in Table 5.10 is a comparison of the feature size of LMS

against WS with GA feature selection and providing a reference towards the computationally time competitiveness of the proposed GATSCNN-2. This Table presents a computational time analysis of DCASE 2020 Task1a top 3 features shown in Table 5.9 and the proposed models. C denotes the number of channels, and in this case, between (Hu et al., 2020, Suh et al., 2020, Gao and McDonnell, 2020) and our features all have 3 channels, being the original feature map followed by the deltas and deltas-deltas. Hence, we excluded it for brevity. Next, the feature map is split into groups and fed to TSCNN, hence, column 'FEAT SIZE' shows the dimension of the input representation of TSCNN. Under column 'FEAT', row 4 and row 6 address the full features of the first and second-orders. Providing a 'fair' computational time comparison between (Hu et al., 2020, Suh et al., 2020, Gao and McDonnell, 2020) and the proposed models is rather challenging. The computation time complexity of training a model is impacted by various factors, such as the number of data augmentations being used, differences in CNN model architecture, number of epochs per training and whether it is an ensemble model. Hence, the attention is drawn towards analyzing the input representation. The controlled setup is described below. The time complexity recorded is based on a single model run (not ensemble model) and based on the average time to train each epoch with a batch size of 32. A restriction is placed on the data augmentation, such that only Mix Up is applied, and the feature size for LMS is the result after temporal crop. As for the model, a TSCNN-liked architecture proposed by (McDonnell and Gao, 2020) for (Suh et al., 2020) & (Gao and McDonnell, 2020) is adapted, except that we increase the number of initial channels to 32 for the first convolution layer. For (Suh et al., 2020), the TSCNN-liked architecture proposed by them without dilated convolution layer is used. (with dilation, the time per epoch increases to 183s) The distinct difference between the TSCNN models and (Hu et al., 2020, Suh et al., 2020, Gao and McDonnell, 2020) is that there is frequency downsampling for the second-order(s). Hence, WS has a larger flatten dimension than LMS but a faster computational time. Overall, this Table presents the importance of employing GA for feature selection to make multi-timescale WS computationally time competitive on top of the improved classification result.

In summary, a fresh approach is presented on using GA feature selection to combine multi-timescales WS. It has shown that it improves the classification accuracy performances and makes multi-timescales WS computationally time competitive. This approach was not realized (Hu et al., 2020, Suh et al., 2020, Gao and McDonnell, 2020). The proposed methods include a direct increase in frequency resolution, improved CNN architecture, improved loss function, incorporating, and an ensemble model. All proposed approaches improve the classification accuracy at the price of added computational time.

### 5.8.2. Comparing T-MSNet with SOTA(s) models presented in various datasets.

In this section, a comparison between the proposed T-MSNet and SOTA(s) models on various datasets is tabulated in Table 5.11 where [1] relates to (Arandjelovic and Zisserman, 2017), [2] relates to (Lopez-Meyer et al., 2021), [3] relates to (Mohaimenuzzaman et al., 2021), [4] relates to (Piczak, 2015), [5] relates to (Zhang et al., 2018d), [6] relates to (Mohaimenuzzaman et al., 2022), [7] relates to (Kim et al., 2022), [8] relates to (Yang et al., 2021), [9] relates to (Koutini et al., 2019), [10] relates to (Mart'in-Morat'o et al., 2021), and * denotes the proposed model T-MSNet. Logloss is not recorded for ESC-50 and US8K as models are compared based on classification accuracy. AVC stands for Audio-visual correspondent, a self-supervised learning approach to build audio and image embeddings. End-to-end means that the network take in the raw waveform as the input representation. CF stands for convolution factorization. KD stands for knowledge distillation. Generally, all the models are CNN. Furthermore. the differences between T-MSNet and theirs is being discussed.

**ESC-50**

T-MSNet has an advantage over human classification accuracy presented (Piczak, 2015). L3-net developed by (Arandjelovic and Zisserman, 2017) uses self-supervised learning with an unlabeled video dataset to build audio and image embeddings. In the case of (Heittola et al., 2020), they have demonstrated the effectiveness of embeddings on the DCASE 2021 dataset. However, the model size complexity cost of embeddings is too high. Hence, it is not recommended for low complexity modeling.

While compared to (Lopez-Meyer et al., 2021), the addition of pretraining the model with a very large dataset, AudioSet (Gemmeke et al., 2017), has improved performance dramatically. However, a lesser gain is observed when the same setup is applied on US8K. This can be attributed to the dataset size; transfer learning from an extensive dataset is more effective when the dataset is small. The effect diminishes when applied to a larger dataset. Although transfer learning from a very large dataset has shown to be very useful, it requires massive resources and training time first to learn the semantics of the very large dataset.

Lastly, (Mohaimenuzzaman et al., 2021) proposed a model that has nearly the same model size as T-MSNet. Pruning and quantization are used to bring down the model's parameters drastically. T-MSNet has a slight edge over the pruning technique in this scenario. Overall, themodels used in ESC-50 are not geared towards low-complexity modeling. Hence, the best benchmark is against (Piczak, 2015) and (Mohaimenuzzaman et al., 2021). Notably, without the limitation on model size, (Gong et al., 2021, Guzhov et al., 2022) are current SOTA models

ACCURACY COMPARISON OF T-MSNET WITH STATE-OF-THE-ART

| | Method | Accuracy/Logloss | Parameters/Size |
|---|---|---|---|
| | | ESC-50 | |
| [1] | $L^3$-net<br>Self-supervised with AVC<br>CNN(8 layer standard)<br>External Dataset | 79.3% | ~4.7M/- |
| [2] | CNN<br>AudioSet | Audioset: 88.64%<br>Scratch: 72.71% | 635,002/- |
| [3] | CNN<br>Pruning Quantization with 8-bit | 81.50% | 131K/157kB |
| [4] | Human accuracy | 81.30% | - |
| * | CNN<br>CF<br>Quantization | 73.75% | 64,788/126KB |
| * | With shuffling | 82.15% | 64,788/126KB |
| | | US8K | |
| [2] | Same as ESC-50 | Audioset: 81.80%<br>Scratch: 74.77% | 635,002/- |
| [5] | CNN<br>CF | 79% | ~508K /2.05MB |
| [6] | CNN of [3]<br>Pruning<br>Quantization | 70.93% | 130K/133.12KB |
| * | Same as ESC-50 | 79.76% | 64,788/126KB |
| * | With shuffling | 81.00% | 64,788/126KB |
| | | DCASE 2021 | |
| [7] | CNN<br>Pruning<br>Mix Quantization<br>KD | EVAL: 76.1%/0.724<br>DEV: 75.9%/0.716 | 66.1K, 8-bit+ 29.4K, 16-bit/122.5KB |
| [8] | CNN<br>KD<br>Pruning (LTH) [53]<br>Quantization | EVAL: 72.9%/0.758<br>DEV: ~79.4%/0.640 | -/125KB |
| [9] | CNN<br>Pruning<br>Quantization | EVAL: 72.1%/0.834<br>DEV: 69.5%/0.890 | 64,625/126.22KB |
| [10] | Official Baseline using CNN | EVAL: 45.6%/1.730<br>DEV: 46.9%/1.461 | 46,246/ 90.3KB |
| * | Same as ESC-50 | DEV: 68.28%/1.148 | |
| * | With shuffling | DEV: 70.39%/1.148 | 64,788/126.6KB |

**Table 5.11.** Acuuracy comparison of T-MSNet with state-of-the-art. The Table presents the consolidated SOTA of 3 datasets.

that achieve ~95% and ~97% classification accuracy, respectively. The standard technique used by both models is pre-learning semantics from the external dataset(s). In the research (Gong et al., 2021), they adapted 'Transformer' network and modified it to be suitable for audio classification. ImageNet and AudioSet are used to pretrain the network. Researchers have incorporated a CLIP framework (Guzhov et al., 2022) with a CNN model dedicated to audio classification. In short, the CLIP framework is the joined learning of multi modalities, typically visual and textual. They extended the number of modalities to include audio modalities. The output of CLIP is a shared multimodal embedding space. Similar to (Arandjelovic and Zisserman, 2017), (Gong et al., 2021, Guzhov et al., 2022) requires tremendous resources and training time.

**US8K**

Similar to ESC-50, US8K is not geared towards low-complexity modeling. Hence, T-MSNet remains competitive against the models presented in Table IV. While comparing to (Arandjelovic and Zisserman, 2017), the inclusion of an additional dataset only shows a slight advantage against T-MSNet. The work (Zhang;Zou and Wang, 2018) uses convolution factorization and shares the same approach as T-MSNet. However, T-MSNet has the edge over their model in terms of both performance and model size. T-MSNet differentiation lies with the inclusion of shuffling modules, which has improved the generalization of the model. Lastly, (Mohaimenuzzaman et al., 2022) reused the CNN model proposed by (Mohaimenuzzaman et al., 2021) and performed a series of pruning and quantization. Their best result that pushes the model size to 133.12KB only stands at 70.93%. In addition, (Guzhov et al., 2022) have also been evaluated with US8K and achieved a SOTA result of 90.07%.

**DCASE 2021**

Lastly, Table V shows the comparison of the top 3 models against T-MSNet. The models are ranked based on DCASE 2021 Task1a challenge result. The main difference between T-MSNet against theirs is the use of a parameters pruning strategy. Parameters pruning removes weights that the employed pruning strategy deemed less important. This strategy has proved to be extremely effective in compressing the size of a very large model (Kim et al., 2022, Yang et al., 2021, Koutini et al., 2019, Mart'in-Morat'o et al., 2021), hence, the popularity of applying this technique. In addition to pruning, (Kim et al., 2022, Yang et al., 2021) uses knowledge distillation to transfer 'learning' from the larger model to the pruned network.

Although T-MSNet falls short compared to logloss, the classification accuracy put us in third place. Furthermore, this thesis presents a different approach to carefully constructing the CNN

architecture using convolution factorization and design patterns. Instead of pruning and knowledge distillation, shuffling modules, a zero-model size cost algorithm, is proposed to improve model generalization.

Table 5.11 shows that pruning, convolution factorization, knowledge distillation, and quantization effectively compress the model. However, supporting pruning and knowledge distillation is the requirement of a well-constructed CNN. In contrast, convolution factorization requires careful planning for each layer, and hence, both have their disadvantages. In addition to the above-stated techniques, shuffling modules, a stochastic method, improves model generalization and favours low-complexity modelling as it does not constitute model size.

### 5.9. Chapter Summary

This chapter provides results and evaluations on the experimentation of the proposed model and the datasets presented in Section 1.4 and can be divided into three parts. The first part begins with the sharing on how the experimentation is being executed. In this discussion the hyper-parameters setting, and feature extraction process is further elaborated.

The second part of the chapter revolves around investigation and ablation studies of the proposed models. The first investigation is an ablation study of TSCNN. TSCNN is designed to tackle the magnitude disparity problem of the first and second-order coefficients. The results shows that by adjusting the proportion between specialized and centralized learning to an optimal position, the TSCNN can slightly elevate the magnitude disparity problem. In addition, the optimal ratio of specialized and centralized learning is highly dependable on the CNN architecture, specially convolution block design, and the input representation, in this case, log mel-spectrogram and wavelet scattering.

The second investigation is a study on normalization. While magnitude disparity problem is typically resolved by performing renormalization, this paper proposed that normalization technique such as batch normalization, that is incorporated in the CNN architecture can replace the renormalization procedure. The results shows that generally, normalization is required to improve the performance of the model, and batch normalization is the most effective normalization technique and can be used to replace renormalization.

Moving on to the next study is an analysis on the timescale of both log-mel spectrogram and wavelet scattering. A range of timescale, TS = [23,46,92,185,371,743,1480]ms is evaluated in this study. It is evident that Heisenberg Uncertainty Principle affects both log-mel spectrogram and wavelet scattering. In addition, the study highlighted the optimal range of timescales, being 46ms, 92ms, 185ms, and 371ms for WS, which is a crucial information when constructing IWS

and multi-timescale WS. Hence, the subsequent investigation continues with the analysis of timescale for IWS, where it is observed that regardless of CNN architecture, TS for the first order must always be smaller than the TS of the second-order when constructing IWS.

Next, is the experimentation on the GA for feature selection, which is delivered in two sessions, first is on GAFS for IWS, and second is on GAFS for Multi-timescale WS. For both cases, GAFS has generally improved the performance of the model in accuracy and time complexity. Hence, a conclusion is drawn stating that there lies redundancy and irrelevancy for IWS and Multi-timescale WS. While a slight improve in accuracy is observed, the reduction in feature dimension and time complexity is obvious. GAFS helps to downsize IWS by ~27% and Multi-timescales WS by 50%, which directly translate to 30% and 40% decreased in computational time, respectively.

While the earlier study is on time complexity, the next set of experimentations study the proposed model T-MSNet, which is a low complexity model with 64,788 parameters and has a size of 126KB. This model fits the profile of deploying in a less computationally powerful devices such as smartphones. T-MSNet is embedded with shuffling modules which is heavily studied as there is a lot of possible combination involve when curating the 'best' T-MSNet. From choosing the type of shuffling modules, being channel shuffling, the proposed sub-spectral shuffling, and temporal shuffling, to in which layer should the shuffling module(s) be activated. This paper tackles this problem, by first experimenting with isolated type of shuffling modules. This means that the T-MSNet will only be embedded with a type of shuffling modules and based on the result, a set of rules is constructed before performing a random search with a mix of shuffling types. Based on the ablation study, sub-spectral shuffling has shown to be the most effective shuffling modules. In addition, sub-spectral shuffling when embedded correctly is the only shuffling module to improve logloss.

Lastly, this chapter wraps up by comparing the proposed model, GATSCNN-2, and T-MSNet with SOTA(s) models presented in DCASE 2020, DCASE 2021, ESC-50, and US8K. GATSCNN-2 comfortably beats the third place, while the score closely matched the first place based on a single model comparison. For T-MSNet, it stands in the third place in DCASE 2021 based on classification accuracy but falls short when comparing with logloss. Low complexity modelling is a rather new area for ASC, hence, researchers who benchmarked their model with ESC-50 and US8K do not usually head towards the direction of developing low complexity model. Therefore, this paper compares models that have low complexity modelling in mind, while noting that there are other more superior models. In both cases, T-MSNet is able to perform rather well, except if it is compares to those models that uses pretraining with Audioset.

# Chapter 6. Conclusion and Future Works

## 6.1. Summary of findings

In the first part of this thesis, an investigation is conducted on wavelet scattering coupled with CNN to tackle ASC task and has resulted in the identification of three weaknesses of wavelet scattering. The three weakness are magnitude disparity problem, Heisenberg Uncertainty Principle, and large frequency dimension.

Magnitude disparity problem is a nature characteristic of wavelet scattering as the cascading wavelet transform operations will cause a huge different in magnitude between the lower and higher orders. In this case, the problem is fixated at first and second-order. Typically, a technique called renormalization is introduced to elevate this problem. However, this thesis proposed a two-stage convolution neural network architecture coupled with batch normalization to ease off this problem. Batch normalization is commonly used to indicate that the normalization is part of the CNN architecture, however, there are also other type of 'network' normalization techniques. The outcome of the proposed TSCNN shows a slight improvement in elevating the problem, while normalization really does the work. Overall, the ablation study of various normalization techniques has pointed out that batch normalization works best for WS and 'network' normalization techniques can be used to replace renormalization.

Heisenberg Uncertainty Principle is often used to describe the problem faced by log mel-spectrogram which uses a fixed timescale to construct the time-frequency representation. The Heisenberg Uncertainty Principle simply means that one do not know the number of sound characteristic residing in a given acoustic scene recording, hence, it is prudent to assume that all the sound characteristic can be captured with a single timescale. While wavelet scattering uses the wavelet theorem to construct the time-frequency representation, it is still slightly affected by Heisenberg Uncertainty Principle as it is constraint by an averaging function which limits the maximum scaling factor the wavelet scattering can achieve. The averaging function is crucial to make the wavelet scattering coefficients invariant to translation. While one way to resolve this issue is to concatenate multiple timescales or resolution, typically called

multiresolution/multi-timescale, this will leads to irrelevancy and redundancy which might impede the model from learning correctly. In light of this, this thesis proposed the adaptation of genetic algorithm for feature selection, to select the optimal subset of the multi-timescale WS. The outcome is a reduction of feature dimension by ~50% and an improvement of classification accuracy, that position the model in the third place in DCASE 2020. This finding reaffirmed the hypothesis of the occurrence of irrelevancy and redundancy.

Lastly, large frequency dimension posts the same problem of having the possibility of irrelevancy and redundancy. In addition, this becomes a much more pressing problem when designing a low complexity model, where time is of the essence. The reduction in feature dimension means lesser convolution operations is required, hence, this has a direct impact on the time complexity. The solution to this problem is rather straight forwards as GAFS can be used to tackle this problem. Similarly, the outcome is positive too, a reduction in feature dimension by ~27% with a slight improvement in accuracy.

Touching on low complexity model, the reason for this application is due to the advancement of technology towards IoT and smart devices. Hence, there is an upward trend towards developing low complexity models such that it can be deployed into a less computational powerful hardware. While GAFS is used to reduce the time complexity, the second part of this thesis, elaborate about model compression techniques employed to reduce the size complexity. Before moving on to the proposed T-MSNet built using model compression techniques, this thesis proposed an enhanced WS called ‘Interleaved Wavelet Scattering’. The idea is a simple mixing of the first and second order with a different timescales as it is not possible to build a low complexity model with multi-timescale WS. Hence, mixing the first and second order will potentially increase the number of timescales, thus, reducing the effect of Heisenberg Uncertainty Principle. An ablation study is conducted to understand the relationship of mixing first and second order with different timescales. Notably, IWS must be mixed with the first order having a smaller timescale than second order, which is in accordance with energy dispersion theorem. Energy dispersion theorem states that more energy is transferred to the higher order as the timescale increases in size. Overall, the study shows that a well configured IWS works.

Back to T-MSNet, the proposed T-MSNet is built with a combination of mobile convolution block and shuffling modules. The pinpoint of this model is the development of sub-spectral shuffling and temporal shuffling. Unlike channel shuffling, sub-spectral shuffling and temporal shuffling is designed for time-frequency representation, where sub-spectral shuffling is shuffling along the frequency axis, and temporal shuffling is shuffling along the time axis. The

advantage of including the shuffling modules is that it does not contribute to the size of the model while improving the model ability to generalize. The experimentation result shows that sub-spectral shuffling has an edge over other shuffling modules for WS and IWS. With the correct positioned sub-spectral shuffling modules, the models are sure to show an improvement in logloss, which suggests that the model ability to generalize has improved.

## 6.2. Future works

### 6.2.1. Testing of robustness for GAFS with TSCNN framework

Besides acoustic scene classification, GAFS with TSCNN framework can also be applied to other acoustic domains or acoustic tagging tasks. In addition, the proposed GAFS for wavelet scattering should work for most of the time-frequency representation. Hence, it can be extended to other time-frequency representations such as log mel-spectrogram.

### 6.2.2. Evaluating the robustness and flexibility of sub-spectral and temporal shuffling modules

Similarly, the proposed T-MSNet can be applied to other audio tagging tasks to evaluate the robustness. In addition, our proposed sub-spectral and temporal shuffling modules are modular and can be integrated into different CNN architectures. It should not be limited to wavelet scattering; we believe this should also work for other time-frequency representations. The proposed shuffling modules can be expanded by exploring the integration with other CNN architecture and different input representations.

### 6.2.3. Better Search Optimization Approach to find the Optimal Configuration for Shuffling Modules

In this thesis, we presented a random search approach with defined criteria to find the ‘optimal’ configuration for the combination of shuffling modules. Hence, there are many possibilities to tap into search optimization techniques, such as using an evolutionary algorithm to improve finding the optimal configuration.

### 6.2.4. Accelerating GAFS with TSCNN framework

The daunting high computational runtime is one major downside of using a genetic algorithm for the feature selection wrapper approach with a convolution neural network. Hence, there is a research space to improve this aspect. Future studies can be developed into two paths: the first path is more towards creating a more efficient genetic algorithm for feature selection (e.g., ‘hybrid’ approach). The second path is rethinking the possibility of using model compression techniques to speed up the CNN, indirectly speeding up the entire GAFS system runtime.

### 6.2.4. Integration of WS with CNN

Another possibility is the integration of scattering transform as part of the algorithm in CNN. Our idea mainly touches on improving frequency transposition invariance (Andén and Mallat, 2014) on the CNN feature maps. Such as changing the CNN 'attention' modules pooling operation to scattering transform. In addition, we can add learning parameters to scale the wavelet scattering such that it becomes adaptive to how the network learns the enhanced feature. Notably, this thesis does not discuss the attention module (Hu et al., 2018) as it is not part of our proposed model design. In simplicity, the attention module extracts information either from the spatial dimensions, in this case, time-frequency representation, and embeds this information into the channel dimensions or vice versa. Notably, the attention module is constructed with a series of convolution operations.

# Appendix, A

RESULT OF VARIOUS COMBINATIONS OF SSS, TPS AND CS

| No | SSS | TPS | CS | Accuracy | Logloss |
|---|---|---|---|---|---|
| *1* | 4 | 0 | 4 | 67.67 | 1.437 |
| *2* | 34 | 0 | 34 | 68.90 | 1.292 |
| *3* | 0 | 4 | 234 | 67.75 | 1.375 |
| *4* | 2 | 34 | 3 | **70.60** | 1.162 |
| *5* | 234 | 0 | 234 | 70.39 | **1.148** |
| *6* | 234 | 0 | 4 | 70.25 | 1.149 |
| *7* | 2 | 34 | 0 | 70.24 | 1.171 |
| *8* | 234 | 0 | 23 | 70.15 | 1.159 |
| *9* | 2 | 4 | 3 | 70.15 | 1.179 |
| *10* | 2 | 0 | 3 | 70.12 | 1.181 |
| *11* | 234 | 34 | 234 | 70.00 | 1.149 |
| *12* | 23 | 0 | 1234 | 69.88 | 1.163 |
| *13* | 23 | 0 | 234 | 69.84 | 1.158 |
| *14* | 2 | 0 | 234 | 69.77 | 1.189 |
| *15* | 23 | 0 | 4 | 69.52 | 1.168 |
| *16* | 2 | 0 | 4 | 69.52 | 1.176 |
| *17* | 2 | 234 | 3 | 69.50 | 1.166 |
| *18* | 23 | 0 | 24 | 69.48 | 1.169 |
| *19* | 234 | 0 | 34 | 69.14 | 1.163 |
| *20* | 234 | 0 | 34 | 69.14 | 1.163 |
| *21* | 234 | 34 | 0 | 68.87 | 1.162 |
| *22* | 123 | 0 | 1234 | 67.66 | 1.174 |
| *23* | 1234 | 0 | 1234 | 67.43 | 1.181 |

**Table Appendix A**. Result of various combinations of SSS, TPS and CS. This table presents 20 different combinations from random search. The first 3 examples demonstrated the performances result if the rules are not followed. Their logloss result is not competitive. The rest of the table is then arranged based on classification accuracy in descending order (meaning the top performance models will be placed at the top of the table).

# Reference


ABADI, M., AGARWAL, A., BARHAM, P., BREVDO, E., CHEN, Z., CITRO, C., CORRADO, G. S., DAVIS, A., DEAN, J. & DEVIN, M. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*.

ABEßER, J. 2020. A Review of Deep Learning Based Methods for Acoustic Scene Classification. *Applied Sciences,* 10.

AHMED, E., SAINT, A., EL RAHMAN SHABAYEK, A., CHERENKOVA, K., DAS, R., GUSEV, G., AOUADA, D. & OTTERSTEN, B. 2018. A survey on Deep Learning Advances on Different 3D Data Representations. Available: https://ui.adsabs.harvard.edu/abs/2018arXiv180801462A [Accessed August 01, 2018].

ALEXANDRE, E., CUADRA, L., ROSA, M. & LOPEZ-FERRERAS, F. 2007. Feature Selection for Sound Classification in Hearing Aids Through Restricted Search Driven by Genetic Algorithms. *IEEE Transactions on Audio, Speech, and Language Processing,* 15**,** 2249-2256.

ALZUBAIDI, L., ZHANG, J., HUMAIDI, A. J., AL-DUJAILI, A., DUAN, Y., AL-SHAMMA, O., SANTAMARÍA, J., FADHEL, M. A., AL-AMIDIE, M. & FARHAN, L. 2021. Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. *Journal of big data,* 8**,** 53-53.

ANDÉN, J. & MALLAT, S. 2011. *Multiscale Scattering for Audio Classification*.

ANDÉN, J. & MALLAT, S. 2014. Deep scattering spectrum. *IEEE Transactions on Signal Processing,* 62**,** 4114-4128.

ANDREUX, M., ANGLES, T., EXARCHAKISGEO, G., LEONARDU, R., ROCHETTE, G., THIRY, L., ZARKA, J., MALLAT, S., ANDÉN, J., BELILOVSKY, E., BRUNA, J., LOSTANLEN, V., CHAUDHARY, M., HIRN, M. J., OYALLON, E., ZHANG, S., CELLA, C. & EICKENBERG, M. 2020. Kymatio: scattering transforms in Python. *J. Mach. Learn. Res.,* 21**,** Article 60.

ANSARI, G. J., SHAH, J. H., FARIAS, M. C. Q., SHARIF, M., QADEER, N. & KHAN, H. U. 2021. An Optimized Feature Selection Technique in Diversified Natural Scene Text for Classification Using Genetic Algorithm. *IEEE Access,* 9**,** 54923-54937.

ARANDJELOVIC, R. & ZISSERMAN, A. Look, Listen and Learn. 2017 IEEE International Conference on Computer Vision (ICCV), 22-29 Oct. 2017 2017. 609-617.

AYTAR, Y., VONDRICK, C. & TORRALBA, A. 2016. Soundnet: Learning sound representations from unlabeled video. *Advances in neural information processing systems,* 29.

BA, J. L., KIROS, J. R. & HINTON, G. E. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

BARCHIESI, D., GIANNOULIS, D., STOWELL, D. & PLUMBLEY, M. D. 2015. Acoustic Scene Classification: Classifying environments from the sounds they produce. *IEEE Signal Processing Magazine,* 32**,** 16-34.

BLUM, A. L. & LANGLEY, P. 1997. Selection of relevant features and examples in machine learning. *Artificial Intelligence,* 97**,** 245-271.

BREGMAN, A. S. & MCADAMS, S. 1994. Auditory Scene Analysis: The Perceptual Organization of Sound. *The Journal of the Acoustical Society of America,* 95**,** 1177-1178.

BRUNA, J. & MALLAT, S. 2013. Invariant Scattering Convolution Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 35**,** 1872-1886.

CHANDRAKALA, S. & JAYALAKSHMI, S. L. 2019. Environmental Audio Scene and Sound Event Recognition for Autonomous Surveillance: A Survey and Comparative Studies. *ACM Comput. Surv.,* 52**,** Article 63.

CHANG, S., PARK, H., CHO, J., PARK, H., YUN, S. & HWANG, K. Subspectral Normalization for Neural Audio Data Processing. ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 6-11 June 2021 2021. 850-854.

CHENG, Y., WANG, D., ZHOU, P. & ZHANG, T. 2017. A survey of model compression and acceleration for deep neural networks. *arXiv preprint arXiv:1710.09282*.

CHOI, S., KIM, T., JEONG, M., PARK, H. & KIM, C. 2021. Meta Batch-Instance Normalization for Generalizable Person Re-Identification. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)***,** 3424-3434.

CLARKSON, B., SAWHNEY, N. & PENTL, A. 1998. Auditory Context Awareness via Wearable Computing.
CLEVERT, D.-A., UNTERTHINER, T. & HOCHREITER, S. 2016. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). *arXiv: Learning*.
DASH, M. & LIU, H. 1997. Feature selection for classification. *Intelligent Data Analysis,* 1**,** 131-156.
DCASE. *Detection and Classification of Acoustic Scenes and Events* [Online]. Available: https://dcase.community/ [Accessed].
DE LA IGLESIA, B. 2013. Evolutionary computation for feature selection in classification problems. *WIREs Data Mining and Knowledge Discovery,* 3**,** 381-407.
DELIANG, W. & GUY, J. B. 2006. Fundamentals of Computational Auditory Scene Analysis. *Computational Auditory Scene Analysis: Principles, Algorithms, and Applications.* IEEE.
DENG, J., DONG, W., SOCHER, R., LI, L. J., KAI, L. & LI, F.-F. ImageNet: A large-scale hierarchical image database. 2009 IEEE Conference on Computer Vision and Pattern Recognition, 20-25 June 2009 2009. 248-255.
DUZDEVICH, D. & DARWIN, C. 2014. Darwin's On the origin of species : a modern rendition. Bloomington and Indianapolis: Indiana University Press,.
FAN, Y., XIAN, Y., LOSCH, M. M. & SCHIELE, B. Analyzing the Dependency of ConvNets on Spatial Information. 2021 Cham. Springer International Publishing, 101-115.
GAO, W. & MCDONNELL, M. D. 2020. ACOUSTIC SCENE CLASSIFICATION USING DEEP RESIDUAL NETWORKS

WITH FOCAL LOSS AND MILD DOMAIN ADAPTATION. Detection and Classification of Acoustic Scenes and Events 2020.
GEMMEKE, J. F., ELLIS, D. P., FREEDMAN, D., JANSEN, A., LAWRENCE, W., MOORE, R. C., PLAKAL, M. & RITTER, M. Audio set: An ontology and human-labeled dataset for audio events. 2017 IEEE international conference on acoustics, speech and signal processing (ICASSP), 2017. IEEE, 776-780.
GHARIB, S., DERRAR, H., NIIZUMI, D., SENTTULA, T., TOMMOLA, J., HEITTOLA, T., VIRTANEN, T. & HUTTUNEN, H. ACOUSTIC SCENE CLASSIFICATION: A COMPETITION REVIEW. 2018 IEEE 28th International Workshop on Machine Learning for Signal Processing (MLSP), 17-20 Sept. 2018 2018. 1-6.
GOLDBERG, D. E. Genetic Algorithms in Search Optimization and Machine Learning. 1988.
GOLDBERG, D. E. 2000. The Design of Innovation: Lessons from Genetic Algorithms, Lessons for the Real World. *Technological Forecasting and Social Change,* 64**,** 7-12.
GOLDBERG, D. E. & SHAKESPEARE, W. Genetic Algorithms. 2002.
GONG, Y., CHUNG, Y.-A. & GLASS, J. R. AST: Audio Spectrogram Transformer. Interspeech, 2021.
GOODFELLOW, I., BENGIO, Y. & COURVILLE, A. 2016. *Deep learning*, MIT press.
GUZHOV, A., RAUE, F., HEES, J. & DENGEL, A. Audioclip: Extending Clip to Image, Text and Audio. ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 23-27 May 2022 2022. 976-980.
HAHNLOSER, R. H. R., SARPESHKAR, R., MAHOWALD, M. A., DOUGLAS, R. J. & SEUNG, H. S. 2000. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature,* 405**,** 947-951.
HAN, X., ZHANG, Z., DING, N., GU, Y., LIU, X., HUO, Y., QIU, J., YAO, Y., ZHANG, A., ZHANG, L., HAN, W., HUANG, M., JIN, Q., LAN, Y., LIU, Y., LIU, Z., LU, Z., QIU, X., SONG, R., TANG, J., WEN, J.-R., YUAN, J., ZHAO, W. X. & ZHU, J. 2021. Pre-trained models: Past, present and future. *AI Open,* 2**,** 225-250.
HE, K., ZHANG, X., REN, S. & SUN, J. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. 2015 IEEE International Conference on Computer Vision (ICCV), 7-13 Dec. 2015 2015. 1026-1034.
HE, K., ZHANG, X., REN, S. & SUN, J. Deep Residual Learning for Image Recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 27-30 June 2016 2016a. 770-778.

HE, K., ZHANG, X., REN, S. & SUN, J. Identity Mappings in Deep Residual Networks. *In:* LEIBE, B., MATAS, J., SEBE, N. & WELLING, M., eds. Computer Vision – ECCV 2016, 2016// 2016b Cham. Springer International Publishing, 630-645.

HEITTOLA, T., MESAROS, A. & VIRTANEN, T. 2020. Acoustic scene classification in dcase 2020 challenge: generalization across devices and low complexity solutions. *arXiv preprint arXiv:2005.14623*.

HINTON, G. E., VINYALS, O. & DEAN, J. 2015. Distilling the Knowledge in a Neural Network. *ArXiv,* abs/1503.02531.

HOLLAND, J. H. 1984. Genetic Algorithms and Adaptation. *In:* SELFRIDGE, O. G., RISSLAND, E. L. & ARBIB, M. A. (eds.) *Adaptive Control of Ill-Defined Systems.* Boston, MA: Springer US.

HONG, J.-H. & CHO, S.-B. 2006. Efficient huge-scale feature selection with speciated genetic algorithm. *Pattern Recognition Letters,* 27**,** 143-150.

HOWARD, A., SANDLER, M., CHEN, B., WANG, W., CHEN, L. C., TAN, M., CHU, G., VASUDEVAN, V., ZHU, Y., PANG, R., ADAM, H. & LE, Q. Searching for MobileNetV3. 2019 IEEE/CVF International Conference on Computer Vision (ICCV), 27 Oct.-2 Nov. 2019 2019. 1314-1324.

HOWARD, A. G., ZHU, M., CHEN, B., KALENICHENKO, D., WANG, W., WEYAND, T., ANDREETTO, M. & ADAM, H. 2017. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *ArXiv,* abs/1704.04861.

HU, H., YANG, C.-H. H., XIA, X., BAI, X., TANG, X., WANG, Y., NIU, S., CHAI, L., LI, J. & ZHU, H. 2020. Device-robust acoustic scene classification based on two-stage categorization and data augmentation. *arXiv preprint arXiv:2007.08389*.

HUAN, H., LI, P., ZOU, N., WANG, C., XIE, Y., XIE, Y. & XU, D. 2021. End-to-End Super-Resolution for Remote-Sensing Images Using an Improved Multi-Scale Residual Network. *Remote Sensing,* 13**,** 666.

HUANG, G., LIU, S., MAATEN, L. V. D. & WEINBERGER, K. Q. 2018. CondenseNet: An Efficient DenseNet Using Learned Group Convolutions. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition***,** 2752-2761.

HÜWEL, A., ADILOĞLU, K. & BACH, J. H. Hearing aid Research Data Set for Acoustic Environment Recognition. ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 4-8 May 2020 2020. 706-710.

IANDOLA, F. N., MOSKEWICZ, M. W., ASHRAF, K., HAN, S., DALLY, W. J. & KEUTZER, K. 2016. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <1MB model size. *ArXiv,* abs/1602.07360.

IBRAHIM, A. B., SEDDIQ, Y. M., MEFTAH, A. H., ALGHAMDI, M., SELOUANI, S. A., QAMHAN, M. A., ALOTAIBI, Y. A. & ALSHEBEILI, S. A. 2020. Optimizing Arabic Speech Distinctive Phonetic Features and Phoneme Recognition Using Genetic Algorithm. *IEEE Access,* 8**,** 200395-200411.

IOFFE, S. & SZEGEDY, C. 2015. Batch normalization: accelerating deep network training by reducing internal covariate shift. *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37.* Lille, France: JMLR.org.

JACOB, B., KLIGYS, S., CHEN, B., ZHU, M., TANG, M., HOWARD, A., ADAM, H. & KALENICHENKO, D. Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 18-23 June 2018 2018. 2704-2713.

JAITLY, N. & HINTON, G. Learning a better representation of speech soundwaves using restricted boltzmann machines. 2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 22-27 May 2011 2011. 5884-5887.

JIA-CHING, W., JHING-FA, W., KUOK WAI, H. & CHENG-SHU, H. Environmental Sound Classification using Hybrid SVM/KNN Classifier and MPEG-7 Audio Low-Level Descriptor. The 2006 IEEE International Joint Conference on Neural Network Proceedings, 16-21 July 2006 2006. 1731-1735.

JUNG, M. & ZSCHEISCHLER, J. A Guided Hybrid Genetic Algorithm for Feature Selection with Expensive Cost Functions. ICCS, 2013.

KEK, X. Y., CHIN, C. S. & LI, Y. An Investigation on Multiscale Normalised Deep Scattering Spectrum with Deep Residual Network for Acoustic Scene Classification. 2021 IEEE/ACIS 22nd International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), 24-26 Nov. 2021 2021. 29-36.

KEK, X. Y., CHIN, C. S. & LI, Y. 2022. Multi-Timescale Wavelet Scattering With Genetic Algorithm Feature Selection for Acoustic Scene Classification. *IEEE Access,* 10**,** 25987-26001.

KHAN, A., SOHAIL, A., ZAHOORA, U. & QURESHI, A. S. 2020. A survey of the recent architectures of deep convolutional neural networks. *Artificial Intelligence Review,* 53**,** 5455-5516.

KIM, B., YANG, S., KIM, J. & CHANG, S. QTI Submission to DCASE 2021: residual normalization for device-imbalanced acoustic scene classification with efficient design. 2022.

KINGMA, D. P. & BA, J. 2015. Adam: A Method for Stochastic Optimization. *CoRR,* abs/1412.6980.

KLAMBAUER, G., UNTERTHINER, T., MAYR, A. & HOCHREITER, S. 2017. Self-normalizing neural networks. *Proceedings of the 31st International Conference on Neural Information Processing Systems.* Long Beach, California, USA: Curran Associates Inc.

KOUTINI, K., EGHBAL-ZADEH, H. & WIDMER, G. 2019. *CP-JKU Submissions to DCASE'19: Acoustic Scene Classification and Audio Tagging with Receptive-Field-Regularized CNNs*.

KRIZHEVSKY, A., SUTSKEVER, I. & HINTON, G. E. 2017. ImageNet classification with deep convolutional neural networks. *Commun. ACM,* 60**,** 84–90.

LECUN, Y., JACKEL, L., CORTES, C., DENKER, J., DRUCKER, H., GUYON, I., MULLER, U., SACKINGER, E., SIMARD, P. & VAPNIK, V. 2000. Learning Algorithms For Classification: A Comparison On Handwritten Digit Recognition. *The Statistical Mechanics Perspective*.

LI, Z., HOU, Y., XIE, X., LI, S., ZHANG, L., DU, S. & LIU, W. Multi-level Attention Model with Deep Scattering Spectrum for Acoustic Scene Classification. 2019 IEEE International Conference on Multimedia & Expo Workshops (ICMEW), 8-12 July 2019 2019. 396-401.

LIN, M., CHEN, Q. & YAN, S. 2013. Network in network. *arXiv preprint arXiv:1312.4400*.

LIN, T.-Y., MAIRE, M., BELONGIE, S., HAYS, J., PERONA, P., RAMANAN, D., DOLLÁR, P. & ZITNICK, C. L. Microsoft coco: Common objects in context. European conference on computer vision, 2014. Springer, 740-755.

LIU, H. & ZHAO, Z. 2014. Manipulating Data and Dimension Reduction Methods: Feature Selection. *In:* MEYERS, R. A. (ed.) *Encyclopedia of Complexity and Systems Science.* Berlin, Heidelberg: Springer Berlin Heidelberg.

LÓPEZ-CIFUENTES, A., ESCUDERO-VIÑOLO, M., BESCÓS, J. & GARCÍA-MARTÍN, Á. 2020. Semantic-aware scene recognition. *Pattern Recognition,* 102**,** 107256.

LOPEZ-MEYER, P., ONTIVEROS, J. A. D. H., LU, H. & STEMMER, G. Efficient End-to-End Audio Embeddings Generation for Audio Classification on Target Applications. ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 6-11 June 2021 2021. 601-605.

LOSHCHILOV, I. & HUTTER, F. 2017. SGDR: Stochastic Gradient Descent with Warm Restarts. *arXiv: Learning*.

MA, N., ZHANG, X., ZHENG, H.-T. & SUN, J. ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design. 2018 Cham. Springer International Publishing, 122-138.

MALLAT, S. 2012. Group Invariant Scattering. *Communications on Pure and Applied Mathematics,* 65**,** 1331-1398.

MALLAT, S. G. 2009. *A wavelet tour of signal processing : the sparse way,* Amsterdam ; Boston, Elsevier/Academic Press.

MART'IN-MORAT'O, I., HEITTOLA, T., MESAROS, A. & VIRTANEN, T. Low-Complexity Acoustic Scene Classification for Multi-Device Audio: Analysis of DCASE 2021 Challenge Systems. DCASE, 2021.

MATURANA, D. & SCHERER, S. VoxNet: A 3D Convolutional Neural Network for real-time object recognition. 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 28 Sept.-2 Oct. 2015 2015. 922-928.

MCDONNELL, M. D. & GAO, W. Acoustic Scene Classification Using Deep Residual Networks with Late Fusion of Separated High and Low Frequency Paths. ICASSP 2020 - 2020 IEEE International

Conference on Acoustics, Speech and Signal Processing (ICASSP), 4-8 May 2020 2020. 141-145.
MCFEE, B., RAFFEL, C., LIANG, D., ELLIS, D. P. W., MCVICAR, M., BATTENBERG, E. & NIETO, O. librosa: Audio and Music Signal Analysis in Python. 2015.
MESAROS, A., HEITTOLA, T. & VIRTANEN, T. Acoustic Scene Classification: An Overview of Dcase 2017 Challenge Entries. 2018 16th International Workshop on Acoustic Signal Enhancement (IWAENC), 17-20 Sept. 2018 2018. 411-415.
MISRA, D. Mish: A Self Regularized Non-Monotonic Activation Function. BMVC, 2020.
MOHAIMENUZZAMAN, M., BERGMEIR, C. & MEYER, B. 2022. Pruning vs XNOR-Net: A Comprehensive Study of Deep Learning for Audio Classification on Edge-Devices. *IEEE Access,* 10**,** 6696-6707.
MOHAIMENUZZAMAN, M., BERGMEIR, C., WEST, I. T. & MEYER, B. 2021. Environmental Sound Classification on the Edge: Deep Acoustic Networks for Extremely Resource-Constrained Devices. *ArXiv,* abs/2103.03483.
MOLCHANOV, P., TYREE, S., KARRAS, T., AILA, T. & KAUTZ, J. 2016. Pruning Convolutional Neural Networks for Resource Efficient Transfer Learning. *ArXiv,* abs/1611.06440.
MOROZOV, S. P., ANDREYCHENKO, A. E., PAVLOV, N. A., VLADZYMYRSKYY, A. V., LEDIKHOVA, N. V., GOMBOLEVSKIY, V. A., BLOKHIN, I. A., GELEZHE, P. B., GONCHAR, A. V. & CHERNINA, V. Y. 2020. MosMedData: Chest CT Scans with COVID-19 Related Findings Dataset. *medRxiv***,** 2020.05.20.20100362.
NGUYEN, M. H., NGUYEN, P. L., NGUYEN, K., LE, V. A., NGUYEN, T. H. & JI, Y. 2021. PM2.5 Prediction Using Genetic Algorithm-Based Feature Selection and Encoder-Decoder Model. *IEEE Access,* 9**,** 57338-57350.
ORESKI, S. & ORESKI, G. 2014. Genetic algorithm-based heuristic for feature selection in credit risk assessment. *Expert Systems with Applications,* 41**,** 2052-2064.
PASZKE, A., GROSS, S., MASSA, F., LERER, A., BRADBURY, J., CHANAN, G., KILLEEN, T., LIN, Z., GIMELSHEIN, N. & ANTIGA, L. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems,* 32.
PEDDINTI, V., SAINATH, T., MAYMON, S., RAMABHADRAN, B., NAHAMOO, D. & GOEL, V. Deep Scattering Spectrum with deep neural networks. 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 4-9 May 2014 2014. 210-214.
PHAYE, S. S. R., BENETOS, E. & WANG, Y. SubSpectralNet – Using Sub-spectrogram Based Convolutional Neural Networks for Acoustic Scene Classification. ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 12-17 May 2019 2019. 825-829.
PICZAK, K. J. 2015. ESC: Dataset for Environmental Sound Classification. *Proceedings of the 23rd ACM international conference on Multimedia.* Brisbane, Australia: Association for Computing Machinery.
POLIKAR, R. 2006. *The Wavelet Tutorial* [Online]. Rowan University, College of Engineering Web Servers. Available: https://cseweb.ucsd.edu/~baden/Doc/wavelets/polikar_wavelets.pdf [Accessed 2022].
RAMACHANDRAN, P., ZOPH, B. & LE, Q. V. 2017. Searching for activation functions. *arXiv preprint arXiv:1710.05941*.
REN, Z., QIAN, K., ZHANG, Z., PANDIT, V., BAIRD, A. & SCHULLER, B. 2018. Deep Scalogram Representations for Acoustic Scene Classification. *IEEE/CAA Journal of Automatica Sinica,* 5**,** 662-669.
RUMELHART, D. E., HINTON, G. E. & WILLIAMS, R. J. 1986. Learning representations by back-propagating errors. *Nature,* 323**,** 533-536.
RUSSAKOVSKY, O., DENG, J., SU, H., KRAUSE, J., SATHEESH, S., MA, S., HUANG, Z., KARPATHY, A., KHOSLA, A., BERNSTEIN, M., BERG, A. C. & FEI-FEI, L. 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision,* 115**,** 211-252.
SAEYS, Y., INZA, I. & LARRAÑAGA, P. 2007. A review of feature selection techniques in bioinformatics. *Bioinformatics,* 23**,** 2507-17.

SALAMON, J. & BELLO, J. P. Feature learning with deep scattering for urban sound analysis. 2015 23rd European Signal Processing Conference (EUSIPCO), 31 Aug.-4 Sept. 2015 2015. 724-728.
SALAMON, J., JACOBY, C. & BELLO, J. P. 2014. A Dataset and Taxonomy for Urban Sound Research. *Proceedings of the 22nd ACM international conference on Multimedia.* Orlando, Florida, USA: Association for Computing Machinery.
SAWHNEY, N. & MAES, P. Situational Awareness from Environmental Sounds. 1997.
SHAHREZAEI, I. H. & KIM, H. C. 2020. Fractal Analysis and Texture Classification of High-Frequency Multiplicative Noise in SAR Sea-Ice Images Based on a Transform- Domain Image Decomposition Method. *IEEE Access,* 8**,** 40198-40223.
SHELHAMER, E., LONG, J. & DARRELL, T. 2017. Fully Convolutional Networks for Semantic Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 39**,** 640-651.
SIEDLECKI, W. & SKLANSKY, J. 1989. A note on genetic algorithms for large-scale feature selection. *Pattern Recognition Letters,* 10**,** 335-347.
SIEDLECKI, W. & SKLANSKY, J. 1993. A note on genetic algorithms for large-scale feature selection. *Handbook of pattern recognition & computer vision.* World Scientific Publishing Co., Inc.
SIFRE, L. & MALLAT, S. 2014. Rigid-motion scattering for texture classification. *arXiv preprint arXiv:1403.1687*.
SIMONYAN, K. & ZISSERMAN, A. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.
SLOSS, A. N. & GUSTAFSON, S. 2020. 2019 Evolutionary Algorithms Review. *In:* BANZHAF, W., GOODMAN, E., SHENEMAN, L., TRUJILLO, L. & WORZEL, B. (eds.) *Genetic Programming Theory and Practice XVII.* Cham: Springer International Publishing.
SPOLAÔR, N., LORENA, A. C. & LEE, H. D. Multi-objective Genetic Algorithm Evaluation in Feature Selection. 2011 Berlin, Heidelberg. Springer Berlin Heidelberg, 462-476.
SRIVASTAVA, R. K., GREFF, K. & SCHMIDHUBER, J. 2015. Highway networks. *arXiv preprint arXiv:1505.00387*.
SUH, S., PARK, S., JEONG, Y. & LEE, T. Designing Acoustic Scene Classification Models with CNN Variants Technical Report. 2020.
SZEGEDY, C., WEI, L., YANGQING, J., SERMANET, P., REED, S., ANGUELOV, D., ERHAN, D., VANHOUCKE, V. & RABINOVICH, A. Going deeper with convolutions. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 7-12 June 2015 2015. 1-9.
TANG, J., ALELYANI, S. & LIU, H. Feature Selection for Classification: A Review. Data Classification: Algorithms and Applications, 2014.
TCHORZ, J., WOLLERMANN, S. & HUSSTEDT, H. Classification of Environmental Sounds for Future Hearing Aid Applications. 2017.
ULYANOV, D., VEDALDI, A. & LEMPITSKY, V. 2016. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*.
UYSAL, A. K. 2018. On Two-Stage Feature Selection Methods for Text Classification. *IEEE Access,* 6**,** 43233-43251.
UZKENT, B., BARKANA, B. & CEVIKALP, H. 2012. Non-speech environmental sound classification using SVMs with a new set of features. *International Journal of Innovative Computing, Information and Control,* 8.
VALENTI, M., SQUARTINI, S., DIMENT, A., PARASCANDOLO, G. & VIRTANEN, T. A convolutional neural network approach for acoustic scene classification. 2017 International Joint Conference on Neural Networks (IJCNN), 14-19 May 2017 2017. 1547-1554.
VERGARA, J. R. & ESTÉVEZ, P. A. 2014. A review of feature selection methods based on mutual information. *Neural Computing and Applications,* 24**,** 175-186.
XIE, S., GIRSHICK, R., DOLLÁR, P., TU, Z. & HE, K. Aggregated residual transformations for deep neural networks. Proceedings of the IEEE conference on computer vision and pattern recognition, 2017. 1492-1500.
XUE, B., CERVANTE, L., SHANG, L., BROWNE, W. N. & ZHANG, M. 2013. MULTI-OBJECTIVE EVOLUTIONARY ALGORITHMS FOR FILTER BASED FEATURE SELECTION IN CLASSIFICATION. *International Journal on Artificial Intelligence Tools,* 22**,** 1350024.

XUE, B., ZHANG, M., BROWNE, W. N. & YAO, X. 2016. A Survey on Evolutionary Computation Approaches to Feature Selection. *IEEE Transactions on Evolutionary Computation,* 20**,** 606-626.

YANG, C.-H. H., HU, H., SINISCALCHI, S. M., WANG, Q., WANG, Y., XIA, X., ZHAO, Y., WU, Y., WANG, Y., DU, J. & LEE, C.-H. 2021. A Lottery Ticket Hypothesis Framework for Low-Complexity Device-Robust Neural Acoustic Scene Classification. *ArXiv,* abs/2107.01461.

YANG, J. & HONAVAR, V. 1998. Feature subset selection using a genetic algorithm. *IEEE Intelligent Systems and their Applications,* 13**,** 44-49.

ZHANG, H., CISSÉ, M., DAUPHIN, Y. & LOPEZ-PAZ, D. 2018a. mixup: Beyond Empirical Risk Minimization. *ArXiv,* abs/1710.09412.

ZHANG, P., CHEN, H., BAI, H. & YUAN, Q. 2019. Deep Scattering Spectra with Deep Neural Networks for Acoustic Scene Classification Tasks. *Chinese Journal of Electronics*.

ZHANG, T., DING, B., ZHAO, X. & YUE, Q. 2018b. A Fast Feature Selection Algorithm Based on Swarm Intelligence in Acoustic Defect Detection. *IEEE Access,* 6**,** 28848-28858.

ZHANG, X., ZHOU, X., LIN, M. & SUN, J. ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 18-23 June 2018 2018c. 6848-6856.